An Introduction to

FFmpeg, Timelapse and Fulldome Video Production,

Special Effects, Color Grading, Streaming, Audio Processing,

Canon 5D-MK4, Panasonic LUMIX GH5S, Kodak PIXPRO SP360 4K, Ricoh Theta V,

Synthesizers and Audio Production,

Image Processing and Astronomy Software

by

Michael Koch, astroelectronic@t-online.de

Version from November 23, 2020
Contents

1 Introduction to FFmpeg ................................................................. 7
  1.1 What can be done with FFmpeg? .................................................. 9
  1.2 If FFmpeg has no graphical user interface, how do we use it? ...10
  1.3 The first example ...................................................................... 12
  1.4 Using variables .......................................................................... 13
2 FFmpeg in detail ............................................................................ 14
  2.1 Convert from one video format to another video format ..........14
  2.2 Fit timelapse length to music length ......................................... 15
  2.3 Timelapse or slideshow from many images, with crossfading ...16
  2.4 Slideshow with different durations ........................................... 18
  2.5 Slideshow with scrolling images .............................................. 18
  2.6 Extract many images from a video ............................................ 21
  2.7 Extract the last frame from a video ........................................... 21
  2.8 Modify brightness, contrast, saturation, gamma and hue .......22
  2.9 Strong contrast enhancement .................................................... 23
  2.10 Inverting a video or image (make a negative) ..........24
  2.11 Correcting the color channels ................................................. 25
  2.12 Colorhold ................................................................................ 26
  2.13 Atmospheric dispersion correction ........................................... 27
  2.14 Amplify filter ........................................................................... 27
  2.15 Extract a time segment from a video ....................................... 28
  2.16 Trim filter ................................................................................ 29
  2.17 Tpad filter, add a few seconds black at the beginning or end ..30
  2.18 Extract the last 30 seconds of a video ...................................... 31
  2.19 Fade-in and fade-out ............................................................... 32
  2.20 Crossfading ............................................................................. 33
  2.21 Crop a video ............................................................................ 34
  2.22 Zoompan ................................................................................ 35
  2.23 Changing the speed: slow motion and timelapse .................36
  2.24 Slow motion or timelapse only for a segment of the video ....37
  2.25 Time Remapping ................................................................... 38
  2.26 Insert a text which is visible for the whole duration .........40
  2.27 Slowly fade a text in and out .................................................... 40
  2.28 Show a running clock in the video .......................................... 43
  2.29 Generation of curved text for fulldome projection ..........45
  2.30 Write text on a transparent layer ............................................ 48
  2.31 Combine multiple videos with concat demuxer ...............49
  2.32 Combine multiple videos with concat filter ....................50
  2.33 Switch between two cameras, using audio from camera1 ....51
  2.34 Stack videos side by side (or on top of each other) .......52
  2.35 Horizontal and vertical flipping ............................................. 52
  2.36 Stack four videos to a 2x2 mosaic ........................................ 53
  2.37 Blink comparator .................................................................. 54
  2.38 Animated GIF ........................................................................ 55
  2.39 Replace one frame in a video by another .........................56
  2.40 Blend filter ............................................................................ 57
  2.41 Circular mask (View through eyepiece) .........................58
  2.42 Binocular view simulation .................................................... 60
  2.43 Subtracting a darkframe ......................................................... 61
  2.44 Gradation curves and vignetting .......................................... 62
  2.45 Color grading with color look-up tables, full workflow ......63
  2.46 Color grading with color look-up tables, simplified workflow 65
  2.47 Size of color-look-up table ................................................... 66
  2.48 Histogram ............................................................................. 71
  2.49 Lagfun filter ......................................................................... 71
  2.50 Star trails ................................................................................ 73
  2.51 Bird trails ............................................................................... 74
  2.52 Rainbow-trail effect ............................................................... 76
  2.53 Temporal slice-stacking effect ............................................. 77
  2.54 Extract and merge planes, split planes .........................79
  2.55 Bluescreen / greenscreen ..................................................... 80
  2.56 Real-time bluescreening ....................................................... 91
  2.57 Datascope ............................................................................ 116
  2.58 Video line jitter effect .......................................................... 117
Panasonic LUMIX GH5S
Cameras and lenses for fulldome video production

GH5S Luminance level
GH5S Recommended settings
GH5S Autofocus
GH5S Metering Modes
GH5S Exposing for VLog-L
GH5S Record formats

Canon 5D-Mark4

GH5S, all 4K 10 bit modes
GH5S, all anamorphic 8 bit modes
GH5S, all anamorphic 10 bit modes
GH5S, all FHD 8 bit modes
GH5S, all FHD 10 bit modes

All Canon 5D-Mark4 video modes for PAL video system
All Canon 5D-Mark4 video modes for NTSC video system
Video tutorials for Canon 5D-Mark4
Favorable camera / fisheye combinations
Read-out chip size of cameras at different video modes
GH5S Mechanical / electronic shutter
GH5S, all 4K 8 bit modes
GH5S, all C4K 10 bit modes
GH5S, all C4K 8 bit modes
GH5S, all 77 video modes
GH5S Field of view with and without SpeedBooster 0.64x
GH5S, all 77 video modes
GH5S, all C4K 8 bit modes
GH5S, all C4K 10 bit modes
GH5S, all 4K 8 bit modes
1 Introduction to FFmpeg

FFmpeg is an open-source software and available for Linux, Windows and OS-X. It's a very powerful command line tool and has no graphic user interface.

Main project website:  www.ffmpeg.org
Download site for the Windows builds:  https://www.gyan.dev/ffmpeg/builds/

How to install the Windows build:
Download the "git essentials" version, open the ZIP file, open the "bin" folder and copy ffmpeg.exe, ffprobe.exe and ffplay.exe to a new folder, for example to c:\ffmpeg\ That's all.
In rare cases, if you need some special libraries, you might want to download the "git full" version instead. But you won't need it for the examples in this book.
An alternative download site is here:  https://github.com/BtbN/FFmpeg-Builds/releases

ffmpeg.exe is the very powerful software for manipulating videos.
ffprobe.exe is a program for viewing the properties of videos, pictures or audio files. It's useful for troubleshooting.
ffplay.exe is a video player. In most cases we don't need it if we use the VLC player instead.
It's also a good idea to save the file doc\ffmpeg-all.html somewhere on your own computer. This file contains (almost) the full documentation for FFmpeg. The most important chapters are "Audio Filters" and "Video Filters".

Additional to the official documentation, there are also two wikis available:
https://trac.ffmpeg.org/wiki

Most examples in this document were tested with Windows 7, and beginning in March 2020 I also used Windows 10.
What are the pros and cons of FFmpeg, compared to other programs for video manipulation?

- Very powerful capabilities.
- It's an active project, updates come almost daily.
- Conversion from almost all formats to almost all other formats.
- No restrictions for video size (width * height), as in many other programs.
- There is a mailing list where you can ask questions in english. Before you ask, make sure that the problem is reproducible in the latest FFmpeg version. Always include the complete FFmpeg console output, because it contains many useful informations.
- FFmpeg is a command line program and has no graphical user interface. At first glimpse this sounds like a big drawback. But it's a nice idea to have all commands in a batch file, because later you can easily make modifications at all arbitrary steps in the workflow. Just modify the batch file and execute it again.
- You will need some time for learning FFmpeg.
- Unfortunately the documentation is the weak point of the project, and many times I wished that the documentation contained more informations and especially more examples.
- It's always a good idea to begin with a working example, and then modify it step by step. I hope that the examples in this book are a good starting point for you.
1.1 What can be done with FFmpeg?

- Convert a video, picture or sound from one format to another.
- Make a (timelapse) video from many pictures.
- Make many pictures from a video.
- Cut segments from a video, for example remove the beginning or the end.
- Add or remove audio, or modify the audio volume.
- Change the video size (width * height).
- Enlarge parts of the video or cut away the borders, for example make a rectangular video square.
- Change the speed, timelapse or slow motion.
- Rotate, mirror or flip.
- Add texts to a video.
- Correct brightness, contrast, gamma, saturation, color temperature, also with look-up-tables.
- Masking, for example superimpose a circular mask over a video.
- Fade-in, fade-out and crossfade for video and audio.
- Morphing, for example curved texts for fulldome projection, or simulation of curved spacetime near block holes.
- Stabilizing of shaky videos
- Deflicker, for reducing brightness steps in timelapse.
- Change the video compression, to make the video smaller.
- and many more interesting things...
1.2 If FFmpeg has no graphical user interface, how do we use it?

There are three possible ways:

1. Open a console window All_Programs / Accessories / Command_Promt (german) Alle_Programme / Zubehör / Eingabeaufforderung
   Another way to open the console window is to press WINDOW R and then enter "cmd".
   But this is not recommended, because in many cases the command lines are quite long.

2. In the Windows File Explorer, in the address bar, you can type cmd and press enter to get a command prompt in the directory you are currently examining.

3. The recommended way is to write a batch file which contains the FFmpeg command line.
   A batch file has the extension *.bat and can be created and modified with any text editor. When you save a batch file with Notepad, make sure that you choose "all files" and save it as *.bat and don't choose "text files", because then the extension would be *.bat.txt (Hint: Configure the explorer so that all file extensions are visible!)
   • You can edit a batch file by right clicking on it, and then choose "Edit".
   • You can execute a batch file by double clicking on the icon or filename.
   • Once you've created a batch file, you can place either it, or a short to it, on your Windows desktop. Then you can drag-and-drop one or more (depending on how you've designed it) media files onto the icon for processing by the batch file.
   • It's recommended to begin with a working example, and then modify it step by step. Make small steps and always make a test run. If it fails, go back to the last working version.
   • The % character has a special meaning inside a batch file. If you need a one % character in the FFmpeg command line, you must replace it in the batch file by two %% characters.
   • It's recommended to insert the command "pause" at the end of the batch file. This means the batch file waits for a keypress. Without this command, the console window would close immediately when FFmpeg has finished, and you wouldn't see if there were any error messages.
   • With the command "set" you can define variables in the batch file.
   • With the command "rem" you can insert comments, so that you later understand how the batch file works. Comments can also begin with :: in the same line as a command. Everything from :: to the end of the line is a comment.
   • If the command line becomes too long, you can insert a ^ character at the end of the line and continue in the next line.
• How to copy and paste the content of the console window: Right click in the title of the Command_Prompt window, Edit -> Select_All, then Edit -> Copy, then paste the content with ctrl-v somewhere else.

• (german) Wenn man den Inhalt des CMD-Fensters kopieren möchte, geht man so vor: Rechtsklick auf die Titelleiste des Fensters, Bearbeiten --> Alles auswählen, dann Bearbeiten -> Kopieren, dann mit Control-V irgendwo anders einfügen.

• If you don't want to write to full path to FFmpeg in each batch file, then you should add the path to the PATH system variable. In this article is described how to do this: https://www.computerhope.com/issues/ch000549.htm#windows10

   The following is copied from the above link:

   1. From the desktop, right-click the very bottom-left corner of the screen to get the "Power User Task Menu".
   2. From the Power User Task Menu, click System.
   3. In the Settings window, scroll down to the Related settings section and click the System info link.
   4. In the System window, click the Advanced system settings link in the left navigation pane.
   5. In the System Properties window, click the Advanced tab, then click the Environment Variables button near the bottom of that tab.
   6. In the Environment Variables window (pictured below), highlight the Path variable in the System variables section and click the Edit button. Add or modify the path lines with the paths you want the computer to access. Each different directory is separated with a semicolon, as shown below.
1.3 The first example

This is a simple batch file:

```
rem A simple batch file for making a video from many pictures

c:\ffmpeg\ffmpeg -framerate 5 -start_number 3551 -i IMG_%%4d.jpg -i birds.mp3 ^
-shortest -codec:v mpeg4 -q:v 2 out.mp4

pause :: wait for a keypress
```

What's the meaning of the parts?

<table>
<thead>
<tr>
<th>rem A simple ...</th>
<th>This is a comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>c:\ffmpeg\ffmpeg</td>
<td>This is the path to ffmpeg.exe</td>
</tr>
<tr>
<td>-framerate 5</td>
<td>This defines how fast the pictures are read in, in this case 5 pictures per second.</td>
</tr>
<tr>
<td>-start_number 3551</td>
<td>This is the number of the first picture, in this case 3551</td>
</tr>
<tr>
<td>-i IMG_%%4d.jpg</td>
<td>This is the filename of the input pictures. The term %%4d stands for a 4-digit number. The filename of the first picture is IMG_3551.jpg and the number will be increased by 1, until no more picture is found. For 3-digit numbers you would write %%3d instead.</td>
</tr>
<tr>
<td>-i birds.mp3</td>
<td>This is the second input file, in this case an audio file.</td>
</tr>
<tr>
<td>^</td>
<td>If the command line becomes too long, you can break it with the ^ character and continue in the next line. FFmpeg will get the whole line without the ^ character.</td>
</tr>
<tr>
<td>-shortest</td>
<td>This option means that the length of the output video is determined by the shortest of the two input files.</td>
</tr>
<tr>
<td>-codec:v mpeg4</td>
<td>This option means that a MPEG4 video will be produced.</td>
</tr>
<tr>
<td>-q:v 2</td>
<td>This is an option for the quality of the output video. 0 is best quality, 2 is normal, 9 is strongest compression.</td>
</tr>
<tr>
<td>out.mp4</td>
<td>Filename of the output video</td>
</tr>
<tr>
<td>pause</td>
<td>This command waits for a keypress, so that you have a chance to see any error messages before the console window closes.</td>
</tr>
<tr>
<td>:: wait for ...</td>
<td>Everything right of :: is a comment.</td>
</tr>
</tbody>
</table>

Important: Options are always written before the file they refer to.

The options "-framerate 5" and "-start_number 3551" refer to the first input file "IMG_%%4d.jpg".
The second input file "birds.mp3" doesn't have any options in this case.
The options "-shortest -codec:v mpeg4 -q:v 2" refer to the output video "out.mp4".
1.4 Using variables

Using variables is much better programming style. This batch file has exactly the same function as the first example:

```
rem A simple batch file for making a video from many pictures

set "FF=c:\ffmpeg\ffmpeg"    :: Path to ffmpeg.exe
set "FR=5"                   :: Framerate for reading in the pictures (Frames per second)
set "SN=3551"                :: Number of the first picture
set "IN=IMG_%%4d.jpg"        :: Filename of the pictures
set "AUDIO=birds.mp3"        :: Audio filename
set "QU=2"                   :: MP4 Quality, 0 is best quality, 2 is normal, 9 is strongest compression
set "OUT=out.mp4"            :: Output filename

%FF% -framerate %FR% -start_number %SN% -i %IN% -i %AUDIO% -shortest -codec:v mpeg4 -q:v %QU% %OUT%

pause                        :: wait for a keypress
```

This is much clearer, because each variable is written in a new line and has its own comment.

It's recommended to use capital letters for the variables, so that you can easily distinguish them from command line options.

All variable names are allowed, but don't use special characters like ÄÖÜ.

You can copy a batch file and save it under a new name for a new project. Then you must only set the variables, so that they fit to the new project. There is no need to modify (or even understand) the command line.

Why are the variable definitions written in " " quotation marks? This is only necessary if you want to add a comment in the same line. Without comments, the quotation marks are unnecessary.
FFmpeg in detail

2.1 Convert from one video format to another video format

Some examples for format conversion:

```plaintext
rem Convert any input format to any output format
ffmpeg -i anyinput.xxx anyoutput.xxx

rem Convert MP4 to mov
ffmpeg -i in.mp4 -acodec copy -vcodec copy -f mov out.mov

rem Convert mov to MP4
ffmpeg -i in.mov -acodec copy -vcodec copy out.mp4

rem Convert mov to MP4 using h265 compression, default preset is medium, default crf is 28
ffmpeg -i in.mov -c:v libx265 -preset slow -crf 25 -acodec copy out.mp4
```
2.2 Fit timelapse length to music length

How to give a timelapse video exactly the same length as the music?
We don't want to cut off the end of the music, and we don't want to hear silence at the end of the timelapse video.
The solution is to adjust the framerate, so that the length of the timelapse becomes equal to the music length.

Framerate = \frac{\text{Number of images}}{\text{Time in seconds}}

In this example we have 30 images and the music is 20 seconds long, so that the framerate must be 1.5.

A simple batch file for combining many images to a video

```batch
rem A simple batch file for combining many images to a video
set "FR=1.5"                 :: Framerate for reading in the images (frames per second)
set "RATE=30"                :: Output framerate
set "SN=3551"                :: Number of the first image
set "IN=IMG_%%4d.jpg"        :: Filename of the images
set "AUDIO=birds.mp3"        :: Audio filename
set "QU=2"                   :: MP4 Quality, 0 is best Quality, 2 is normal, 9 is strongest compression
set "OUT=out.mp4"            :: Output file
ffmpeg -framerate %FR% -start_number %SN% -i %IN% -i %AUDIO% -r %RATE% -shortest -codec:v mpeg4 -q:v %QU% %OUT%
pause                        :: Wait for a keypress
```

In this example we have two different framerates, which have different purpose:

- `-framerate %FR%` this is the framerate for reading in the images
- `-r %RATE%` this is the framerate of the output video.

These two framerates are totally independent from each other, and can be different. If the images are read in slower than the output framerate, FFmpeg will automatically duplicate images. If the images are read in faster, then FFmpeg will automatically skip images.
2.3 Timelapse or slideshow from many images, with crossfading

```
rem Make a timeelapse or slideshow from many images, with crossfading

set "RATE=30"    :: Output framerate
set "SN=3551"    :: Number of first image
set "IN=IMG_%%4d.jpg"  :: Filename of the images
set "W=2000"     :: Image width
set "H=1500"     :: Image height
set "QU=2"       :: MP4 Quality, 0 is best Quality, 2 is normal, 9 is strongest compression
set "OUT=out.mp4" :: Output file

:: A is the duration how long each image is shown (without crossfading), here 1.0 sec
:: B is the duration of the crossfade, here 0.5 sec
set "C=3"        :: set C = (A+B)/B (you must calculate this integer manually)
set "D=2"        :: set D = 1/B     (you must calculate this floating point value manually)

ffmpeg -start_number %SN% -i %IN% ^
   -vf zoompan=d=%C%:fps=%D%:s=%W%x%H%,framerate=fps=%RATE%:interp_start=0:interp_end=255:scene=100 ^
   -codec:v mpeg4 -q:v %QU% %OUT%

pause                        :: Wait for a keypress
```

Inside the video filter (beginning with -vf) we have in this example two filters, which are applied one after the other. The first is "zoompan" and the second is "framerate".

You must calculate the variables C and D manually, because there are no expressions allowed inside the "zoompan" filter.

Detailed explanations for this filter chain:
```
-vf zoompan=d=%C%:fps=%D%:s=%W%x%H%,framerate=%RATE%:interp_start=0:interp_end=255:scene=100
```

In this filter chain two video filters are applied consecutively, separated by a (,) comma.

1. "zoompan", with the parameters "d", "fps" and "s"
2. "framerate", with the parameters "fps", "interp_start", "interp_end", and "scene"

https://www.ffmpeg.org/ffmpeg-all.html#zoompan

The zoompan filter is here not used for zooming in, but for duplicating the frames and passing them to the next filter with a certain framerate. "d" specifies how often each frame is repeated. "fps" is the output framerate of this filter. "s" is the size of the output frames. This must be specified in most cases, because the default is 1280x720.

https://www.ffmpeg.org/ffmpeg-all.html#framerate

The framerate filter can calculate intermediate images between consecutive images. This is not a motion interpolation but a crossfade. "fps" is the output framerate. It's not required to explicitely write this parameter; you could also write framerate=fps=%RATE%:.... The remaining three parameters "interp_start", "interp_end", and "scene" specify, when interpolation is active and when not. With those values that I used (0, 255, 100), interpolation is always active.

These two filters together produce a video in which each image is shown for a certain duration, followed by a crossfade to the next image which also has a certain duration. Both durations can be choosen freely, these are the values A and B in the comments. From these values you must manually calculate the variables C and D, which are used in the command line. I haven't yet found a way to make this calculation automatically. It's possible to make calculations in the batch file, but this works only with integer precision.

If you omit the zoompan filter and use only the framerate filter, the next crossfade would immediately follow when the previous has ended. With other words: You always have a crossfade and there is no time where the image is shown without crossfade. That's why we use the trick with the zoompan filter. Now it's still the case that one crossfade follows immediately on the previous one, but now we have crossfades between identical images, because the images were duplicated by the zoompan filter. A crossfade between identical images isn't visible, of course.
2.4 Slideshow with different durations

ffmpeg -i img%4d.jpg -vf "zoompan=d=25+'50*eq(in,3)'+100*eq(in,5)" out.mp4

In this example each frame is shown one second (25 frames), except the 4th image which is shown 3 seconds (25+50 frames) and the 6th image which is shown 5 seconds (25+100 frames). Please note that the image numbering starts with 0, if not specified differently with "-start_number".

Please note that it might also be useful to specify the size of the output frames with the "s" option, because the default size is 1280x720.

2.5 Slideshow with scrolling images

Images scrolling from left to right:

```
set "IN=test%%3d.jpg"    :: Input images
set "N=6"                 :: Number of images
set "SX=400"              :: X Size
set "SY=300"              :: Y Size
set "T=5"                 :: Time in seconds for scrolling from one image to the next image
set "FPS=30"              :: Output framerate
set "OUT=out.mp4"         :: Output filename

rem Make some test images
ffmpeg -f lavfi -i testsrc2=size=%SX%x%SY%:duration=%N%:rate=1 -start_number 0 -y test%%3d.jpg

rem Make a scrolling slideshow
ffmpeg -framerate 1/%T% -start_number 1 -i %IN% -framerate 1/%T% -start_number 0 -i %IN% -filter_complex [0][1]hstack,fps=%FPS%,crop=w=iw/2:x='iw/2*(1-mod(t,%T%)/%T%)' -y %OUT%
```

pause
Images scrolling from right to left:

```bash
ffmpeg -framerate 1/%T% -start_number 0 -i %IN% -framerate 1/%T% -start_number 1 -i %IN% -filter_complex [0][1]hstack,fps=%FPS%,crop=w=iw/2:x='iw/2*mod(t,%T%)/%T%' -y %OUT%
```

Images scrolling from top to bottom:

```bash
ffmpeg -framerate 1/%T% -start_number 1 -i %IN% -framerate 1/%T% -start_number 0 -i %IN% -filter_complex [0][1]vstack,fps=%FPS%,crop=h=ih/2:y='ih/2*(1-mod(t,%T%)/%T%)' -y %OUT%
```

Images scrolling from bottom to top:

```bash
ffmpeg -framerate 1/%T% -start_number 0 -i %IN% -framerate 1/%T% -start_number 1 -i %IN% -filter_complex [0][1]vstack,fps=%FPS%,crop=h=ih/2:y='ih/2*mod(t,%T%)/%T%' -y %OUT%
```

This is similar, but now showing two images simultaneously side by side. The width of the output video is twice the width of the input images:

```bash
set "IN=test%%3d.jpg" :: Input images
set "N=6" :: Number of images
set "SX=400" :: X Size
set "SY=300" :: Y Size
set "T=5" :: Time in seconds for scrolling from one image to the next image
set /a "D=%T%*(%N%-2)" :: Total duration in seconds
set "FPS=30" :: Output framerate
set "OUT=out.mp4" :: Output filename

rem Make some test images
ffmpeg -f lavfi -i testsrc2=size=%SX%x%SY%:duration=%N%:rate=1 -start_number 0 -y test%%3d.jpg

rem Make a scrolling slideshow
ffmpeg -framerate 1/%T% -start_number 2 -i %IN% -framerate 1/%T% -start_number 0 -i %IN% -filter_complex [0][1][2]hstack=inputs=3,fps=%FPS%,crop=w=2*iw/3:x='iw/3*(1-mod(t,%T%)/%T%)' -t %D% -y %OUT%
```

pause
Note: "set /a" is a Windows batch command and calculates a variable (in this case: the total duration of the output video). Only integer arithmetic is possible, no floating point. This is necessary in this batch file, because the "-t" option doesn't accept expressions, and using the "trim" filter as a workaround is also impossible, because it doen's accept expressions.

```
set /a "total_duration = %IN% + %IN% + %IN%"
```

Same thing as before, but scrolling from right to left:

```
ffmpeg -framerate 1/%T% -start_number 0 -i %IN% -framerate 1/%T% -start_number 1 -i %IN% -framerate 1/%T% -start_number 2 -i %IN% -filter_complex [0][1][2]hstack=inputs=3,fps=%FPS%,crop=w=2*iw/3:x='iw/3*mod(t,%T%)/%T%' -t %D% -y %OUT%
```

Same thing as before, but scrolling from top to bottom:

```
ffmpeg -framerate 1/%T% -start_number 2 -i %IN% -framerate 1/%T% -start_number 1 -i %IN% -framerate 1/%T% -start_number 0 -i %IN% -filter_complex [0][1][2]vstack=inputs=3,fps=%FPS%,crop=h=2*ih/3:y='ih/3*(1-mod(t,%T%)/%T%)' -t %D% -y %OUT%
```

Same thing as before, but scrolling from bottom to top:

```
ffmpeg -framerate 1/%T% -start_number 0 -i %IN% -framerate 1/%T% -start_number 1 -i %IN% -framerate 1/%T% -start_number 2 -i %IN% -filter_complex [0][1][2]vstack=inputs=3,fps=%FPS%,crop=h=2*ih/3:y='ih/3*mod(t,%T%)/%T%' -t %D% -y %OUT%
```
2.6 Extract many images from a video

```
rem Extract many images from a video
ffmpeg -i in.mp4 -vf fps=0.2 -y image%%4d.jpg
pause :: Wait for a keypress
```

This batch file reads the file in.mp4 and produces images with the filenames image0000.jpg, image0001.jpg, image0002.jpg, and so on.

-`-vf fps=0.2` this specifies that images are extracted with a framerate of 0.2, which means one frame every 5 seconds.

  Omit this option if you want to extract all images.

-`-y` this option means that FFmpeg will overwrite any output files that already exist with the same filename, without asking. If you omit this option, FFmpeg would ask before overwriting a file.

This example will extract each n_th frame from a video, beginning with the 0_th frame:

```
set "IN=video.mp4" :: Input video
set "STEP=10" :: Step width
set "OUT=image%%4d.jpg" :: Output images filename
ffmpeg -i %IN% -vf framestep=%STEP% -y %OUT%
pause
```

2.7 Extract the last frame from a video

```
ffmpeg -sseof -0.2 -i in.mp4 -q:v 1 -update 1 last_frame.jpg
pause
```
2.8  Modify brightness, contrast, saturation, gamma and hue

rem Modify brightness, contrast, saturation, gamma and hue

set "INPUT=PanoView.mp4"  :: Input video
set "OUTPUT=out.mp4"        :: Output video
set "CONTRAST=1.0"          :: Contrast in range -1000 to 1000, normal is 1.0
set "BRIGHT=0.0"           :: Brightness in range -1.0 bis 1.0, normal is 0.0
set "SATUR=1.2"            :: Saturation in range 0.0 bis 3.0, normal is 1.0
set "GAMMA=1.0"            :: Gamma in range 0.1 to 10.0, normal is 1.0
set "HUE=20"               :: Color correction (hue), negative shifts towards red and positive towards blue, normal is 0
set "QU=2"                 :: Typical values are in the -30...+30 range
set "CONTRAST=1.0"         :: Contrast in range -1000 to 1000, normal is 1.0
set "BRIGHT=0.0"           :: Brightness in range -1.0 bis 1.0, normal is 0.0
set "SATUR=1.2"            :: Saturation in range 0.0 bis 3.0, normal is 1.0
set "GAMMA=1.0"            :: Gamma in range 0.1 to 10.0, normal is 1.0
set "HUE=20"               :: Color correction (hue), negative shifts towards red and positive towards blue, normal is 0
set "QU=2"                 :: MP4 Quality, 0 is best Quality, 2 is normal, 9 is strongest compression

ffmpeg -i %INPUT% -vf hue=h=%HUE%,eq=contrast=%CONTRAST%:brightness=%BRIGHT%:saturation=%SATUR%:gamma=%GAMMA% ^
-q:v %QU% -codec:v mpeg4 %OUTPUT%

-pause

-vf is the command for "Video Filter". There are many different filters, see chapter "Video Filter" in the FFmpeg documentation.

In this case we use two filters, which are separated by a (,) comma.

- The first filter is "hue" and makes a rotation of the color circle.
- The second filter is "eq" and adjusts contrast, brightness, saturation and gamma.

From a mathematically point of view these functions work as follows:

- Contrast is a multiplication by a constant. Please note that what contrast does is scale the distance of a pixel's value from the median value i.e. 128 for a 8-bit input. So, if a pixel channel has a value of 100, then a contrast of 3 results in a value of 128 + 3*(100-128) = 44.
- Brightness is the addition of a constant.
- Saturation is difficult to describe mathematically. Setting saturation to 0 would produce a black and white video.
- Gamma is a nonlinear distortion of the transfer function. When you increase the gamma value, details in dark areas become better visible.

It doesn't care in which order you list the parameters in the command line. They are always executed in the order contrast -> brightness -> gamma.
## 2.9 Strong contrast enhancement

There are several filters that can be used for a strong contrast enhancement:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Example for strong contrast enhancement by a factor 5: Input range [0.1 ... 0.3], Output range [0.0 ... 1.0], Output = 5 * (Input - 0.1)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq</td>
<td>eq=brightness=0.3,eq=contrast=5</td>
<td>The pivot point for contrast is always at 0.5 which means you have to adjust both brightness and contrast. The eq filter must be invoked two times, because we need first the brightness adjustment and then the contrast adjustment.</td>
</tr>
<tr>
<td></td>
<td>eq=brightness=0.3:contrast=5</td>
<td>This doesn't work as expected because the eq filter is invoked only one time, which means the order is contrast before brightness and that's the wrong order in this case.</td>
</tr>
<tr>
<td></td>
<td>eq=brightness=1.5:contrast=5</td>
<td>This doesn't work because the brightness value isn't in the allowed range [-1 ... +1]</td>
</tr>
<tr>
<td>geq</td>
<td>For 8-bit video: geq=lum='5*(lum(X,Y)-25.6)':cr='cr(X,Y)':cb='cb(X,Y)'; For 8-bit video with limiter: geq=lum='clip(5*(lum(X,Y)-25.6),0,255)':cr='cr(X,Y)':cb='cb(X,Y)'</td>
<td>Not recommended because it's slow, has no built-in limiter and the function must be different for 8-bit and 10-bit videos.</td>
</tr>
<tr>
<td></td>
<td>For 10-bit video: geq=lum='5*(lum(X,Y)-102.4)':cr='cr(X,Y)':cb='cb(X,Y)'</td>
<td></td>
</tr>
<tr>
<td>colorlevels</td>
<td>colorlevels=rimin=0.1:gimin=0.1:bimin=0.1:rimax=0.3:gimax=0.3:bimax=0.3</td>
<td>Best method because you can directly set the black and white points. The only drawback is that you have to write the same values three times, but that can be done with variables in the batch file.</td>
</tr>
<tr>
<td>curves</td>
<td>curves=all='0/0 0.1/0 0.3/1 1/1'</td>
<td>This is a nonlinear transfer function because it uses a smooth curve through the specified points.</td>
</tr>
</tbody>
</table>
### 2.10 Inverting a video or image (make a negative)

There are several methods for inverting (which means black becomes white, and vice versa):

<table>
<thead>
<tr>
<th>Filter</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq</td>
<td>eq=contrast=-1</td>
<td>This changes only bright to dark and vice versa, but keeps the colors as they are.</td>
</tr>
<tr>
<td>negate</td>
<td>negate</td>
<td>This negates all channels and changes the colors to their complementary colors.</td>
</tr>
<tr>
<td>geq</td>
<td>geq=x='255-r(X,Y)' : g='255-g(X,Y)' : b='255-b(X,Y)'</td>
<td>Same result as negate. Can also be used if only one or two channels are to be inverted. The functions must be different for 10-bit videos.</td>
</tr>
</tbody>
</table>
2.11 Correcting the color channels

Color corrections can be made with the "colorchannelmixer" filter. In this example we amplify the red channel and attenuate the blue channel. The values must be in the range [-2 ... +2].

| set "INPUT=7Z7A2065.MOV" :: Input video                  |
| set "OUTPUT=out.mp4" :: Output video                     |
| set "R=1.2" :: Factor for red channel                    |
| set "G=1.0" :: Factor for green channel                  |
| set "B=0.8" :: Factor for blue channel                   |
| set "QU=2" :: MP4 Quality, 0 ist best Quality, 2 is normal, 9 is strongest compression |

```
ffmpeg -i %INPUT% -vf colorchannelmixer=rr=%R%:gg=%G%:bb=%B% -q:v %QU% -codec:v mpeg4 -y %OUTPUT%
pause
```

Exchange red and green components:

```
ffmpeg -i %INPUT% -vf colorchannelmixer=0:1:0:0:1:0:0:0:0:0:1:0 -y %OUTPUT%
pause
```
2.12 Colorhold

This video filter removes all color informations except for one certain color. It has three parameters:

"color" is the color to be preserved, can be specified by name or by RGB values, for example "orange" can be replaced by FFA500 (or 0xFFA500 or #FFA500)

"similarity" is a fraction, 0.01 means only the specified color is preserved, 1.0 means all colors are preserved.

Note: The normalization of the "similarity" value was changed in May 2020. Old values must now be divided by sqrt(3) to get the same result as before.

"blend" is a fraction, 0.0 makes pixels fully gray, higher values result in more preserved color.

This example preserves only colors from yellow to orange to light brown:

```
ffmpeg -i 7Z7A2027.jpg -filter_complex split[1][2];[1]colorhold=color="orange":similarity=0.29:blend=0[3];[2][3]hstack
-y out.jpg
```

Output of this example:
2.13 Atmospheric dispersion correction

It's possible to shift the RGB channels with respect to each other:

```bash
set "IN=P1000479.mov"       :: Input video
set "OUT=out.mp4"            :: Output video
set "RV=5"                   :: Vertical shift of red channel
set "BV=-5"                  :: Vertical shift of blue channel

ffmpeg -i %IN% -lavfi "rgbashift=rv=%RV%:bv=%BV%" -y %OUT%
```

2.14 Amplify filter

The "amplify" filter amplifies differences between adjacent frames. Good for motion detection, but it's also sensitive to noise.
When you have a fisheye camera pointing upwards, it's unavoidable that you are visible in the video at the beginning and the end, because you must start and stop the camera. That means we must cut off the beginning and the end.

```
rem  Extract a time segment from a video

set "INPUT=PanoView.mp4"  :: Input video
set "OUTPUT=out.mp4"      :: Output video
set "START=2.0"           :: Start time in seconds
set "LENGTH=3.0"          :: Length of the segment in seconds

ffmpeg -ss %START% -t %LENGTH% -i %INPUT% -c copy %OUTPUT%
```

The arguments for -ss and -t can also be specified in hours, minutes and seconds:

1:20 = 1 minute, 20 seconds
1:10:30 = 1 hour, 10 minutes, 30 seconds

Instead of the length it's also possible to specify the end time with the -to option.

If you want to save the output video with exactly the same quality as the input video (without re-encoding), then use the -c copy option. In this case it makes no sense to specify the output video quality.

```
ffmpeg -ss 5 -i input.mov -t 10 -c copy output.mov
```

The same thing can also be done with the "trim" filter.
2.16 Trim filter

Drop everything except the second minute of input:

```bash
ffmpeg -i in.mp4 -vf trim=60:120 out.mp4
```

Keep only the first second:

```bash
ffmpeg -i in.mp4 -vf trim=duration=1 out.mp4
```
2.17 Tpad filter, add a few seconds black at the beginning or end

Method 1, using the "tpad" filter:

```
set "IN=my_video.mp4" :: Input video
set "DUR=3" :: Duration in seconds
set "OUT=out.mp4" :: Output video

ffmpeg -i %IN% -vf tpad=start_duration=%DUR% %OUT%
```

```
pause
```

The "tpad" filter inserts frames at the beginning or at the end of a video. These frames contain either a uniform color or a copy of the first or last frame. The default color is black.

Method 2, using the concat filter:

```
set "IN=my_video.mp4" :: Input video
set "DUR=3" :: Duration in seconds
set "OUT=out.mp4" :: Output video

ffmpeg -i %IN% -an -filter_complex 'color=black:duration=%DUR%[black];[black][0:0]concat=n=2:v=1:a=0[v]' -map [v] %OUT%
```

```
pause
```
2.18 Extract the last 30 seconds of a video

When I make real-time videos of meteors, I let the Panasonic LUMIX GH5S camera record continuously. When I see a meteor, I speak to the soundtrack in which part of the sky I’ve seen it, and after about 10 seconds I press the REC button to stop the recording, and immediately start a new recording. That means after downloading the videos to the computer, meteors are always at the end of the videos. There is no need to watch the videos in full length (that would be boring). This batch file extracts the last 30 seconds of the video which is drag-and-dropped over it, and for the output filename the string "P1" is replaced by "CUT" (e.g. P1000336.MOV becomes CUT000336.MOV). It’s lossless because the "-c copy" option is used.

```
set INPUT=%1
set OUTPUT=%INPUT:P1=CUT%
ffmpeg -sseof -30 -i %INPUT% -c copy %OUTPUT%
pause
```

This batch file (for Windows 7) does the same thing for all P1*.MOV files in the current folder:

```
for %%f in (P1*.MOV) do call :for_body %%f
goto :the_end

:for_body
  set INPUT=%1
  set OUTPUT=%INPUT:P1=CUT%
  ffmpeg -sseof -30 -i %INPUT% -c copy -y %OUTPUT%
exit /b

:the_end
pause
```
2.19 Fade-in and fade-out

Fade-in and fade-out for a video of known length (only for video, not for audio). Here the times are expressed in frames:

```bash
ffmpeg -i input.mp4 -vf 'fade=in:0:30,fade=out:9650:30' output.mp4
```

Fade-in and fade-out of a video of known length (both video and audio). Here the times are in seconds:

```bash
ffmpeg -i input.mp4 -vf 'fade=in:st=0:f=1,fade=out:st=32:d=1' -af 'afade=in:st=0:d=1,afade=out:st=32:d=1' output.mp4
```

This is a workaround for fade in/out a video with unknown duration:

```bash
ffmpeg -i input.mp4 -sseof -1 -copyts -i input.mp4 -filter_complex 
"[1]fade=out:0:30[t];[0][t]overlay,fade=in:0:30[v]; anullsrc,atriim=0:2[at];[0][at]acrossfade=d=1,afade=d=1[a]"
-map "[v]" -map "[a]" -c:v libx264 -crf 22 -preset veryfast -shortest output.mp4
```

The trick is to feed the same input twice. From the second input only the last second is used. The timestamps are preserved. A fade-out is applied to the short second input, and then both files are combined with overlay. For audio a 2 seconds dummy with silence is created, and then crossfaded with the input audio. The -shortest option cuts the output to the same length as the input.

Another workaround for making fade-in and fade-out for audio of unknown length:

```bash
ffmpeg -i input.mp4 -filter_complex "afade=d=0.5, areverse, afade=d=0.5, areverse" output.mp4
```

The same thing does also work for video, but keep in mind that you need a lot of memory for the reverse filter:

```bash
ffmpeg -i input.mp4 -filter_complex "fade=d=0.5, reverse, fade=d=0.5, reverse" output.mp4
```

Another option is to use acrossfade with a silent track, but this works not for video because there is no crossfade filter for video:

```bash
ffmpeg -i input.mp4 -filter_complex "aevalsrc=0:d=0.6 [a_silence]; [0:a:0] [a_silence] acrossfade=d=0.6" output.mp4
```

Afade curves are shown on this wiki page: [https://trac.ffmpeg.org/wiki/AfadeCurves](https://trac.ffmpeg.org/wiki/AfadeCurves)
2.20 Crossfading

The different types of xfade crossfadings are shown on this wiki page:
https://trac.ffmpeg.org/wiki/Xfade
2.21  

Crop a video

Cropping means to cut off the borders, and in the next step you can also set the size (width * height) of the output video:

```plaintext
rem  Crop and set the output size
set "INPUT=PanoView.mp4"  :: Input video
set "OUTPUT=out.mp4"      :: Output video
set "CROP=1224:1224:0:0"  :: Specify the visible part: Width, height, left edge, top edge
set "SIZE=800x800"        :: Width and height of the output video (can be smaller or larger than the input video)
                           :: Keep the width/height ratio constant, otherwise the video looks distorted,
                           :: for example a circle would become an ellipse.
set "QU=2"                :: MP4 Quality, 0 is best Quality, 2 is normal, 9 is strongest compression
ffmpeg -i %INPUT% -vf crop=%CROP% -s %SIZE% -q:v %QU% -codec:v mpeg4 %OUTPUT%
```

In the crop filter you can use the variables "iw" and "ih", which are the width and height of the input video.

If the 3rd and 4th parameter (coordinates of top left corner) isn't specified, the crop will be automatically centered.

- `crop=ih:ih`  makes a centered square crop, useful for fulldome videos
- `crop=iw/2:ih:0`  returns the left half of the input video
- `crop=iw/2:ih:iw/2`  returns the right half of the input video
- `crop=iw/4:ih/4`  strong enlargement by a factor 4 in the center of the video

The "pad" filter does the opposite thing, it adds paddings with a uniform color to the video.
2.22 Zoompan

This is a very powerful filter. It can also be used for making slideshows. The "d" option specifies how long each image is shown.

Parameters:
'
out_time' or 'ot'  Timestamp in seconds of each output frame produced by zoompan.
'
in_time' or 'it'   Timestamp in seconds of each input frame to the zoompan filter.

In most cases it's useful to specify the size of the output frames with the "s" option, because the default is 1280x720.

(I'm still working on this chapter...)
2.23 Changing the speed: slow motion and timelapse

```
rem Changing the speed (slow motion or timelapse)

set "INPUT=PanoView.mp4"    :: Input video
set "OUTPUT=out.mp4"        :: Output video
set "RATE=30"               :: Output framerate
set "SPEED=3.0"             :: Speed factor, smaller than 1 = timelapse, 1 = real time, larger than 1 = slow motion
set "QU=2"                  :: MP4 Quality, 0 is best Quality, 2 is normal, 9 is strongest compression

ffmpeg -i %INPUT% -vf setpts=%SPEED%*PTS -r %RATE% -q:v %QU% -codec:v mpeg4 -an -y %OUTPUT%
pause
```

In this example the settings for "RATE" and "SPEED" are totally independent from each other. FFmpeg will automatically skip or duplicate frames, if required.

Example: If both input and output frame rate are 30, and if SPEED = 3, then each frame will automatically duplicated 2 times, so that we see it 3 times in the output video. If SPEED = 0.5, then each second frame is skipped.

In this example the slow motion or timelapse effect affects only video and not audio. It makes sense to disable the audio channel with the -an option.

The "setpts" filter is described in the "Multimedia Filters" section in the FFmpeg documentation.

There's a wiki page covering this subject: https://trac.ffmpeg.org/wiki/How%20to%20speed%20up%20/%20slow%20down%20a%20video

The timebase (TB in setpts filter) is expressed in seconds [s].
The framerate (FR in setpts filter) is expressed in 1/seconds [s^-1]
In many cases the timebase is the reciprocal of the framerate, but this isn't always the case.
Some more examples:
Double speed:  setpts=0.5*PTS  Half speed: setpts=2.0*PTS
Delay by x frames (assuming the framerate is constant): setpts=PTS+x/(FR*TB) or tpad=x
Delay by x seconds:  setpts=PTS+x/TB or tpad=x/framerate
Start counting PTS from zero:  setpts=PTS-STARTPTS

36
2.24  Slow motion or timelapse only for a segment of the video

See the comments for explanation.

```bash
set "IN=7Z7A2089.mov" :: Input Video
set "T1=5" :: Start time T1
set "T2=8.5" :: Time T2 when slow motion begins
set "T3=9.7" :: Time T3 when slow motion ends
set "T4=11" :: End time T4
set "SPEED=5" :: Speed factor, smaller than 1 = timelapse, larger than 1 = slow motion
set "FR=30" :: Output framerate
set "OUT=out.mp4" :: Output video

ffmpeg -i %IN% -filter_complex "[0:v]trim=%T1%:%T2%,,setpts=PTS-STARTPTS[v1];[0:v]trim=%T2%:%T3%,,setpts=%SPEED%*(PTS-STARTPTS)[v2];[0:v]trim=%T3%:%T4%,,setpts=PTS-STARTPTS[v3];[v1][v2][v3]concat=n=3:v=1" -an -r %FR% -q:v 2 -y out.mp4
```
2.25 Time Remapping

This is an example for a gradual ramp into and out of slow motion:

```
ffmpeg -f lavfi -i testsrc2=size=vga:duration=10:rate=20 -lavfi "^ 
[0]trim=0.0:3.2,setpts=(PTS-STARTPTS)[1];^ 
[0]trim=3.2:3.6,setpts=(PTS-STARTPTS)/0.80[2];^ 
[0]trim=3.6:4.0,setpts=(PTS-STARTPTS)/0.60[3];^ 
[0]trim=4.0:6.0,setpts=(PTS-STARTPTS)/0.40[4];^ 
[0]trim=6.0:6.4,setpts=(PTS-STARTPTS)/0.60[5];^ 
[0]trim=6.4:6.8,setpts=(PTS-STARTPTS)/0.80[6];^ 
[0]trim=6.8:10.0,setpts=(PTS-STARTPTS)[7];^ 
[1][2][3][4][5][6][7]concat=n=7:v=1" -y out.mp4 
```

This is an example for a 10s input video where the framerate changes linearly from 20 to 10:

```
ffmpeg -f lavfi -i testsrc2=size=vga:duration=10:rate=20 -lavfi " 
[0]trim=0:1,setpts=(PTS-STARTPTS)/0.975[1]; 
[0]trim=1:2,setpts=(PTS-STARTPTS)/0.925[2]; 
[0]trim=2:3,setpts=(PTS-STARTPTS)/0.875[3]; 
[0]trim=3:4,setpts=(PTS-STARTPTS)/0.825[4]; 
[0]trim=4:5,setpts=(PTS-STARTPTS)/0.775[5]; 
[0]trim=5:6,setpts=(PTS-STARTPTS)/0.725[6]; 
[0]trim=6:7,setpts=(PTS-STARTPTS)/0.675[7]; 
[0]trim=7:8,setpts=(PTS-STARTPTS)/0.625[8]; 
[0]trim=8:9,setpts=(PTS-STARTPTS)/0.575[9]; 
[0]trim=9:10,setpts=(PTS-STARTPTS)/0.525[10];[1][2][3][4][5][6][7][8][9][10]concat=n=10:v=1" -y out.mp4 
```

The length of the output video is 13.65s

Use the following example carefully, as I'm not 100% convinced that the approach is correct. This is based on an posting from Nicolas George in the FFmpeg user mailing list, September 23, 2019. In the first equation it's unclear if t is the time in the input video or in the output video.
So, to compute the timestamp of a frame with variable speed:

* Express your frame rate as a complete formula: \( t \rightarrow v \)

* Integrate it: \( t \rightarrow f \).

* Find the reciprocal: \( f \rightarrow t \).

Let's assume we have a 10s video and the framerate changes linearly from 20 at the beginning to 10 at the end:

\[
v = 20 - t \quad v(0) = 20 \quad v(10) = 10
\]

After integrating we get:

\[
f = 20 \times t - 0.5 \times t^2
\]

The inverse function is:

\[
t = 20 - \sqrt{400 - 2 \times f}
\]

Create a test video with framerate=20 and length=10s:

```
ffmpeg -f lavfi -i testsrc2=size=vga:duration=10:rate=20 -y test.mp4
```

Apply the time remapping:

```
ffmpeg -i test.mp4 -lavfi setpts='(20-sqrt(400-2*N))/TB' -y out.mp4
```

The resulting video gets slower towards the end (too slow, in fact), and the length is 18.95s and that seems to be wrong. With a constant framerate of 20 the length is 10s, with a constant framerate of 10 the length is 20s, and if the framerate changes from 20 to 10 the length should be about 15s. I don't fully understand what's going on here.

Keywords for searching: "Time remapping", "Time ramp", "Slow motion ramp", "Speed ramp"
2.26       Insert a text which is visible for the whole duration

set "IN=input.mov"     :: Input video
set "OUT=output.mp4"   :: Output video
set "FONT=arial.ttf"   :: Font
set "TEXT=Hello_World" :: Text (no space characters allowed, see next example)
set "COLOR=yellow"     :: Text color
set "SIZE=20"          :: Font size
set "POS_X=(w-tw)/2"   :: X position of text, use (w-tw)/2 for centering
set "POS_Y=(h-th)/2"   :: Y position of text, use (h-th)/2 for centering

ffmpeg -i %IN% -vf drawtext='fontfile=%FONT%:text=%TEXT%:fontcolor=%COLOR%:fontsize=%SIZE%:x=%POS_X%:y=%POS_Y%' -c:v mpeg4 -q:v 1 -y %OUT%

pause

2.27       Slowly fade a text in and out

rem Slowly fade a text in and out

set "INPUT=PanoView.mp4"   :: Input video
set "OUTPUT=out.mp4"       :: Output video
set "QU=2"                 :: MP4 Quality, 0 is best Quality, 2 is normal, 9 is strongest compression
set "NAME=TEXT1"           :: Unique name for this text
set "FONT=arial.ttf"       :: Font
set "TEXT=MeinText.txt"    :: Text filename (must be UTF-8 coded, if the text contains non-ASCII characters like ä, ö, ü. The text can be ASCII coded if no special characters are used.
set "COLOR=yellow"         :: Text color
set "SIZE=250"             :: Font size
The text must be saved as a *.txt file. If the text contains non-ASCII special characters like ä, ö, ü then the encoding must be UTF-8. If the text contains only ASCII characters, then "ANSI" encoding is possible as well.

In some cases `drawtext` shows a non-printable character (for example an empty rectangle) at the beginning of the text. This is a BOM (Byte Order Mark) that was automatically added to the text by some Windows programs at the beginning of the file. Older versions of Notepad (on Windows 7) show this behaviour and you can't disable it. The BOM consists of three bytes \texttt{EF_{hex}BB_{hex}BF_{hex}}.

See also here: [https://en.wikipedia.org/wiki/UTF-8#Byte_order_mark](https://en.wikipedia.org/wiki/UTF-8#Byte_order_mark)

There are several solutions for this problem:
• Open the text file with a hex editor and remove the first three characters (EF$_{hex}$ BB$_{hex}$ BF$_{hex}$). For example you can use Hex Editor MX: http://hexedit.nextsoft.de/

• If you can't find a newer 32-bit Notepad version for Windows 7, you can use Notepad++ instead. Select "UTF-8" in the "Encoding" menu. https://notepad-plus-plus.org/

• Newer versions of Notepad (on Windows 10) have a selection between "UTF-8" and "UTF-8 with BOM". Using "UTF-8" will solve the problem.

Problem: You want to show the content of a credits file scrolling up. The file contains many lines of different lengths.

```
 drawtext=textfile=credits.txt:x=(w-text_w)/2:y=h-100*t
```

The variable text_w is the width of the longest line in the text file. This line is center-aligned in the frame, and all other (shorter) lines are left-aligned to the same X position as the longest line. But that's not what you want. Is it somehow possible that each line is center-aligned?

Solution: See the workaround with *.ass subtitles in this document.
2.28 Show a running clock in the video

In this example a running clock is inserted in each frame of the video, in the format "hours:minutes:seconds.milliseconds"

```
set "IN=P1000479.mov" :: Input video
set "OUT=sylvia.mp4" :: Output video

set "BP_R=0.015" :: Black point red, positive value makes background darker
set "BP_G=0.005" :: Black point green, positive value makes background darker
set "BP_B=0.015" :: Black point blue, positive value makes background darker

set "WP=0.26" :: White point

set "S=300" :: Start time
set "T=40" :: Duration

set "FONT=arial.ttf" :: Font
set "COLOR=white" :: Font color
set "BOXCOLOR=black" :: Background color
set "SIZE=30" :: Font size
set "POSITION_X=0" :: X position of clock
set "POSITION_Y=(h-th)" :: Y position of clock
set "OF=2340" :: Offset time in seconds, shown in the first frame
set "I=0.04" :: Time intervall from one frame to the next = 1/framerate

set CLOCK=
drawtext='fontfile=%FONT%:text=%%{eif:\mod((%OF++I*n)/3600,24)\:"d'\:2})\:"%%{eif:\mod((%OF++I*n)/60,60)\:"d'\:2})\:"%%{eif:\mod(\%OF++I*n*1000,1000)\:"d'\:3}:fontcolor=%COLOR%:boxcolor=%BOXCOLOR%:box=1:fontsize=%SIZE%:x=%POSITION_X%:y=%POSITION_Y%

ffmpeg -ss %S% -i %IN% -vf "colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%,%CLOCK%" -pix_fmt yuv420p -t %T% -y %OUT%
```
This batch file does the same thing and is simpler:

```plaintext
set "IN=P1000479.mov"     :: Input video
set "OUT=sylvia.mp4"       :: Output video
set "BP_R=0.015"           :: Black point red, positive value makes background darker
set "BP_G=0.005"           :: Black point green, positive value makes background darker
set "BP_B=0.015"           :: Black point blue, positive value makes background darker
set "WP=0.26"              :: White point
set "S=300"                :: Start time
set "T=40"                 :: Duration
set "FONT=arial.ttf"       :: Font
set "COLOR=white"          :: Font color
set "BCOLOR=black"         :: Background color
set "SIZE=30"              :: Font size
set "POS_X=0"              :: X position of clock
set "POS_Y=(h-th)"         :: Y position of clock
set "OFFSET=2340"          :: Offset time in seconds, added to the timestamp of the first frame

set CLOCK=drawtext='fontfile=%FONT%:text=%%{pts:hms:%OFFSET%}:fontcolor=%COLOR%:boxcolor=%BCOLOR%:box=1:fontsize=%SIZE%
x=%POS_X%:y=%POS_Y%'

ffmpeg -ss %S% -i %IN% -vf "colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%,%CLOCK%" -pix_fmt yuv420p -t %T% -y %OUT%
pause
```

This is another example, using the "timecode" option of the drawtext filter:

```plaintext
ffmpeg -f lavfi -i testsrc2=size=hd720:duration=10 -vf
drawtext=fontsize=60:fontcolor=Black:fontfile='arial.ttf':timecode='00:00:00:00':r=25:x=20:y=40 -y out.mp4
```

pause
2.29 Generation of curved text for fulldome projection

rem Create a video with curved text fade-in fade-out, silent audio

set "SIZE=1200" :: Video size (square)
set "QU=2" :: MP4 quality level, 0 is best quality, 2 is normal, 9 is strong compression
set "FPS=30" :: Output Frame rate
set "FONT=arial.ttf" :: font
set "FSIZE=60" :: font size
set "COLOR=white" :: text color
set "BACK=black" :: background color
set "DUR=10" :: duration of video
set "TEXT=text13.txt" :: text file
set "POS_X=(w-tw)/2" :: X text position, for centered text: (w-tw)/2
set "POS_Y=h*0.9" :: Y text position
set "S=1" :: start time for text
set "E=9" :: end time for text
set "FI=2" :: fade-in duration (may be small, but not zero)
set "FO=2" :: fade-out duration (may be small, but not zero)
set "OUTPUT=text13.mp4" :: Output filename

ffmpeg -f lavfi -i color=c=%BACK% -i xmap_3648.pgm -i ymap_3648.pgm -f lavfi -i anullsrc -r %FPS% -t %DUR% -aspect "1:1" -lavfi scale=3648:3648,drawtext='fontfile=%FONT%:textfile=%TEXT%:fontcolor_expr=%COLOR%@%%{e\:clip((t-%S%/\%FI)+(%E%-t)/\%FO%\:between(t,%S%+%FI%,%E%),0,1)}:fontsize=%FSIZE%:x=%POS_X%:y=%POS_Y\%',format=pix_fmts=rgb24,remap -s %SIZE%x%SIZE% -c:v mpeg4 -c:a aac -shortest -q:v %QU% -y %OUTPUT%

I have to admit that this is a complicated command line. The actual core is the "remap" filter, with which you can create arbitrary distortions. The distortion is described in the two files xmap_3648.pgm and ymap_3648.pgm. In these files the pixel in the input video from which it is retrieved is indicated for each pixel. You have to write a (C#) program that can create these files.

-i color=c=black creates a black image
-i anullsrc creates an empty audio track
This is the C# code for generating the xmap and ymap files:

```csharp
int a = (int)numericUpDown1.Value; // get the size of the square map
double c = (double)numericUpDown2.Value; // this is the aspect ratio of the text, normal = 1
int b = a/2;
int xx, yy;

TextWriter xmap = File.CreateText("xmap_" + a.ToString() + ".pgm");
xmap.Write("P2\n");
xmap.Write("# Xmap file for fulldome remap \n");
xmap.Write(a.ToString() + " " + a.ToString() + " \n");
xmap.Write("65535\n");

TextWriter ymap = File.CreateText("ymap_" + a.ToString() + ".pgm");
ymap.Write("P2\n");
ymap.Write("# Ymap file for fulldome remap \n");
ymap.Write(a.ToString() + " " + a.ToString() + " \n");
ymap.Write("65535\n");

for (int y = 0; y < a; y++)
{
    for (int x = 0; x < a; x++)
    {
        xx = x;
        yy = y;
        if (y > b)
        {
            xx = b + (int)(b / c * Math.Atan((double)(x - b) / (double)(y - b)));
            yy = b + (int)Math.Sqrt((x - b) * (x - b) + (y - b) * (y - b));
            if (xx < 0) xx = 0;
            if (yy < 0) yy = 0;
            if (xx > a - 1) xx = a - 1;
            if (yy > a - 1) yy = a - 1;
        }
        xmap.Write(xx + " ");
        ymap.Write(yy + " ");
    }
    xmap.Write("\n");
    ymap.Write("\n");
}

xmap.Write("\n");
ymap.Write("\n");
xmap.Close();
ymap.Close();
```
This is a simpler example for generating curved text for full-dome projection, using the v360 filter:

```plaintext
set "UP=30"                   :: Up-looking angle in degrees (center of the rectangular video)
set "H=64"                    :: Horizontal field of view, this is for 16:9 aspect ratio
set "V=36"                    :: Vertical field of view, this is for 16:9 aspect ratio
set "SIZE=1200"               :: Square size of the output video
set "FONT=arial.ttf"          :: font
set "FSIZE=120"               :: font size
set "COLOR=white"             :: text color
set "BACK=black"              :: background color
set "TEXT=text13.txt"         :: text file
set "POS_X=(w-tw)/2"          :: X text position, for centered text: (w-tw)/2
set "POS_Y=(h-th)/2"          :: Y text position, for centered text: (h-th)/2
set "S=1"                     :: start time for text
set "E=9"                     :: end time for text
set "FI=2"                    :: fade-in duration (may be small, but not zero)
set "FO=2"                    :: fade-out duration (may be small, but not zero)
set "DUR=10"                  :: duration of video
set "OUT=out.mp4"             :: Output video

ffmpeg -f lavfi -i color=%BACK%:size=hd1080 -vf drawtext='fontfile=%FONT%:textfile=%TEXT%:fontcolor_expr=%COLOR%@\% {e\:clip\((t-%S%)/%FI%*between(t,%S%,%S++%FI%)+(\%E%-t)/%FO%*between(t,%S++%FI%,%E %),0,1)}:fontsize=%FSIZE%:x=%POS_X%:y=%POS_Y%',v360=input=flat:ih_fov=%H%:iv_fov=%V ::output=fisheye:h_fov=180:v_fov=180:pitch='90-%UP%':w=%SIZE%:h=%SIZE% -t %DUR% -y %OUT% pause
```
2.30 Write text on a transparent layer

In this example text is written on a transparent background (black@0). This video is scaled to the same size as the input video with the "scale2ref" filter. Finally the text video is overlaid over the main video.

The advantage of this method is that you can modify the geometry of the text before overlaying it. For example you can use the "displace", "perspective", "remap", "rotate" or "v360" filters for modifying the geometry.

set "IN=R0010008_er.mp4" :: Input video
set "OUT=out.mp4" :: Output video with overlaid test

ffmpeg -i %IN% -f lavfi -i color=black@0,format=rgba -lavfi [1][0]scale2ref[a][b], [a]drawtext="fontsize=80:text='TEST':box=1:boxcolor=red:boxborderw=10:fontcolor=yellow:x=(w-text_w)/2:y=(h-text_h)/2"[c];[b][c]overlay -t 5 -y %OUT%

Note: It is required to add "format=rgba" after the "color" video source. Otherwise the format negotiation could fail and agree on a yuv420 format (which doesn't have a transparency layer).

How the "scale2ref" filter works:

This filter has two inputs and two outputs. The first input is the video that shall be scaled, and the second input is the reference video from which the size is used. The first output is the scaled video, and the second output is a copy of the second input. The filter has many options but none of them are required for the basic function, as in this example.
2.31 Combine multiple videos with concat demuxer

The concat demuxer combines several videos without re-encoding. It's very fast.

```bash
rem   Final cut with concat demuxer
ffmpeg -f concat -i concat_list.txt -c copy -y MyVideo.mp4
```

You simply write all existing scenes into a text file (here: concat_list.txt), which looks like this:

<table>
<thead>
<tr>
<th>file</th>
<th>:</th>
<th></th>
<th>Title: A year in the woods</th>
</tr>
</thead>
<tbody>
<tr>
<td>text1.mp4</td>
<td>:</td>
<td>10</td>
<td>When and where</td>
</tr>
<tr>
<td>text2.mp4</td>
<td>:</td>
<td>10</td>
<td>Live video in the wood</td>
</tr>
<tr>
<td>Videos/scene20.mp4</td>
<td>:</td>
<td>12</td>
<td>Live video, camera</td>
</tr>
<tr>
<td>text22.mp4</td>
<td>:</td>
<td>10</td>
<td>In 15 months...</td>
</tr>
<tr>
<td>Videos/scene22.mp4</td>
<td>:</td>
<td>52</td>
<td>the end</td>
</tr>
</tbody>
</table>

To the right of the double colons are optional comments (e.g. the length of the scenes and a short description). Comments can also begin with #.

This method, however, requires that all scenes have
- the same size (width and height)
- the same video codec
- the same framerate
- the same audio codec
- the same number of audio tracks (take care when you use a camera which writes only a mono soundtrack)
- the same audio sample rate

If one of these conditions isn’t met, an error message is issued. You can then look at the properties of the files with FFprobe or Exiftool to find out where the files differ.

See also here: https://trac.ffmpeg.org/wiki/Concatenate
2.32  Combine multiple videos with concat filter

In this example the concat filter is used for input videos of the same size and no audio.

Each of the -ss and -t specifies the start time and length of the next input file. You can remove these options if you want to use the full videos.

The value n=3 passed to the concat filter should match the number of input files.

This filter does re-encode the videos, so the process is slow but you can also specify the encoding quality.

```
set "I1=my_video1.mp4" :: Input video 1
set "S1=0"                :: Set start time 1
set "L1=4"                :: Set length 1
set "I2=my_video2.mp4"    :: Input video 2
set "S2=3"                :: Set start time 2
set "L2=3"                :: Set length 2
set "I3=my_video3.mp4"    :: Input video 3
set "S3=6"                :: Set start time 3
set "L3=2"                :: Set length 3
set "OUT=out.mp4"         :: Output video

ffmpeg -ss %S1% -t %L1% -i %I1% -ss %S2% -t %L2% -i %I2% -ss %S3% -t %L3% -i %I3% -lavfi "concat=n=3:v=1:a=0" -an %OUT%
```

See also here: [https://trac.ffmpeg.org/wiki/Concatenate](https://trac.ffmpeg.org/wiki/Concatenate)
2.33 Switch between two cameras, using audio from camera1

rem Create a 6 seconds red video with 400Hz tone
ffmpeg -f lavfi -i color=c=red:s=vga -f lavfi -i sine=frequency=400 -t 6 -y video1.mp4

rem Create a 6 seconds test video with 1200Hz tone
ffmpeg -f lavfi -i testsrc2=s=vga -f lavfi -i sine=frequency=1200 -t 6 -y video2.mp4

rem Switch to video2 from 2 to 4 seconds, but use always the audio from video1
ffmpeg -i video1.mp4 -i video2.mp4 -filter_complex blend=all_expr='if(between(T,2,4),B,A)' -y test.mp4

Note: In this example both videos start at the same time. The video2 segment from 2 to 4 seconds is inserted in the output video from 2 to 4 seconds. You get this output video:

<table>
<thead>
<tr>
<th></th>
<th>0 &lt; t &lt; 2</th>
<th>2 &lt; t &lt; 4</th>
<th>4 &lt; t &lt; 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>from video1 (0...2)</td>
<td>from video2 (2...4)</td>
<td>from video1 (4...6)</td>
</tr>
<tr>
<td>Audio</td>
<td>from video1 (0...2)</td>
<td>from video1 (2...4)</td>
<td>from video1 (4...6)</td>
</tr>
</tbody>
</table>

If you want to insert the video2 segment from 0 to 2 seconds in the output video from 2 to 4 seconds, use this command line instead:

ffmpeg -i video1.mp4 -i video2.mp4 -filter_complex [1]tpad=start_duration=2[2];[0][2]blend=all_expr='if(between(T,2,4),B,A)' -y test.mp4

In this case you get this output video:

<table>
<thead>
<tr>
<th></th>
<th>0 &lt; t &lt; 2</th>
<th>2 &lt; t &lt; 4</th>
<th>4 &lt; t &lt; 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>from video1 (0...2)</td>
<td>from video2 (0...2)</td>
<td>from video1 (4...6)</td>
</tr>
<tr>
<td>Audio</td>
<td>from video1 (0...2)</td>
<td>from video1 (2...4)</td>
<td>from video1 (4...6)</td>
</tr>
</tbody>
</table>
2.34 Stack videos side by side (or on top of each other)

```bash
set "IN1=left.mp4"
set "IN2=right.mp4"
set "OUT=out.mp4"
rem use "hstack" for horizontal stacking and "vstack" for vertical stacking
ffmpeg -i %IN1% -i %IN2% -filter_complex hstack -an -shortest -c:v mpeg4 -y %OUT%
```

Note: If the videos have different width or height, use the "xstack" filter instead.

2.35 Horizontal and vertical flipping

This can be done with the "hflip" and "vflip" filters.
2.36 Stack four videos to a 2x2 mosaic

```bash
set "IN1=topleft.mp4"
set "IN2=topright.mp4"
set "IN3=bottomleft.mp4"
set "IN4=bottomright.mp4"
set "OUT=mosaic.mp4"

ffmpeg -i %IN1% -i %IN2% -i %IN3% -i %IN4% -filter_complex [0:v][1:v]hstack[t];[2:v][3:v]hstack[b];[t][b]vstack -an -shortest -c:v mpeg4 -q:v 1 -y %OUT%
```

Other method using xstack:

```bash
set "IN1=topleft.mp4"
set "IN2=topright.mp4"
set "IN3=bottomleft.mp4"
set "IN4=bottomright.mp4"
set "OUT=mosaic.mp4"

ffmpeg -i %IN1% -i %IN2% -i %IN3% -i %IN4% -filter_complex "xstack=inputs=4:layout=0_0|0_h0|w0_0|w0_h0" -shortest %OUT%
```

Display 4 inputs into a vertical 1x4 grid, note that the input videos may have different widths (vstack can't handle this case).

```bash
ffmpeg -i %IN1% -i %IN2% -i %IN3% -i %IN4% -filter_complex "xstack=inputs=4:layout=0_0|0_h0|0_h0|0_h0+h1|0_h0+h1+h2" %OUT%
```
2.37 Blink comparator

This is an example of a blink comparator. It creates an animated GIF that continuously toggles between two (or more) images.

```plaintext
rem Blink comparator, animated GIF
set "IN=pluto_%%1d.jpg" :: Filename of the images
set "FR=2.0" :: Frame rate
set "OUT=out.gif" :: Animated GIF output file

ffmpeg -framerate %FR% -i %IN% -q:v 0 -y %OUT%

pause
```

Please note that there is a known problem with FFmpeg's GIF encoder which may result in wrong colors in the output file. See the next chapter for a workaround.

If you want to create an MP4 instead, then you have to specify how long it should be and the input and output framerates:

```plaintext
rem Blink comparator, MP4
set "IN=pluto_%%1d.jpg" :: Filename of the images
set "FI=2.0" :: Framerate for reading in the pictures
set "T=10" :: Lenght in seconds
set "FO=25" :: Output framerate
set "OUT=out.mp4" :: Output MP4 file

ffmpeg -loop 1 -framerate %FI% -i %IN% -t %T% -r %FO% -q:v 0 -y %OUT%

pause
```

The parameter "-loop 1" causes the same images to be read in again and again. If you do this, you have to limit the length of the video somehow, in this case with "-t 10".
2.38 Animated GIF

There is a known problem with FFmpeg's GIF encoder which may result in wrong colors in the animated GIF output file. If you encounter this problem, you can use the following workaround which uses the palettegen and paletteuse filters. Thanks to Javier Infante Porro for posting this workaround in the FFmpeg user mailing list on September 26, 2019.

```bash
set "IN=in.gif" :: Input video (animated GIF)
set "COL=8" :: Number of colors (including one transparent color)
set "OUT=out.gif" :: Output video (animated GIF)

ffmpeg -i %IN% -lavfi "split[s0][s1];[s0]palettegen=max_colors=%COL%;[s1][p]paletteuse" -y %OUT%
pause
```

Please note that one entry in the palette is reserved for the transparent color by default. So when you set the max_colors parameter to 8, you have only 7 different visible colors. If you don’t want a transparent color, you must disable it with the reserve_transparent=0 option.

Much more about this subject can be found here:
http://blog.pkh.me/p/21-high-quality-gif-with-ffmpeg.html
2.39 Replace one frame in a video by another

This example shows how to replace a single image in a video with another image. You may have heard of a trick to insert a product image into a film for advertising purposes, only for the duration of a single frame. For example, if the frame rate is 25 frames per second, then a single frame will be shown for 40ms. That’s too short to recognize the product clearly, but it’s long enough to make viewers feel that they want this product. If, for example, a bratwurst or popcorn is shown for 40ms in the film, the sales figures for exactly these products increase after the end of the film. Although the viewer is not aware of why he has now gotten an appetite for a bratwurst or popcorn.

```
set "IN=scene8.mp4"          :: Input video
set "BW=bratwurst.jpg"       :: Image of bratwurst
set "W=1920"                 :: Width of input video
set "H=1080"                 :: Height of input video
set "T=3.0"                  :: Time when the image shall be insert
set "OUT=out.mp4"            :: Output video

ffmpeg -i %IN% -i %BW% -lavfi "[1]scale=w=%W%:h=%H%,setpts=%T%/TB[im];[0][im]overlay=eof_action=pass" -c:a copy -q:v 0 %OUT%
pause
```

The "scale" filter scales the image to the same size as the input video. If the image already has the correct size, you can omit this filter. The "setpts" filter sets the time for the image. The "overlay" filter then combines the two sources. The audio track is taken unchanged from the input video.

The same thing can also be done with the freezeframes filter:

```
set "IN=scene8.mp4"          :: Input video
set "IN2=test.mp4"           :: Second input which contains the replacement frame
set "F=75"                   :: Number of the frame to be replaced
set "R=1"                    :: Number of the replacement frame from the second input

ffmpeg -i %IN% -i %IN2% -lavfi freezeframes=first=%F%;last=%F%;replace=%R% out.mp4
```

pause
Unfortunately the FFmpeg documentation doesn't explain what all the modes do. So you have to look it up in the source code. The default mode is "normal".

<table>
<thead>
<tr>
<th>Mode</th>
<th>Function</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>$A \times \text{opacity} + B \times (1 - \text{opacity})$</td>
<td>Output is a mix of A and B, the default opacity is 1</td>
</tr>
<tr>
<td>addition</td>
<td>$A + B$</td>
<td>Output is $(A+B)$ with an upper limit at white level</td>
</tr>
<tr>
<td>average</td>
<td>$(A + B) / 2$</td>
<td>Output is the arithmetic mean of A and B</td>
</tr>
<tr>
<td>subtract</td>
<td>$A - B$</td>
<td>Output is $(A-B)$ with a lower limit at black level</td>
</tr>
<tr>
<td>multiply</td>
<td>$A \times B$</td>
<td></td>
</tr>
<tr>
<td>difference</td>
<td>$\text{abs}(A - B)$</td>
<td>Output is the absolute difference of A and B</td>
</tr>
<tr>
<td>grainextract</td>
<td>$50%_\text{gray_level} + A - B$</td>
<td>Output is $(A-B)$, shifted to 50% gray level, with limits at black and white levels</td>
</tr>
<tr>
<td>darken</td>
<td>$\text{min}(A,B)$</td>
<td>Output is the minimum of A and B</td>
</tr>
<tr>
<td>lighten</td>
<td>$\text{max}(A,B)$</td>
<td>Output is the maximum of A and B</td>
</tr>
<tr>
<td>and</td>
<td>$A &amp; B$</td>
<td></td>
</tr>
<tr>
<td>or</td>
<td>$A \mid B$</td>
<td></td>
</tr>
<tr>
<td>xor</td>
<td>$A ^ B$</td>
<td></td>
</tr>
</tbody>
</table>

This table contains only a subset of the modes. There are more of them, but it's difficult to understand what they actually do.
2.41 Circular mask (View through eyepiece)

This batch file simulates the view through an eyepiece of a telescope. The outside of the circular field of view is black.

```
set "IN=P1000715.mov"       :: Input video
set "SIZE=3840x2160"        :: Video size
set "D=0.7"                 :: Circle diameter relative to image height
set "OUT=out.mp4"           :: Output video

ffmpeg -f lavfi -i color=black:s=%SIZE% -lavfi format=argb,geq=a='255*gt(hypot(((2*X-W)/H),(2*Y/H)-1),%D%):r=0:g=0:b=0 -frames 1 -y mask.png
ffmpeg -i %IN% -i mask.png -lavfi overlay=format=yuv422p10 -y %OUT%
```

Note: format=yuv422p10 is only required for 10-bit videos. The default output format of the overlay filter is yuv420.

This batch file simulates the view through an eyepiece with an unsharp edge:

```
set "IN=P1000715.mov"       :: Input video
set "SIZE=3840x2160"        :: Video size
set "D=0.95"                :: Circle diameter, relative to image height
set "T=0.1"                 :: Width of smooth transition region, relative to image height
                              :: (can be made small, but not zero)
set "OUT=out.mp4"           :: Output video

ffmpeg -f lavfi -i color=black:s=%SIZE% -lavfi format=argb,geq=a='clip(128+128/%T%*(hypot(((2*X-W)/H),(2*Y/H)-1)-%D%),0,255)':r=0 -frames 1 -y mask.png
ffmpeg -i %IN% -i mask.png -lavfi overlay=format=yuv422p10 -y %OUT%
```

How is the circle with the unsharp edge made?
hypot(X-x0,Y-y0) | This is the distance from the center x0,y0 of the circle.
--- | ---
gt(hypot(X-x0,Y-y0),radius) | If you compare with a radius (in pixels), then you get a circle with a sharp edge. This function is 0 inside and 1 outside of the circle. Multiply by 255 and you are done. However if you want a smooth edge, you must get rid of the gt() function.
--- | ---
hypot(X-x0,Y-y0)-radius | This is the distance of a point from the edge of the circle. This function is negative inside the circle, it’s 0 at the edge and it’s positive outside of the circle.
--- | ---
256/width*(hypot(X-x0,Y-y0)-radius) | Multiply by a factor 256 and divide by the width of the transition band (in pixels).
--- | ---
128+256/width*(hypot(X-x0,Y-y0)-radius) | Add 128 (for middle gray). This means at the exact radius you have middle gray. If you don’t add 128, then you have black at the exact radius, and the transition is fully outside of the circle. If you add 255, then you have white at the exact radius, and the transition is fully inside of the circle.
--- | ---
clip(128+256/width*(hypot(X-x0,Y-y0)-radius),0,255) | Finally you must clip to the 0...255 range (black to white). The result is a circular mask with a smooth edge.
2.42  Binocular view simulation

This batch file simulates the view through a binocular:

set "IN=P1000715.mov"       :: Input video
set "SIZE=3840x2160"        :: Video size
set "D=0.8"                 :: Circle diameter, relative to image height
set "P=0.5"                 :: Pupil distance, relative to image height
set "T=0.1"                 :: Width of smooth transition region, relative to image height
                                   :: (can be made small, but not zero)
set "OUT=out.mp4"           :: Output video

ffmpeg -f lavfi -i color=black:s=%SIZE% -lavfi format=argb,geq=a='clip(128+128/%T%*min((hypot(((2*X-W-%P%*H)/H),(2*Y/H)-1)-%D%), (hypot(((2*X-W+%P%*H)/H),(2*Y/H)-1)-%D%)),0,255)':r=0 -frames 1 -y mask.png

ffmpeg -i %IN% -i mask.png -lavfi overlay=format=yuv422p10 -y %OUT%

pause

Note: format=yuv422p10 is only required for 10-bit videos. The default output format of the overlay filter is yuv420.
### 2.43 Subtracting a darkframe

Noise, hot pixels and amplifier glow in a low-light video can be reduced by subtracting a darkframe. Make a dark video with the same settings and at the same temperature as your main video. The only difference is that you put the cap on the lens. Then you can average many (up to 128) frames from the dark video and save the darkframe lossless as 16-bit PNG:

```plaintext
set "DARKVID=Dark.mov" :: Dark video
ffmpeg -i %DARKVID% -vf "tmix=128,format=rgb48" -frames 1 -y dark.png
```

Now you can subtract this darkframe from all frames of your video:

```plaintext
set "IN=meteor.mov" :: Input video
set "OUT=meteor-dark.mp4" :: Output video
ffmpeg -i %IN% -i dark.png -filter_complex "format=rgb48[a];[a][1]blend=subtract" -y %OUT%
```

pause
2.44 Gradation curves and vignetting

Note: This is obsolete. It’s better done with a Color look-up table.

-- An image is opened in GIMP.
-- Colors::Values --> Select suitable points for black, white and gray in the image.
-- Click on "Edit these settings as curves".
-- make fine corrections on the curves
-- Set as many points as possible on the curves, because later they will be interpolated by straight lines.
-- Click on "Export settings as file".
-- Check the box "Use old file format for curves".
-- filename: curves.gimp
-- Save

-- Then call the GIMP2ACV converter (1). This converter reads the file curves.gimp, converts it and saves it as curves.acv. The file curves.gimp must be located in the same folder where the converter is called.
-- In the batch file for the FFmpeg editing the corresponding video filter is called: -vf curves=psfile='curves.acv'

Vignetting at the edge of the image can be compensated automatically: -vf vignette=0.5:mode=backward
mode=backward makes the image corners brighter, mode=forward makes them darker. The number 0.5 must be set so that the corners are neither too bright nor too dark.

The two filters can be combined in this way:

```
-vf vignette=0.5:mode=backward,curves=psfile='curves.acv'
```

2.45  Color grading with color look-up tables, full workflow

A color look-up table (CLUT) is a mathematical rule according to which any color is replaced by another color.

There are different file formats for the CLUT:

The *.cube format normally has a color space of typically 25 * 25 * 25 or 33 * 33 * 33 entries, so that the table contains $25^3 = 15625$ or $33^3 = 35937$ different colors. Colors between the specified table entries are interpolated. You can also create tables with $64^3$ entries, but for most applications $25^3$ or $33^3$ entries are sufficient.


It's also possible to save a CLUT in any uncompressed image format. The complete workflow is now described step by step for a 10-bit video. This workflow can be simplified, see the next chapter.

Step 1: With this batch file a single image is extracted from the 10-bit video at a suitable location and saved lossless as 16-bit PNG:

```batch
set "IN=Video_62.mov"        :: Input video
set "T=35"                   :: Time where image is extracted
ffmpeg -ss %T% -i %IN% -frames 1 -y image.png
pause
```

Step 2: This batch file is used to create a CLUT (= Color-look-up-Table). This is a PNG image with 512x512 pixels that contains exactly one pixel of each possible color. I'm not yet sure if the image has to have 16 bit resolution at this point. At least it doesn't hurt. If 8 bits are enough, you would omit "-pix_fmt rgb48be".

The LEVEL parameter determines how many different colors are contained in the CLUT. The height and width of the square image is $LEVEL^3$, at LEVEL=8 there are $64^3=262144$ colors and the image has $512^2=262144$ pixels. It is important that the file is saved in an uncompressed or lossless compressed format, so PNG is well suited.
set "LEVEL=8"
ffmpeg -f lavfi -i haldclutsrc=%LEVEL% -frames 1 -pix_fmt rgb48be clut.png
pause

Step 3: The extracted image is opened in GIMP.
Step 4: The color table will be opened in GIMP, selected with "Select all" and copied with ctrl-c.
Step 5: The first image is clicked and then the color table is inserted in the upper left corner with "Paste in Place". Since the first image is much larger than the color table, the table does not interfere at this position.
Step 6: Right click on "Floating Selection" and select "To New Layer".
Step 7: Right click on the newly created "Pasted Layer" and select "Merge Down".
Step 8: Now the image is edited as it should look in the video. And of course the color table in the upper left corner will be edited as well. Color corrections, color temperature, color saturation, gradation curve, brightness, contrast. The image may contain visible noise. Later in the video, the noise doesn’t stand out so much, because it is partly averted by the fast sequence of images. Operations that cannot be described by a color look-up table, such as noise reduction, soft focus or sharpening, are not permitted.
Step 9: The finished image is trimmed to a size of 512x512 pixels so that only the color table in the upper left corner remains. Image > Canvas Size > Width=512, Height=512, then click on "Resize".
Step 10: Export the image under the name clut2.png as 16-bit PNG and select "16bpc RGB" as pixel format. GIMP can now be closed.
Step 11: This color look-up table is now applied to the whole video with FFmpeg. The color table is applied with 10 bit accuracy. Colors not included in the table are interpolated. Only then is the color table converted to 8 bit accuracy and an MP4 generated:

set "IN=Video_62.mov"       :: Input video
ffmpeg -i %IN% -i clut2.png -filter_complex [0][1]haldclut out.mp4
pause
2.46    Color grading with color look-up tables, simplified workflow

The above workflow can be simplified as follows:

Step 1: In this batch file FFmpeg does immediately combine the CLUT with the extracted image:

```
set "IN=P1000099.mov"          :: Input video
set "T=5"                      :: Time where image is extracted
ffmpeg -ss %T% -i %IN% -f lavfi -i haldclutsrc=8 -filter_complex "[1]format=pix_fmts=rgb48be[a];[a]
      [0]xstack=inputs=2:layout=0_0|w0_0" -frames 1 -y Image_with_CLUT.png
```

Step 2: This image is now processed in GIMP (or any other suitable image processing software) and then exported with the same file name as 16-bit PNG. You can edit brightness, contrast, gamma, saturation and hue. You can also adjust the curves. Of course, all modifications must be applied to the whole image consisting of the video frame and the clut. Filters like noise reduction, sharpening or softening are not allowed.

Step 3: This batch file does first use the crop filter to remove the image so that only the CLUT remains. Why the small brightness correction is necessary before applying the haldclut filter isn't yet fully understood. In the second FFmpeg run the CLUT is applied to the input video. Then the CLUT is deleted because it's no longer required.

```
set "IN=P1000099.mov"        :: Input video
set "BR=0.06"                :: Small brightness adjustment before applying the CLUT
ffmpeg -i Image_with_CLUT.png -vf crop=512:512:0:0 -y clut.png
ffmpeg -i %IN% -i CLUT.png -filter_complex [0]eq=brightness=%BR%[a];[a][1]haldclut -y out.mp4
del clut.png
pause
```
2.47 Size of color-look-up tables

The size of the Color-look-up table for the haldclut filter depends on the "Level" parameter as follows:

<table>
<thead>
<tr>
<th>Level n</th>
<th>Size of CLUT n^3 x n^3</th>
<th>Typical file size of CLUT as 16-bit PNG</th>
<th>Edge length of the RGB cube n^2</th>
<th>Number of support points (= number of pixels) n^6</th>
<th>Distance of support points 8-bit 256 / n^2</th>
<th>Distance of support points 16-bit 65536 / n^2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>64x64 Pixels</td>
<td>16.4 kB</td>
<td>16</td>
<td>4096</td>
<td>16</td>
<td>4096</td>
</tr>
<tr>
<td>5</td>
<td>125x125 Pixels</td>
<td>61.9 kB</td>
<td>25</td>
<td>15625</td>
<td>10.2</td>
<td>2621</td>
</tr>
<tr>
<td>6</td>
<td>216x216 Pixels</td>
<td>179 kB</td>
<td>36</td>
<td>46656</td>
<td>7.11</td>
<td>1820</td>
</tr>
<tr>
<td>7</td>
<td>343x343 Pixels</td>
<td>436 kB</td>
<td>49</td>
<td>117649</td>
<td>5.22</td>
<td>1337</td>
</tr>
<tr>
<td>8</td>
<td>512x512 Pixels</td>
<td>0.97 MB</td>
<td>64</td>
<td>262144</td>
<td>4</td>
<td>1024</td>
</tr>
<tr>
<td>9</td>
<td>729x729 Pixels</td>
<td>81 MB</td>
<td>81</td>
<td>531441</td>
<td>3.16</td>
<td>809</td>
</tr>
<tr>
<td>10</td>
<td>1000x1000 Pixels</td>
<td>100 MB</td>
<td>100</td>
<td>1000000</td>
<td>2.56</td>
<td>655</td>
</tr>
<tr>
<td>11</td>
<td>1331x1331 Pixels</td>
<td>121 MB</td>
<td>121</td>
<td>1771561</td>
<td>2.12</td>
<td>542</td>
</tr>
<tr>
<td>12</td>
<td>1728x1728 Pixels</td>
<td>144 MB</td>
<td>144</td>
<td>2985984</td>
<td>1.78</td>
<td>455</td>
</tr>
<tr>
<td>13</td>
<td>2197x2197 Pixels</td>
<td>169 MB</td>
<td>169</td>
<td>4826809</td>
<td>1.51</td>
<td>388</td>
</tr>
<tr>
<td>14</td>
<td>2744x2744 Pixels</td>
<td>196 MB</td>
<td>196</td>
<td>7529536</td>
<td>1.31</td>
<td>334</td>
</tr>
<tr>
<td>15</td>
<td>3375x3375 Pixels</td>
<td>225 MB</td>
<td>225</td>
<td>11390625</td>
<td>1.14</td>
<td>291</td>
</tr>
<tr>
<td>16</td>
<td>4096x4096 Pixels</td>
<td>256 MB</td>
<td>256</td>
<td>16777216</td>
<td>1</td>
<td>256</td>
</tr>
</tbody>
</table>
The size of *.cube files is as follows:

<table>
<thead>
<tr>
<th>LUT_3D_SIZE</th>
<th>Number of points</th>
<th>Typical file size</th>
<th>Distance of support points (8-bit color)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>n^3</td>
<td>256 / (n-1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>125</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>729</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>4913</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>15625</td>
<td>10.667</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>35937</td>
<td>1 MB</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>262144</td>
<td>7 MB</td>
<td>8</td>
</tr>
<tr>
<td>65</td>
<td>274625</td>
<td>7 MB</td>
<td>4</td>
</tr>
</tbody>
</table>

*.cube LUT files can be loaded into the FOTGA DP500/A70TLS monitor as follows:

Save the *.cube file in the root folder of a FAT32 USB stick. Insert the USB stick into the monitor. Press on the menu wheel, select "LUT Settings" and "LUT Import". All LUTs from the USB stick will be imported (up to 8 of them), and all already existing LUTs will be deleted, except those that are in the firmware (VLOG is one of them). Each single LUT file must be less than 7.9 MB.

It's also possible to apply LUTs to the HDMI output of the GH5S camera. Use "HDMI Rec Output" -> "apply LUT". The LUT must be saved on the SD card.
This is the identity *.cube LUT for 8 colors:

```plaintext
LUT_3D_SIZE 2
# black
0 0 0
# red
1 0 0
# green
0 1 0
# yellow
1 1 0
# blue
0 0 1
# magenta
1 0 1
# cyan
0 1 1
# white
1 1 1
```

This is the identity *.cube LUT for 27 colors:

```plaintext
LUT_3D_SIZE 3
# black
0.0 0.0 0.0
# dark red
0.5 0.0 0.0
# red
1.0 0.0 0.0
# dark green
0.0 0.5 0.0
# dark yellow
0.5 0.5 0.0
# red-green = orange
1.0 0.5 0.0
# green
0.0 1.0 0.0
# yellow-green
0.5 1.0 0.0
# yellow
1.0 1.0 0.0
# dark blue
0.0 0.0 0.5
# dark magenta
0.5 0.0 0.5
# red-magenta
1.0 0.0 0.5
# dark cyan
```
This is a C# program for creating *.cube LUT files, in this case a simple gamma function:

```csharp
using System;
using System.IO;
using System.Globalization;

namespace My_LUT_Creator
{
    class Program
    {
        static void Main(string[] args)
        {
            Double r_in, g_in, b_in, r_vlog, g_vlog, b_vlog, r_out, g_out, b_out;
            CultureInfo invC = CultureInfo.InvariantCulture;
            int size = 65; // LUT_3D_SIZE
            string filename = "My_VLOG_GAMMA=3.cube"; // Filename
            Double gamma = 3; // Gamma

            using (TextWriter outp = File.CreateText(filename))
            {
                outp.WriteLine("LUT_3D_SIZE " + size);
                for (int b = 0; b < size; b++)
                {
                    b_in = b / (size - 1.0);
                    b_vlog = (b_in - 0.125) / 0.625;
                    if (b_vlog < 0) b_vlog = 0;
                    if (b_vlog > 1) b_vlog = 1;
                    for (int g = 0; g < size; g++)
                    {
                        g_in = g / (size - 1.0);
                        g_vlog = (g_in - 0.125) / 0.625;
                        if (g_vlog < 0) g_vlog = 0;
                        if (g_vlog > 1) g_vlog = 1;
                        for (int r = 0; r < size; r++)
                        {
                            r_in = r / (size - 1.0);
                            r_vlog = (r_in - 0.125) / 0.625;
                            if (r_vlog < 0) r_vlog = 0;
                            if (r_vlog > 1) r_vlog = 1;
                            r_out = Math.Pow(r_vlog, 1.0 / gamma);
                            g_out = Math.Pow(g_vlog, 1.0 / gamma);
                            b_out = Math.Pow(b_vlog, 1.0 / gamma);
                            outp.WriteLine(String.Format(invC, "{0:F4}", r_out) + " " +
                                String.Format(invC, "{0:F4}", g_out) + " " +
                                String.Format(invC, "{0:F4}", b_out));
                        }
                    }
                }
            }
        }
    }
}
```
2.48  Histogram

This batch file generates a histogram for the R,G,B components from a video:

```
set "IN=MVI_2562.mov"       :: Input video
ffmpeg -i %IN% -vf format=pix_fmts=rgb24,histogram=levels_mode=logarithmic -y out.mp4
pause
```

2.49  Lagfun filter

The lagfun filter makes short pulses of light appear longer, with an exponential decay curve. Good for meteors in the night sky.

It works as follows:

The previous output frame is multiplied by the decay constant, which is in the range [0 ... 1] and a typical value is 0.95. This image is used as the next output frame. But if a pixel in the next input frame is brighter, then the brighter value is used. So all pixels have a fast rise time constant and a slow decay time constant. Like an oscilloscope screen with a long persistence time.

\[
\text{Time constant in seconds} = \frac{1}{(1 - \text{decay}) \times \text{framerate}}
\]

The time constant is the duration during which a signal drops from level 1.0 to \(1/e \approx 0.368\)

```
rem  Example for lagfun, left side of output video is without lagfun and right side is with lagfun
set "SN=1400"                :: Start number
set "CONTRAST=2.0"           :: Contrast in range [-1000 ... 1000], normal is 1.0
set "BRIGHT=0.22"            :: Brightness in range [-1.0 ... 1.0], normal is 0.0
set "GAMMA=2.5"              :: Gamma in range [0.1 ... 10.0], normal is 1.0
set "DEF=10"                 :: Deflicker frames
set "DECAY=0.95"             :: Decay factor
set "QU=2"                   :: MP4 quality level, 0 is best quality, 2 is normal, 9 is strong compression
set "FPS=30"                 :: Output framerate
```
set "OUT=meteors.mp4"        :: Output filename

ffmpeg -start_number %SN% -i IMG_%%4d.jpg ^
-filter_complex "eq=contrast=%CONTRAST%:brightness=%BRIGHT%:gamma=%GAMMA%:deflicker=size=%DEF%,split[a][b];
[b]lagfun=decay=%DECAY%[c];[a][c]hstack" -r 30 -codec:v mpeg4 -q:v %QU% -y %OUT%

pause

The lagfun filter has a "planes" option, but this option doesn't work with pixel format RGB24. You must use GBRP pixel format. See also the workaround in the next chapter.

In this example the lagfun filter is only applied to the green channel:

set "IN=input.mov"           :: Input video
set "DECAY=0.95"             :: Decay factor
set "OUT=out.mp4"            :: Output video

ffmpeg -i %IN% -vf "format=gbrep,lagfun=decay=%DECAY%:planes=1" -y %OUT%

pause
2.50 Star trails

The lagfun filter can also be used for making startrail videos:

```plaintext
rem  Make a small white star
ffmpeg -f lavfi -i color=white:s=2x2 -y -frames 1 star.png
rem  Make a 10 seconds video of a moving white star over a black background
ffmpeg -f lavfi -i color=black:s=1920x1080 -loop 1 -i star.png -lavfi overlay=x=10+190*t:y=10+100*t -t 10 -y star.mp4
rem  Make a startrail video
ffmpeg -i star.mp4 -vf lagfun -y startrail.mp4
```

Note: The first two command lines in this batch file are only for generating a simulated star in front of a black background. If you have a real input video, you can directly feed it to the third command line.

If you set the decay option to 1, the trains will remain for infinite time.

If you set the value slightly smaller than 1.0, for example 0.95, then the trails will decay.

It's also possible to make star trails of finite length with the tmedian filter:

```plaintext
set "R=10" :: Set radius for tmedian filter
rem  Make star trails of finite length
ffmpeg -i star.mp4 -vf tmedian=radius=%R%:percentile=1 -y startrail.mp4
```

Note: The number of frames seems to be twice the number that is specified as "radius".
2.51  Bird trails

Paul Bourke did make a nice image of bird trails here:  [http://paulbourke.net/fun/garminfun/birdtrails.jpg](http://paulbourke.net/fun/garminfun/birdtrails.jpg)

It's also possible to do this with FFmpeg's lagfun filter. Because the filter works only with bright objects in front of a dark background, I'm using here a trick: Negate the input video, apply lagfun with decay=1, then negate again.

```
set "S=5"                     :: Specify that only each n-th frame is used
rem  Make a small black bird
ffmpeg -f lavfi -i color=black:s=6x6 -y -frames 1 bird.png
rem  Make a 10 seconds video of a moving black bird over a white background
ffmpeg -f lavfi -i color=white:s=1920x1080 -loop 1 -i bird.png -lavfi overlay=x=10+190*t:y=10+100*t -t 10 -y bird.mp4
rem  Make a bird trail video
ffmpeg -i bird.mp4 -vf select='not(mod(n,%S%))',negate,lagfun=decay=1,negate -y birdtrail.mp4
```

Note: The first two command lines in this batch file are only for generating a simulated black bird in front of a white background. If you have a real input video, you can directly feed it to the third command line.

A similar effect can be achieved with the "tmedian" filter, which picks the smallest pixel value out of the last n frames. In this case the trails have a finite length. Please note that the number seems to be twice the number that you specified.

```
set "S=5"                     :: Specify that only each n-th frame is used
set "R=10"                    :: Set radius for tmedian filter
rem  Make a bird trail video, with trails of finite length
ffmpeg -i bird.mp4 -vf select='not(mod(n,%S%))',tmedian=radius=%R%:percentile=0 -y birdtrail.mp4
```

This is an example for a bird trails video from a Kodak PIXPRO SP360 4K camera:

```plaintext
set "IN=116_0002.mp4" :: Input video
set "N=6" :: Specify that only each n-th frame is used
set "FOV=235" :: Input field of view in degrees
set "YAW=-25" :: Yaw angle in degrees
set "PITCH=-50" :: Pitch angle in degrees
set "HFOV=60" :: Output horizontal field of view in degrees
set "VFOV=60" :: Output vertical field of view in degrees
set "W=800" :: Output width
set "H=800" :: Output height
set "CONTR=1.5" :: Contrast
set "BRIGHT=-0.2" :: Brightness
set "S=0" :: Start point
set "T=32" :: Duration
set "OUT=birdtrail.mp4" :: Output video

rem Make a bird trail video
ffmpeg -ss %S% -i %IN% -vf select='not(mod(n,%N%))',v360=input=fisheye:output=rectilinear:ih_fov=%FOV%:iv_fov=%FOV%:yaw=%YAW%:pitch=%PITCH%:h_fov=%HFOV%:v_fov=%VFOV%:w=%W%:h=%H%:negate,lagfun=decay=1:planes=1,negate,eq=contrast=%CONTR:brightness=%BRIGHT% -t %T% -y %OUT%

pause
```

Note: The planes=1 option for the lagfun filter means that the filter is only applied to the luminance plane. The colors stay as they are.
2.52 Rainbow-trail effect

I found the original version of this effect here: [http://oiioioioxiii.blogspot.com/2020/07/ffmpeg-improved-rainbow-trail-effect.html](http://oiioioioxiii.blogspot.com/2020/07/ffmpeg-improved-rainbow-trail-effect.html)

This is my version as a Windows batch file:

```batch
rem Make a 10 seconds test video of a white dot moving over a bluescreen
ffmpeg -f lavfi -i color=blue:s=1920x1080 -f lavfi -i color=white:s=60x60 -lavfi overlay=x=960+800*sin(t):y=540+300*sin(2*t) -t 10 -y dot.mp4

rem Rainbow-trail effect
set "IN=dot.mp4" :: Input video
set "KEY=0x0000FF" :: Color key, use 0x00FF00 for greenscreen or 0x0000FF for bluescreen
set "D=0.1" :: Delay time per color
set "OUT=out.mp4" :: Output video
set "VIOLET=colorchannelmixer=2:0:0:0:0:0:0:0:2:0:0:0"
set "INDIGO=colorchannelmixer=.5:0:0:0:0:0:0:0:2:0:0:0"
set "BLUE=colorchannelmixer=0:0:0:0:0:0:0:0:2:0:0:0"
set "GREEN=colorchannelmixer=0:0:0:0:2:0:0:0:0:0:0:0"
set "YELLOW=colorchannelmixer=0:0:0:0:2:0:0:0:0:0:0:0"
set "ORANGE=colorchannelmixer=2:0:0:0:0:0:0:0:2:0:0:0"
set "RED=colorchannelmixer=2:0:0:0:0:0:0:0:2:0:0:0"
ffmpeg -i %IN% -lavfi "split[a][b];[b]colorkey=%KEY%:0.3:0.1,extractplanes=a,split=7[b1][b2][b3][b4][b5][b6][b7];[b1]%RED%,setpts=PTS+%D*7/TB[b1];[b2]%ORANGE%,setpts=PTS+%D*6/TB,chromakey=black:0.01:0.1[b2];[b1][b2]overlay[b1];[b3]%YELLOW%,setpts=PTS+%D*5/TB,chromakey=black:0.01:0.1[b3];[b1][b3]overlay[b1];[b4]%GREEN%,setpts=PTS+%D*4/TB,chromakey=black:0.01:0.1[b4];[b1][b4]overlay[b1];[b5]%BLUE%,setpts=PTS+%D*3/TB,chromakey=black:0.01:0.1[b5];[b1][b5]overlay[b1];[b6]%INDIGO%,setpts=PTS+%D*2/TB,chromakey=black:0.01:0.1[b6];[b1][b6]overlay[b1];[b7]%VIOLET%[b7],setpts=PTS+%D*/TB,chromakey=black:0.01:0.1[b7];[b1][b7]overlay[b1];[a]colorkey=%KEY%:0.4:0.1[a];[b1][a]overlay" -y %OUT%
```

pause
2.53 Temporal slice-stacking effect

In this example the video is split into 6 horizontal slices, which are delayed by 0-5 frames. Nice effect for dancing videos.

```bash
rem Make a 10 seconds video of a white vertical bar

ffmpeg -f lavfi -i color=black:s=1920x1080 -f lavfi -i color=white:s=20x1080 -lavfi overlay=x=960+800*sin(t):y=0 -t 10 -y bar.mp4

ffmpeg -i bar.mp4 -vf "split=6[a0][a2][a3][a4][a5];[a0]crop=h=ih/6:y=0[b0];[a1]setpts=PTS+1/(FR*TB),crop=h=ih/6:y=ih/6[b1];[a2]setpts=PTS+2/(FR*TB),crop=h=ih/6:y=2*ih/6[b2];[a3]setpts=PTS+3/(FR*TB),crop=h=ih/6:y=3*ih/6[b3];[a4]setpts=PTS+4/(FR*TB),crop=h=ih/6:y=4*ih/6[b4];[a5]setpts=PTS+5/(FR*TB),crop=h=ih/6:y=5*ih/6[b5];[b0][b1][b2][b3][b4][b5]vstack=6" -y out.mp4

pause
```

For a different approach see also http://oiioiioixiii.blogspot.com/2017/11/ffmpeg-temporal-slice-stacking-effect.html

The same can also be done with the tpad filter instead of setpts:

```bash
ffmpeg -i bar.mp4 -vf "split=6[a0][a1][a2][a3][a4][a5];[a0]crop=h=ih/6:y=0[b0];[a1]tpad=1,crop=h=ih/6:y=ih/6[b1];[a2]tpad=2,crop=h=ih/6:y=2*ih/6[b2];[a3]tpad=3,crop=h=ih/6:y=3*ih/6[b3];[a4]tpad=4,crop=h=ih/6:y=4*ih/6[b4];[a5]tpad=5,crop=h=ih/6:y=5*ih/6[b5];[b0][b1][b2][b3][b4][b5]vstack=6" -y out.mp4

pause
```

The main idea in the above script is to combine the video with one or more delayed versions of itself:

```bash
ffmpeg -i test.mp4 -vf "split[a][b];[b]setpts=PTS+5/(FR*TB)[c];[a][c]vstack" -y out.mp4
```

Or with tpad filter:

```bash
ffmpeg -i test.mp4 -vf "split[a][b];[b]tpad=start=5:start_mode=clone[c][a][c]vstack" -y out.mp4
```

The two above examples consume less memory if the split filter is omitted, and instead the same input video is loaded twice:
ffmpeg -i test.mp4 -i test.mp4 -vf "[0]setpts=PTS+5/(FR*TB)[a];[a][1]vstack" -y out.mp4
2.54 Extract and merge planes, split planes

Extract RGB channels, apply a filter to the G channel, then merge all channels to the output video:

```plaintext
set "IN=input.mov"          :: Input video
set "DECAY=0.95"             :: Decay factor
set "OUT=out.mp4"            :: Output video

ffmpeg -i %IN% -lavfi "format=rgb24,extractplanes=r+g+b[r][g][b];[g]lagfun=decay=%DECAY%[gg][gg][b]
[r]mergeplanes=0x001020:gbrp" -y %OUT%
```

Use different delay for RGB planes:

```plaintext
set "IN=test.mp4"          :: Input video
set "DELAY_R=2"            :: Number of delayed frames for red
set "DELAY_G=1"            :: Number of delayed frames for green
set "OUT=out.mp4"          :: Output video

ffmpeg -i %IN% -lavfi "format=rgb24,extractplanes=r+g+b[r][g][b];[r]tpad=%DELAY_R%[rr];[g]tpad=%DELAY_G%[gg][gg][b]
[rr]mergeplanes=0x001020:gbrp" -y %OUT%
```
Bluescreen / greenscreen

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BG</td>
<td>background.mov</td>
<td>Background video</td>
</tr>
<tr>
<td>S1</td>
<td>10</td>
<td>Start time for background video</td>
</tr>
<tr>
<td>BGSAT</td>
<td>1.4</td>
<td>Saturation for background</td>
</tr>
<tr>
<td>BLUE</td>
<td>blue.mov</td>
<td>Foreground video with blue screen</td>
</tr>
<tr>
<td>S2</td>
<td>13</td>
<td>Start time for foreground video</td>
</tr>
<tr>
<td>CW</td>
<td>800</td>
<td>Crop width</td>
</tr>
<tr>
<td>CH</td>
<td>1080</td>
<td>Crop height</td>
</tr>
<tr>
<td>CX</td>
<td>500</td>
<td>Crop X</td>
</tr>
<tr>
<td>CY</td>
<td>0</td>
<td>Crop Y</td>
</tr>
<tr>
<td>COLOR</td>
<td>0x223395</td>
<td>Measured average blue screen color in format 0xRRGGBB</td>
</tr>
<tr>
<td>SIM</td>
<td>0.11</td>
<td>Similarity for blue color</td>
</tr>
<tr>
<td>SC</td>
<td>0.35</td>
<td>Scale factor for foreground video</td>
</tr>
<tr>
<td>X</td>
<td>800</td>
<td>X Position of foreground</td>
</tr>
<tr>
<td>Y</td>
<td>310</td>
<td>Y Position of foreground</td>
</tr>
<tr>
<td>T</td>
<td>28</td>
<td>Duration</td>
</tr>
<tr>
<td>OUT</td>
<td>out.mp4</td>
<td>Output video</td>
</tr>
</tbody>
</table>

**Note:** For measuring the average color of the bluescreen, you can extract an image and save it as PNG. Open this image with Fitswork, draw a rectangle and then make a right click in the rectangle and choose "Statistik für den Bereich anzeigen".

**Note:** The normalization of chromakey's "similarity" value was changed in May 2020. Old values must now be divided by sqrt(2) to get the same result as before.
I did try to insert an "eq=gamma=1.4" filter after the scale filter, but that didn't work. It seems that the eq filter destroys the alpha channel. This workaround works with "alphaextract" and "alphamerge" filters:

```bash
ffmpeg -ss %S1% -i %BG% -ss %S2% -i %BLUE% -filter_complex "[0]eq=saturation=%BGSAT%[BG];[1]crop=%CW%:%CH%:%CX%:%CY%;chromakey=%COLOR%:%SIM%,scale=iw*%SC%:ih*%SC%,format=rgba,split[FG1][FG2];[FG1]alphaextract[A];[FG2]eq=gamma=1.4[FG3];[FG3][A]alphamerge[FG4];[BG][FG4]overlay=%X%:%Y%" -t %T% -y %OUT%
```

Note: When the person in the bluescreen video makes fast movements, it's best to use short exposure times. Otherwise the fast moving object gets smeared with the blue background, and in extreme cases might become so blue that it's detected as background.

See also: despill filter, colorkey filter

The documentation for the "despill" filter is rather incomplete:

<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>int</td>
<td>set the screen type (from 0 to 1) (default green)</td>
</tr>
<tr>
<td>green</td>
<td>0</td>
<td>greenscreen</td>
</tr>
<tr>
<td>blue</td>
<td>1</td>
<td>bluescreen</td>
</tr>
<tr>
<td>mix</td>
<td>float</td>
<td>set the spillmap mix (from 0 to 1) (default 0.5)</td>
</tr>
<tr>
<td>expand</td>
<td>float</td>
<td>set the spillmap expand (from 0 to 1) (default 0)</td>
</tr>
<tr>
<td>red</td>
<td>float</td>
<td>set red scale (from -100 to 100) (default 0)</td>
</tr>
<tr>
<td>green</td>
<td>float</td>
<td>set green scale (from -100 to 100) (default -1)</td>
</tr>
<tr>
<td>blue</td>
<td>float</td>
<td>set blue scale (from -100 to 100) (default 0)</td>
</tr>
<tr>
<td>brightness</td>
<td>float</td>
<td>set brightness (from -10 to 10) (default 0)</td>
</tr>
<tr>
<td>alpha</td>
<td>boolean</td>
<td>change alpha component (default false)</td>
</tr>
</tbody>
</table>

Some more informations are available through the command ffmpeg -h filter=despill

This filter has support for timeline through the 'enable' option.
This is the same bluescreen example as before, with additional despill filter:

```plaintext
set "BG=background.mov"       :: Background video
set "S1=10"                   :: Start time for background video
set "BGSAT=1.4"               :: Saturation for background
set "BLUE=blue.mov"           :: Foreground video with blue screen
set "S2=13"                   :: Start time for foreground video
set "CW=800"                  :: Crop width
set "CH=1080"                 :: Crop height
set "CX=500"                  :: Crop X
set "CY=0"                    :: Crop Y
set "COLOR=0x223395"          :: Measured average blue screen color
set "SIM=0.12"                :: Similarity for blue color
set "SC=0.35"                 :: Scale factor for foreground video
set "D_TYPE=blue"             :: Despill type, blue or green
set "D_MIX=0.7"               :: Despill mix parameter
set "D_EXP=1.0"               :: Despill expand parameter
set "D_BR=1.0"                :: Despill brightness parameter
set "D_GREEN=0"               :: Despill green parameter, must be -1 for greenscreen
set "D_BLUE=-1"               :: Despill blue parameter, must be -1 for bluescreen
set "X=800"                   :: X Position
set "Y=310"                   :: Y Position
set "T=28"                    :: Duration
set "OUT=out.mp4"             :: Output video

rem Extract an image from the bluescreen video, for measuring the average blue screen color

rem ffmpeg -ss 10 -i %BLUE% -vf crop=%CW%:%CH%:%CX%:%CY% -frames 1 -y image.png

ffmpeg -ss %S1% -i %BG% -ss %S2% -i %BLUE% -filter_complex "][0]eq=saturation=%BGSAT%[BG];[1]crop=%CW%:%CH%:%CX%:%CY%
,chromakey=%COLOR%:%SIM%,despill:type=%D_TYPE%:mix=%D_MIX%:expand=%D_EXP%:brightness=%D_BR
:green=%D_GREEN%:blue=%D_BLUE%,scale=iw*%SC%:ih*%SC%[FG];[BG][FG]overlay=%X%:%Y%" -t %T% -y %OUT%

pause
```
This is an example for real-time bluescreen processing. The background video comes from a file and the foreground video comes from the Panasonic GH5S camera via a HDMI to USB converter. I'm here using FFplay instead of FFmpeg, so that the result is visible in real time:

```plaintext
set "BG=background.mov"       :: Background video
set "LOOP_N=50"               :: Background video: Number of frames in loop
set "COLOR=0x0000ff"          :: Bluescreen color
set "SIM=0.35"                :: Similarity for blue color: larger value means more is recognized as background
set "SC=0.6"                  :: Scale factor for foreground video
set "D_TYPE=blue"             :: Despill type, blue or green
set "D_MIX=0.7"               :: Despill mix parameter
set "D_EXP=1.0"               :: Despill expand parameter
set "D_BR=1.0"                :: Despill brightness parameter
set "D_GREEN=0"               :: Despill green parameter, must be -1 for greenscreen
set "D_BLUE=-1"               :: Despill blue parameter, must be -1 for bluescreen
set "X=0"                     :: X Position
set "Y=0"                     :: Y Position

rem ffplay -f dshow -video_size 1920x1080 -framerate 30 -vcodec mjpeg video="USB Video"
rem ffplay -f lavfi movie=filename="video=USB Video":f=dshow
ffplay -f lavfi movie=filename="video=USB Video":f=dshow:discontinuity=0.5, scale=iw*0.5*%SC%:ih*0.5*%SC%, chromakey=%COLOR%:%SIM%, despill=type=%D_TYPE%:mix=%D_MIX%:expand=%D_EXP%:brightness=%D_BR%:green=%D_GREEN%:blue=%D_BLUE% [FG];movie=%BG%, loop=-1:%LOOP_N%, scale=960x540[BG]; [BG][FG] overlay=%X%:%Y%

Why is the "movie" source used in this example? That's because FFplay doesn't allow "filter_complex", which means you have only one input stream. But the workaround with the "movie" source inside "lavfi" allows multiple inputs. The drawback of this method is that you can't specify the properties of the input device, which means you can't tell the HDMI to USB converter which size, framerate and codec it shall use. It seems it uses some default values.

It's better to use FFmpeg with the sdl2 output devive:

```plaintext
 ffmpeg -f dshow -video_size 1920x1080 -framerate 30 -vcodec mjpeg -i video="USB Video" -i %BG% -lavfi [0]scale=iw*0.5*%SC%:ih*0.5*%SC%, chromakey=%COLOR%:%SIM%, despill=type=%D_TYPE%:mix=%D_MIX%:expand=%D_EXP%:brightness=%D_BR%:green=%D_GREEN%:blue=%D_BLUE%[FG]; [1]loop=-1:%LOOP_N%, scale=960x540[BG]; [BG][FG] overlay=%X%:%Y%, format=rgb24 -window_x 0 -window_y 0 -f sdl2 -
```

83
Same as before, but use low-framerate uncompressed output from the HDMI to USB converter:

```bash
ffmpeg -f dshow -video_size 1920x1080 -framerate 5 -pixel_format yuyv422 -i video="USB Video" -i %BG% -lavfi [0]scale=iw*0.5*%SC%:ih*0.5*%SC%,chromakey=%COLOR%:%SIM%,despill=type=%D_TYPE%:mix=%D_MIX%:expand=%D_EXP%:brightness=%D_BR%:green=%D_GREEN%:blue=%D_BLUE%[FG];[1]loop=-1:%LOOP_N%,scale=960x540[BG];[BG][FG]overlay=%X%:%Y%,format=rgb24 -window_x 0 -window_y 0 -f sdl2 -
```

How does the "despill" algorithm work?

```python
def despill_factor(spillmix, spillexpand):
    factor = (1 - spillmix) * (1 - spillexpand)

    if type == "bluescreen"
        spillmap = blue - (red * spillmix + green * factor)
    else
        spillmap = green - (red * spillmix + blue * factor)

    if spillmap < 0:
        spillmap = 0;

    red = red + spillmap * (redsclae + brightness)
    green = green + spillmap * (greenscale + brightness)
    blue = blue + spillmap * (bluescale + brightness)

    if alpha == true
        alpha = 1 - spillmap
```

It's difficult to understand, and it seems to be totally different from the algorithm described here (in German):


This table shows the spillmap value for 7 input colors and different values for "mix" and "expand", for type = bluescreen and brightness = 0. All non-zero spillmap values are marked in yellow.

"spillmap" is for the original formula: spillmap = blue - (red * spillmix + green * factor)
"spillmap2" is for a modified formula: spillmap2 = blue - (red * spillmix + blue * factor)
Differences between "spillmap" and "spillmap2" are marked with <--
| Input: R=0.30 G=0.30 B=0.30 gray | mix=0.00 | expand=0.00 | spillmap=0.00 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.40 blue-gray | mix=0.00 | expand=0.00 | spillmap=0.10 | spillmap2=0.00 |
| Input: R=0.30 G=0.40 B=0.30 green-gray | mix=0.00 | expand=0.00 | spillmap=0.10 | spillmap2=0.00 |
| Input: R=0.40 G=0.40 B=0.30 yellow-gray | mix=0.00 | expand=0.50 | spillmap=0.15 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.30 gray | mix=0.00 | expand=0.00 | spillmap=0.15 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.40 blue-gray | mix=0.00 | expand=0.00 | spillmap=0.25 | spillmap2=0.00 |
| Input: R=0.30 G=0.40 B=0.30 green-gray | mix=0.00 | expand=0.00 | spillmap=0.25 | spillmap2=0.00 |
| Input: R=0.40 G=0.40 B=0.30 yellow-gray | mix=0.00 | expand=0.50 | spillmap=0.10 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.30 gray | mix=0.00 | expand=1.00 | spillmap=0.30 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.40 blue-gray | mix=0.00 | expand=1.00 | spillmap=0.40 | spillmap2=0.00 |
| Input: R=0.30 G=0.40 B=0.30 green-gray | mix=0.00 | expand=1.00 | spillmap=0.30 | spillmap2=0.00 |
| Input: R=0.40 G=0.40 B=0.30 yellow-gray | mix=0.00 | expand=1.00 | spillmap=0.30 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.30 gray | mix=0.00 | expand=1.00 | spillmap=0.30 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.40 blue-gray | mix=0.00 | expand=1.00 | spillmap=0.40 | spillmap2=0.00 |
| Input: R=0.30 G=0.40 B=0.30 green-gray | mix=0.00 | expand=1.00 | spillmap=0.30 | spillmap2=0.00 |
| Input: R=0.40 G=0.40 B=0.30 yellow-gray | mix=0.00 | expand=1.00 | spillmap=0.30 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.30 gray | mix=0.50 | expand=0.00 | spillmap=0.00 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.40 blue-gray | mix=0.50 | expand=0.00 | spillmap=0.10 | spillmap2=0.00 |
| Input: R=0.30 G=0.40 B=0.30 green-gray | mix=0.50 | expand=0.00 | spillmap=0.10 | spillmap2=0.00 |
| Input: R=0.40 G=0.40 B=0.30 yellow-gray | mix=0.50 | expand=0.00 | spillmap=0.10 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.30 gray | mix=0.50 | expand=0.50 | spillmap=0.08 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.40 blue-gray | mix=0.50 | expand=0.50 | spillmap=0.18 | spillmap2=0.00 |
| Input: R=0.30 G=0.40 B=0.30 green-gray | mix=0.50 | expand=0.50 | spillmap=0.18 | spillmap2=0.00 |
| Input: R=0.40 G=0.40 B=0.30 yellow-gray | mix=0.50 | expand=0.50 | spillmap=0.18 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.30 gray | mix=0.50 | expand=1.00 | spillmap=0.15 | spillmap2=0.00 |
| Input: R=0.30 G=0.30 B=0.40 blue-gray | mix=0.50 | expand=1.00 | spillmap=0.25 | spillmap2=0.00 |
| Input: R=0.30 G=0.40 B=0.30 green-gray | mix=0.50 | expand=1.00 | spillmap=0.15 | spillmap2=0.00 |
Even after seeing these results, it's still difficult to describe what the mix and expand parameters do:

- If mix=0, then more or less all colors are despilled (not only blue).
- If mix=1, then the expand value doesn't care.
- Useful mix values seem to be in the range 0.5 to 1.0
- Using mix=0 and expand=0 doesn't deactivate the despill filter with the original formula. But it does so with the modified formula.
- If expand=1, the results are identical for the original and the modified formula.
Here is the C# source code for making the above table:

```csharp
using System;
using System.Windows.Forms;
using System.Globalization;

namespace despill
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            block(0.0, 0.0);
            block(0.0, 0.5);
            block(0.0, 1.0);
            block(0.5, 0.0);
            block(0.5, 0.5);
            block(0.5, 1.0);
            block(1.0, 0.0);
            block(1.0, 0.5);
            block(1.0, 1.0);
        }

        void block(double mix, double exp)
        {
            desp(0.3, 0.3, 0.3, "gray        ", mix, exp);
            desp(0.3, 0.3, 0.4, "blue-gray   ", mix, exp);
            desp(0.3, 0.4, 0.3, "green-gray  ", mix, exp);
            desp(0.3, 0.4, 0.4, "cyan-gray   ", mix, exp);
            desp(0.4, 0.3, 0.3, "red-gray    ", mix, exp);
            desp(0.4, 0.3, 0.4, "magenta-gray", mix, exp);
            desp(0.4, 0.4, 0.3, "yellow-gray ", mix, exp);
            richTextBox1.AppendText("\n");
        }
    }
}
```
```csharp
void desp(double r, double g, double b, string color, double mix, double exp)
{
    CultureInfo invC = CultureInfo.InvariantCulture;
    richTextBox1.AppendText("Input: ");
    richTextBox1.AppendText("R=") + r.ToString("F2", invC) + " ");
    richTextBox1.AppendText("G=") + g.ToString("F2", invC) + " ");
    richTextBox1.AppendText("B=" + b.ToString("F2", invC) + " ");
    richTextBox1.AppendText(color + " ");
    richTextBox1.AppendText("mix=") + mix.ToString("F2", invC) + " ");
    richTextBox1.AppendText("expand=") + exp.ToString("F2", invC) + " ");
    double factor = (1 - mix) * (1 - exp);
    double map = b - (r * mix + g * factor);
    if (map < 0) map = 0;
    richTextBox1.AppendText("spillmap=") + map.ToString("F2", invC) + " ");
    map = b - (r * mix + b * factor);
    if (map < 0) map = 0;
    richTextBox1.AppendText("spillmap2=") + map.ToString("F2", invC) + " ");
    richTextBox1.AppendText("\n");
}
```
In this example I did try different values \((0, 0.3, 0.5, 0.7, 1.0)\) for the "mix" and "expand" parameters. The "brightness" parameter was set to 0 and the "blue" parameter was -1. My arm was moving fast in front of a bluescreen, and so it got smeared with the blue color. The three images marked in red
rectangles show a small improvement.
This is a test for different values of the brightness parameter (0, 1 and 2), for mix = 0.7, expand = 1.0, red = 0, green = 0, blue = -1.
2.56 Real-time bluescreening

This is an example of a C# program as a real-time GUI for FFmpeg. FFmpeg gets a live foreground video from the GH5S camera via HDMI to USB converter, and the background video is looped from a file. The GUI has scrollbars for scaling and moving the foreground video, and for choosing the parameters for colorkey and despill. The parameters can be changed in real time and are sent to the FFmpeg process via ZMQ.

Hint: If you click on "Start FFmpeg" and nothing happens, you may have forgotten to plug in the HDMI to USB converter. You must plug it in after you have started the computer. It doesn't work if it's already plugged in when you start the computer. However it's not necessary to plug in a HDMI signal from a camera, because the converter has a built-in test image (8 color bars).
Form1.cs:

```csharp
using NetMQ;  // use Nuget to add the NetMQ package to this project
using NetMQ.Sockets;
using System;
using System.IO;
using System.Diagnostics;
using System.Globalization;
using System.Windows.Forms;
using System.Runtime.InteropServices;
namespace FFbluescreen
{
    public partial class Form1 : Form
    {
        // global variables:
        Process p;
        RequestSocket client;
        string frames;
        string background = "";
        string hexColor, similarity;
        string despillType, hdmiMode, mix, expand, brightness, red, green, blue, alpha;
        string width;
        double scale;
        string x, y;
        int timeout;
        CultureInfo InvC = CultureInfo.InvariantCulture;
        string message;
        public Form1()
        {
            InitializeComponent();
            readAllControls();
            timeout = 500;  // there are error messages if this value is set too small
            comboBox1.SelectedIndex = 0;  // default is "blue"
            comboBox2.SelectedIndex = 0;  // default is "mjpeg 30fps"
        }

        private void Form1_Shown(object sender, EventArgs e)
        {
            client = new RequestSocket();
            client.Connect("tcp://localhost:5555");  // This works, but I don't understand it:
            // What does "tcp://" mean? What does "localhost:" mean?
            // Why 5555? If you can explain it, please let me know
            string[] arguments = Environment.GetCommandLineArgs();  // get the background video by command line argument,
            if (arguments.Length == 2)  // this works also for drag-and-drop
                background = arguments[1];
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
        }
    }
}
```
if ((p != null) && !p.HasExited)
p.Kill();

private void button1_Click(object sender, EventArgs e) // start FFmpeg process
{
    if (background == "")
        MessageBox.Show("You must first select a background video!");
    else
    {
        if (p == null)
        {
            ProcessStartInfo startInfo = new ProcessStartInfo();
            if (checkBox1.Checked)
                startInfo.UseShellExecute = true; // enable console window
            else
                startInfo.UseShellExecute = false; // disable console window
            startInfo.CreateNoWindow = true;
            startInfo.FileName = "ffmpeg";
            startInfo.Arguments = FFmpegArguments();
            p = Process.Start(startInfo);

            // Now try to move the console window to the right
            System.Threading.Thread.Sleep(2000);
            WinAPI.MoveWindow(p.MainWindowHandle, 960, 0, 960, 540, true); // This doesn't work as expected. It moves the video window.
            // If you know how to move the console window, please let me know.
        }
    }
}

public class WinAPI
{
    [DllImport("user32.dll")]
    public static extern bool MoveWindow(IntPtr hWnd, int X, int Y, int nWidth, int nHeight, bool bRepaint);
}

private string FFmpegArguments()
{
    readAllControls();
    string arguments = "-f dshow -video_size 1920x1080 ";
    if (hdmiMode == "mjpeg 30fps")
        arguments += "-framerate 30 -vcodec mjpeg ";
    else
        arguments += "-framerate 5 -pixel_format yuyv422 ";
    arguments += "-i video="/"USB Video"" ";
    arguments += "-lavfi [0]zmq, colorkey@my=color=" + hexColor;
}
arguments += ":similarity=" + similarity;
arguments += ",despill@my=type=" + despillType;
arguments += ":mix=" + mix;
arguments += ":expand=" + expand;
arguments += ":brightness=" + brightness;
arguments += ":red=" + red;
arguments += ":green=" + green;
arguments += ":blue=" + blue;
arguments += ":alpha=" + alpha;
arguments += ",scale@my=" + width + ":-1[FG]";
arguments += ";[1]loop=-1:" + frames + "[BG]";
arguments += ";[BG][FG]overlay@my=" + x + ":" + y + ";
if (checkBox3.Checked)
    arguments += ",crop=960:540"
else
    arguments += ",scale=960x540"

Console.WriteLine(arguments);
return (arguments);

private void button2_Click(object sender, EventArgs e)    // stop FFmpeg process
{
    if ((p != null) && !p.HasExited)
    {
        p.Kill();
        p = null;
    }
}

private void button3_Click(object sender, EventArgs e)    // select background video
{
    openFileDialog1.ShowDialog();
    background = openFileDialog1.FileName;
}

private void vScrollBar1_Scroll(object sender, ScrollEventArgs e)    // colorkey R
{
    sendcolor();
}

private void vScrollBar2_Scroll(object sender, ScrollEventArgs e)    // colorkey G
{
    sendcolor();
}

private void vScrollBar3_Scroll(object sender, ScrollEventArgs e)    // colorkey B
{
    sendcolor();
}
public void sendcolor()  // colorkey color
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendMessage("colorkey@my color " + hexColor);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar4_Scroll(object sender, ScrollEventArgs e)  // colorkey similarity
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendMessage("colorkey@my similarity " + similarity);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)  // despill type
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendMessage("despill@my type " + despillType);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar7_Scroll(object sender, ScrollEventArgs e)  // despill mix
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendMessage("despill@my mix " + mix);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar8_Scroll(object sender, ScrollEventArgs e)  // despill expand
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendMessage("despill@my expand " + expand);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}
private void vScrollBar9_Scroll(object sender, ScrollEventArgs e) // despill brightness
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("despill@my brightness " + brightness);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar11_Scroll(object sender, ScrollEventArgs e) // despill red
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("despill@my red " + red);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar12_Scroll(object sender, ScrollEventArgs e) // despill green
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("despill@my green " + green);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar13_Scroll(object sender, ScrollEventArgs e) // despill blue
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("despill@my blue " + blue);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void checkBox2_CheckedChanged(object sender, EventArgs e) // despill alpha
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("despill@my alpha " + alpha);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}
private void vScrollBar10_Scroll(object sender, ScrollEventArgs e) // scale
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("scale@my w " + width);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
        client.SendFrame("overlay@my x " + x);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
        client.SendFrame("overlay@my y " + y);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar5_Scroll(object sender, ScrollEventArgs e) // overlay x
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("overlay@my x " + x);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void vScrollBar6_Scroll(object sender, ScrollEventArgs e) // overlay y
{
    readAllControls();
    if ((p != null) && !p.HasExited)
    {
        client.SendFrame("overlay@my y " + y);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(timeout), out message);
    }
}

private void button4_Click(object sender, EventArgs e) // write FFmpeg command line to batch file
{
    File.WriteAllText("bluescreen.bat", "ffmpeg "+ FFmpegArguments() + 
    + "pause
    
    void readAllControls()
    {
        frames = numericUpDown1.Value.ToString();
        scale = (110 - vScrollBar10.Value) * 0.01;
        width = (1920 * scale).ToString("F0", InvC);
        textBox6.Text = (100 * scale).ToString("F0", InvC) + "%";
        hexColor = "0x" + (255 - vScrollBar1.Value).ToString("X2") + // Make the hex color string
Form1.Designer.cs:

```csharp
(255 - vScrollBar2.Value).ToString("X2") +
(255 - vScrollBar3.Value).ToString("X2");
textBox1.Text = hexColor;

similarity = (1.0 - vScrollBar4.Value * 0.01).ToString("F2", InvC);
textBox3.Text = similarity;

x = (vScrollBar5.Value * 0.01 * 1920 * (1.0 - scale)).ToString("F0", InvC);
textBox7.Text = x;

y = (vScrollBar6.Value * 0.01 * 1080 * (1.0 - scale)).ToString("F0", InvC);
textBox8.Text = y;

mix = (1.0 - vScrollBar7.Value * 0.01).ToString("F2", InvC);
textBox2.Text = mix;

expand = (1.0 - vScrollBar8.Value * 0.01).ToString("F2", InvC);
textBox4.Text = expand;

brightness = (vScrollBar9.Value * -0.05).ToString("F2", InvC);
textBox5.Text = brightness;

red = (vScrollBar11.Value * -0.01).ToString("F2", InvC);
textBox9.Text = red;

green = (vScrollBar12.Value * -0.01).ToString("F2", InvC);
textBox10.Text = green;

blue = (vScrollBar13.Value * -0.01).ToString("F2", InvC);
textBox11.Text = blue;

despillType = comboBox1.Text;
hdmiMode = comboBox2.Text;

if (checkBox2.Checked)
    alpha = "true";
else
    alpha = "false";
```
private System.ComponentModel.IContainer components = null;

Verwendete Ressourcen bereinigen.

protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#region Vom Windows Form-Designer generierter Code

private void InitializeComponent()
{
    this.button1 = new System.Windows.Forms.Button();
    this.vScrollBar1 = new System.Windows.Forms.VScrollBar();
    this.vScrollBar2 = new System.Windows.Forms.VScrollBar();
    this.label1 = new System.Windows.Forms.Label();
    this.vScrollBar3 = new System.Windows.Forms.VScrollBar();
    this.label2 = new System.Windows.Forms.Label();
    this.label3 = new System.Windows.Forms.Label();
    this.textBox1 = new System.Windows.Forms.TextBox();
    this.checkBox1 = new System.Windows.Forms.CheckBox();
    this.vScrollBar5 = new System.Windows.Forms.VScrollBar();
    this.label5 = new System.Windows.Forms.Label();
    this.label6 = new System.Windows.Forms.Label();
    this.vScrollBar6 = new System.Windows.Forms.VScrollBar();
    this.vScrollBar4 = new System.Windows.Forms.VScrollBar();
    this.label8 = new System.Windows.Forms.Label();
    this.textBox3 = new System.Windows.Forms.TextBox();
    this.vScrollBar7 = new System.Windows.Forms.VScrollBar();
    this.vScrollBar8 = new System.Windows.Forms.VScrollBar();
    this.vScrollBar9 = new System.Windows.Forms.VScrollBar();
    this.label9 = new System.Windows.Forms.Label();
    this.label10 = new System.Windows.Forms.Label();
    this.label11 = new System.Windows.Forms.Label();
    this.label12 = new System.Windows.Forms.Label();
    this.label13 = new System.Windows.Forms.Label();
    this.label14 = new System.Windows.Forms.Label();
    this.textBox2 = new System.Windows.Forms.TextBox();

#endregion
this.textBox4 = new System.Windows.Forms.TextBox();
this.textBox5 = new System.Windows.Forms.TextBox();
this.label15 = new System.Windows.Forms.Label();
this.vScrollBar10 = new System.Windows.Forms.VScrollBar();
this.textBox6 = new System.Windows.Forms.TextBox();
this.textBox7 = new System.Windows.Forms.TextBox();
this.textBox8 = new System.Windows.Forms.TextBox();
this.vScrollBar11 = new System.Windows.Forms.VScrollBar();
this.textBox9 = new System.Windows.Forms.TextBox();
this.label14 = new System.Windows.Forms.Label();
this.label17 = new System.Windows.Forms.Label();
this.textBox10 = new System.Windows.Forms.TextBox();
this.vScrollBar12 = new System.Windows.Forms.VScrollBar();
this.label16 = new System.Windows.Forms.Label();
this.textBox11 = new System.Windows.Forms.TextBox();
this.vScrollBar13 = new System.Windows.Forms.VScrollBar();
this.checkBox2 = new System.Windows.Forms.CheckBox();
this.comboBox1 = new System.Windows.Forms.ComboBox();
this.label17 = new System.Windows.Forms.Label();
this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();
this.button3 = new System.Windows.Forms.Button();
this.numericUpDown1 = new System.Windows.Forms.NumericUpDown();
this.label18 = new System.Windows.Forms.Label();
this.label19 = new System.Windows.Forms.Label();
this.button2 = new System.Windows.Forms.Button();
this.comboBox2 = new System.Windows.Forms.ComboBox();
this.checkBox3 = new System.Windows.Forms.CheckBox();
((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).BeginInit();
this.SuspendLayout();
// button1
this.button1.Location = new System.Drawing.Point(8, 296);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(168, 36);
this.button1.TabIndex = 0;
this.button1.Text = "Start FFmpeg";
this.button1.UseVisualStyleBackColor = true;
this.button1.Click += new System.EventHandler(this.button1_Click); // vScrollBar1
this.vScrollBar1.LargeChange = 1;
this.vScrollBar1.Location = new System.Drawing.Point(192, 96);
this.vScrollBar1.Maximum = 255;
this.vScrollBar1.Name = "vScrollBar1";

// Start FFmpeg
this.button1_Click(sender, e) {
    // Initialize FFmpeg
    // Set up command line arguments
    // Execute FFmpeg
    // Get output
    // Close FFmpeg
}

// Close Form
this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.FormClosing);
```csharp
this.vScrollBar1.Size = new System.Drawing.Size(21, 354);
this.vScrollBar1.TabIndex = 2;
this.vScrollBar1.Value = 255;
this.vScrollBar1.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar1_Scroll);

// vScrollBar2

this.vScrollBar2.LargeChange = 1;
this.vScrollBar2.Location = new System.Drawing.Point(224, 96);
this.vScrollBar2.Maximum = 255;
this.vScrollBar2.Name = "vScrollBar2";
this.vScrollBar2.Size = new System.Drawing.Size(24, 354);
this.vScrollBar2.TabIndex = 3;
this.vScrollBar2.Value = 255;
this.vScrollBar2.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar2_Scroll);

// label1

this.label1.AutoSize = true;
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label1.Location = new System.Drawing.Point(192, 32);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(21, 20);
this.label1.TabIndex = 4;
this.label1.Text = "R";

// vScrollBar3

this.vScrollBar3.LargeChange = 1;
this.vScrollBar3.Location = new System.Drawing.Point(256, 96);
this.vScrollBar3.Maximum = 255;
this.vScrollBar3.Name = "vScrollBar3";
this.vScrollBar3.Size = new System.Drawing.Size(24, 354);
this.vScrollBar3.TabIndex = 5;
this.vScrollBar3.Value = 255;
this.vScrollBar3.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar3_Scroll);

// label2

this.label2.AutoSize = true;
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label2.Location = new System.Drawing.Point(224, 32);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(22, 20);
this.label2.TabIndex = 6;
this.label2.Text = "G";

// label3

```


```csharp
this.label3.Location = new System.Drawing.Point(256, 32);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(25, 24);
this.label3.TabIndex = 7;
this.label3.Text = "B";

this.textBox1.Location = new System.Drawing.Point(184, 56);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(96, 26);
this.textBox1.TabIndex = 12;

this.checkBox1.AutoSize = true;
this.checkBox1.Location = new System.Drawing.Point(8, 384);
this.checkBox1.Name = "checkBox1";
this.checkBox1.Size = new System.Drawing.Size(177, 17);
this.checkBox1.TabIndex = 15;
this.checkBox1.Text = "Show FFmpeg Console Window";
this.checkBox1.UseVisualStyleBackColor = true;

this.vScrollBar5.LargeChange = 1;
this.vScrollBar5.Location = new System.Drawing.Point(832, 96);
this.vScrollBar5.Name = "vScrollBar5";
this.vScrollBar5.Size = new System.Drawing.Size(24, 354);
this.vScrollBar5.TabIndex = 16;
this.vScrollBar5.Value = 50;
this.vScrollBar5.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar5_Scroll);

this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(840, 32);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(16, 20);
this.label5.TabIndex = 17;
this.label5.Text = "x";
```


this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(896, 32);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(16, 20);
this.label6.TabIndex = 18;
this.label6.Text = "y";
//
// vScrollBar6
//
this.vScrollBar6.LargeChange = 1;
this.vScrollBar6.Location = new System.Drawing.Point(888, 96);
this.vScrollBar6.Name = "vScrollBar6";
this.vScrollBar6.Size = new System.Drawing.Size(24, 354);
this.vScrollBar6.TabIndex = 19;
this.vScrollBar6.Value = 50;
this.vScrollBar6.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar6_Scroll);
//
// vScrollBar4
//
this.vScrollBar4.LargeChange = 1;
this.vScrollBar4.Location = new System.Drawing.Point(304, 96);
this.vScrollBar4.Name = "vScrollBar4";
this.vScrollBar4.Size = new System.Drawing.Size(24, 354);
this.vScrollBar4.TabIndex = 20;
this.vScrollBar4.Value = 65;
this.vScrollBar4.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar4_Scroll);
//
// label8
//
this.label8.Location = new System.Drawing.Point(280, 32);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(72, 24);
this.label8.TabIndex = 22;
this.label8.Text = "similarity";
//
// textBox3
//
this.textBox3.Location = new System.Drawing.Point(288, 56);
this.textBox3.Name = "textBox3";
this.textBox3.Size = new System.Drawing.Size(49, 26);
this.textBox3.TabIndex = 23;
//
// vScrollBar7
//
this.vScrollBar7.LargeChange = 1;
this.vScrollBar7.Location = new System.Drawing.Point(368, 96);
this.vScrollBar7.Name = "vScrollBar7";
this.vScrollBar7.Size = new System.Drawing.Size(24, 354);
this.vScrollBar7.TabIndex = 29;
this.vScrollBar7.Value = 100;
this.vScrollBar7.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar7_Scroll);

// vScrollBar8

this.vScrollBar8.LargeChange = 1;
this.vScrollBar8.Location = new System.Drawing.Point(424, 96);
this.vScrollBar8.Name = "vScrollBar8";
this.vScrollBar8.Size = new System.Drawing.Size(24, 354);
this.vScrollBar8.TabIndex = 30;
this.vScrollBar8.Value = 100;
this.vScrollBar8.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar8_Scroll);

// vScrollBar9

this.vScrollBar9.LargeChange = 1;
this.vScrollBar9.Location = new System.Drawing.Point(488, 96);
this.vScrollBar9.Maximum = 200;
this.vScrollBar9.Minimum = -200;
this.vScrollBar9.Name = "vScrollBar9";
this.vScrollBar9.Size = new System.Drawing.Size(24, 354);
this.vScrollBar9.TabIndex = 31;
this.vScrollBar9.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar9_Scroll);

// label9

this.label9.Location = new System.Drawing.Point(192, 8);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(104, 24);
this.label9.TabIndex = 32;
this.label9.Text = "COLORKEY";

// label10

this.label10.Location = new System.Drawing.Point(832, 8);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(88, 24);
this.label10.TabIndex = 33;
this.label10.Text = "OVERLAY";
// label11
//
// System.Drawing.GraphicsUnit.Point, ((byte)(0)));
// this.label11.Location = new System.Drawing.Point(360, 8);
// this.label11.Name = "label11";
// this.label11.Size = new System.Drawing.Size(80, 24);
// this.label11.TabIndex = 34;
// this.label11.Text = "DESPILL";

// label12
//
// System.Drawing.GraphicsUnit.Point, ((byte)(0)));
// this.label12.Location = new System.Drawing.Point(368, 32);
// this.label12.Name = "label12";
// this.label12.Size = new System.Drawing.Size(40, 24);
// this.label12.TabIndex = 35;
// this.label12.Text = "mix";

// label13
//
// System.Drawing.GraphicsUnit.Point, ((byte)(0)));
// this.label13.Location = new System.Drawing.Point(408, 32);
// this.label13.Name = "label13";
// this.label13.Size = new System.Drawing.Size(64, 24);
// this.label13.TabIndex = 36;
// this.label13.Text = "expand";

// label14
//
// System.Drawing.GraphicsUnit.Point, ((byte)(0)));
// this.label14.Location = new System.Drawing.Point(472, 32);
// this.label14.Name = "label14";
// this.label14.Size = new System.Drawing.Size(72, 24);
// this.label14.TabIndex = 37;
// this.label14.Text = "bright";

// textBox2
//
// System.Drawing.GraphicsUnit.Point, ((byte)(0)));
// this.textBox2.Location = new System.Drawing.Point(360, 56);
// this.textBox2.Name = "textBox2";
// this.textBox2.Size = new System.Drawing.Size(49, 26);
// this.textBox2.TabIndex = 38;
// textBox4
this.textBox4.Location = new System.Drawing.Point(416, 56);
this.textBox4.Name = "textBox4";
this.textBox4.Size = new System.Drawing.Size(49, 26);
this.textBox4.TabIndex = 39;
// textBox5
this.textBox5.Location = new System.Drawing.Point(472, 56);
this.textBox5.Name = "textBox5";
this.textBox5.Size = new System.Drawing.Size(56, 26);
this.textBox5.TabIndex = 40;
// label15
this.label15.Location = new System.Drawing.Point(744, 8);
this.label15.Name = "label15";
this.label15.Size = new System.Drawing.Size(64, 24);
this.label15.TabIndex = 41;
this.label15.Text = "SCALE";
// vScrollBar10
// this.vScrollBar10.LargeChange = 1;
this.vScrollBar10.Location = new System.Drawing.Point(760, 96);
this.vScrollBar10.Minimum = 10;
this.vScrollBar10.Name = "vScrollBar10";
this.vScrollBar10.Size = new System.Drawing.Size(24, 354);
this.vScrollBar10.TabIndex = 42;
this.vScrollBar10.Value = 10;
this.vScrollBar10.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar10_Scroll);
// textBox6
this.textBox6.Location = new System.Drawing.Point(744, 56);
this.textBox6.Name = "textBox6";
this.textBox6.Size = new System.Drawing.Size(56, 26);
this.textBox6.TabIndex = 43;
textBox7
textBox8
vScrollBar11
textBox9
label4
label7
label14
TextBox7
TextBox8
vScrollBar11
TextBox9
label4
label7
label14
System.Drawing.Point(824, 56);
"textBox7"
System.Drawing.Size(49, 26);
44;
"textBox8"
System.Drawing.Point(880, 56);
"textBox8"
System.Drawing.Size(49, 26);
45;
552, 96);
-100;
"vScrollBar11"
System.Drawing.Size(24, 354);
new System.Windows.Forms.ScrollEventHandler(this.vScrollBar11_Scroll);
536, 56);
"textBox9"
System.Drawing.Size(56, 26);
47;
true;
new System.Drawing.Point(544, 32);
32);
"red"
new System.Drawing.Point(536, 56);
new System.Drawing.Point(552, 96);
1;
new System.Drawing.Point(544, 32);
new System.Drawing.Point(552, 96);
this.vScrollBar11.Location = new System.Drawing.Point(552, 96);
this.vScrollBar11.Minimum = -100;
this.vScrollBar11.Name = "vScrollBar11";
this.vScrollBar11.Size = new System.Drawing.Size(24, 354);
this.vScrollBar11.TabPage = 46;
this.vScrollBar11.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar11_Scroll);
// textBox9
// label4
// label7
// label14
// this.label14.AutoSize = true;
// this.label14.Location = new System.Drawing.Point(544, 32);
// this.label14.Name = "label14";
// this.label14.Size = new System.Drawing.Size(32, 20);
// this.label14.TabIndex = 48;
// this.label4.Text = "red";
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(600, 32);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(50, 20);
this.label7.TabIndex = 51;
this.label7.Text = "green";
//
// textBox10
//
this.textBox10.Location = new System.Drawing.Point(600, 56);
this.textBox10.Name = "textBox10";
this.textBox10.Size = new System.Drawing.Size(57, 26);
this.textBox10.TabIndex = 50;
//
// vScrollBar12
//
this.vScrollBar12.LargeChange = 1;
this.vScrollBar12.Location = new System.Drawing.Point(616, 96);
this.vScrollBar12.Minimum = -100;
this.vScrollBar12.Name = "vScrollBar12";
this.vScrollBar12.Size = new System.Drawing.Size(24, 354);
this.vScrollBar12.TabIndex = 49;
this.vScrollBar12.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar12_Scroll);
//
// label16
//
this.label16.AutoSize = true;
this.label16.Location = new System.Drawing.Point(672, 32);
this.label16.Name = "label16";
this.label16.Size = new System.Drawing.Size(39, 20);
this.label16.TabIndex = 54;
this.label16.Text = "blue";
//
// textBox11
//
this.textBox11.Location = new System.Drawing.Point(664, 56);
this.textBox11.Name = "textBox11";
this.textBox11.Size = new System.Drawing.Size(56, 26);
this.textBox11.TabIndex = 53;
//
// vScrollBar13
//
this.vScrollBar13.LargeChange = 1;
this.vScrollBar13.Location = new System.Drawing.Point(680, 96);
this.vScrollBar13.Minimum = -100;
this.vScrollBar13.Name = "vScrollBar13";
this.vScrollBar13.Size = new System.Drawing.Size(24, 354);
this.vScrollBar13.TabIndex = 52;
this.vScrollBar13.Scroll += new System.Windows.Forms.ScrollEventHandler(this.vScrollBar13Scroll);

// checkBox2
this.checkBox2.AutoSize = true;
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.checkBox2.Location = new System.Drawing.Point(600, 8);
this.checkBox2.Name = "checkBox2";
this.checkBox2.Size = new System.Drawing.Size(117, 24);
this.checkBox2.TabIndex = 55;
this.checkBox2.Text = "modify alpha";
this.checkBox2.UseVisualStyleBackColor = true;
this.checkBox2.CheckedChanged += new System.EventHandler(this.checkBox2_CheckedChanged);

// comboBox1
this.comboBox1.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.comboBox1.FormattingEnabled = true;
this.comboBox1.Items.AddRange(new object[] {
"blue",
"green"});
this.comboBox1.Location = new System.Drawing.Point(504, 8);
this.comboBox1.Name = "comboBox1";
this.comboBox1.Size = new System.Drawing.Size(64, 24);
this.comboBox1.TabIndex = 56;
this.comboBox1.SelectedIndexChanged += new System.EventHandler(this.comboBox1_SelectedIndexChanged);

// label17
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.label17.Location = new System.Drawing.Point(456, 8);
this.label17.Name = "label17";
this.label17.Size = new System.Drawing.Size(48, 24);
this.label17.TabIndex = 57;
this.label17.Text = "type:";

// openFileDialog1
this.openFileDialog1.FileName = "openFileDialog1";

// openFileDialog1.FileName = "openFileDialog1";

// openFileDialog1.FileName = "openFileDialog1";
// button3
//
this.button3.Location = new System.Drawing.Point(8, 16);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(168, 88);
this.button3.TabIndex = 58;
this.button3.Text = "Select Background Video";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.button3_Click);

// numericUpDown1
//
this.numericUpDown1.Location = new System.Drawing.Point(40, 144);
this.numericUpDown1.Maximum = new decimal(new int[] {250, 0, 0, 0});
this.numericUpDown1.Minimum = new decimal(new int[] {1, 0, 0, 0});
this.numericUpDown1.Name = "numericUpDown1";
this.numericUpDown1.Size = new System.Drawing.Size(56, 26);
this.numericUpDown1.TabIndex = 60;
this.numericUpDown1.Value = new decimal(new int[] {50, 0, 0, 0});

// label18
//
this.label18.AutoSize = true;
this.label18.Location = new System.Drawing.Point(8, 120);
this.label18.Name = "label18";
this.label18.Size = new System.Drawing.Size(169, 16);
this.label18.TabIndex = 61;
this.label18.Text = "Number of Frames in Loop:";

// label19
//
this.label19.AutoSize = true;
this.label19.Location = new System.Drawing.Point(8, 120);
this.label19.Name = "label19";
this.label19.Size = new System.Drawing.Size(169, 16);
this.label19.TabIndex = 61;
this.label19.Text = "Number of Frames in Loop:";
this.label19.Location = new System.Drawing.Point(720, 32);
this.label19.Name = "label19";
this.label19.Size = new System.Drawing.Size(102, 20);
this.label19.TabIndex = 62;
this.label19.Text = "(Foreground)";

// button2
//
this.button2.Location = new System.Drawing.Point(8, 344);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(168, 36);
this.button2.TabIndex = 1;
this.button2.Text = "Stop FFmpeg";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);

// button4
//
this.button4.Location = new System.Drawing.Point(8, 408);
this.button4.Name = "button4";
this.button4.Size = new System.Drawing.Size(168, 44);
this.button4.TabIndex = 63;
this.button4.Text = "Write FFmpeg command line to file "bluescreen.bat"";
this.button4.UseVisualStyleBackColor = true;
this.button4.Click += new System.EventHandler(this.button4_Click);

// comboBox2
//
this.comboBox2.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
this.comboBox2.FormattingEnabled = true;
this.comboBox2.Items.AddRange(new object[] { "mjpeg 30fps", "yuyv422 5fps" });
this.comboBox2.Location = new System.Drawing.Point(16, 208);
this.comboBox2.Name = "comboBox2";
this.comboBox2.Size = new System.Drawing.Size(136, 24);
this.comboBox2.TabIndex = 64;

// label20
//
this.label20.AutoSize = true;
this.label20.Location = new System.Drawing.Point(16, 246);
this.label20.Name = "label20";
this.label20.Size = new System.Drawing.Size(228, 16);
this.label20.TabIndex = 66;
this.label20.Text = "ffmpeg -i input.mpg -vcodec mjpeg -framerate 30 -f mjpeg -y output.mp4";
this.label20.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
this.label20.Visible = true;
this.label20.Width = 87;
this.label20.Location = new System.Drawing.Point(16, 184);
this.label20.Name = "label20";
this.label20.Size = new System.Drawing.Size(141, 16);
this.label20.TabIndex = 65;
this.label20.Text = "HDMI to USB Adapter:";
//
// checkBox3
//
this.checkBox3.AutoSize = true;
this.checkBox3.Location = new System.Drawing.Point(8, 256);
this.checkBox3.Name = "checkBox3";
this.checkBox3.Size = new System.Drawing.Size(70, 17);
this.checkBox3.TabIndex = 66;
this.checkBox3.Text = "2 x Zoom";
this.checkBox3.UseVisualStyleBackColor = true;
//
// Form1
//
this.AutoScaleBaseSize = new System.Drawing.Size(6F, 13F);
this.ClientSize = new System.Drawing.Size(943, 461);
this.Controls.Add(this.checkBox3);
this.Controls.Add(this.label20);
this.Controls.Add(this.comboBox2);
this.Controls.Add(this.button4);
this.Controls.Add(this.label19);
this.Controls.Add(this.label18);
this.Controls.Add(this.numericUpDown1);
this.Controls.Add(this.button3);
this.Controls.Add(this.label17);
this.Controls.Add(this.comboBox1);
this.Controls.Add(this.checkBox2);
this.Controls.Add(this.label16);
this.Controls.Add(this.textBox11);
this.Controls.Add(this.vScrollBar13);
this.Controls.Add(this.label7);
this.Controls.Add(this.textBox10);
this.Controls.Add(this.vScrollBar12);
this.Controls.Add(this.label4);
this.Controls.Add(this.textBox9);
this.Controls.Add(this.vScrollBar11);
this.Controls.Add(this.textBox8);
this.Controls.Add(this.textBox7);
this.Controls.Add(this.vScrollBar10);
this.Controls.Add(this.label15);
this.Controls.Add(this.textBox4);
this.Controls.Add(this.label14);
this.Controls.Add(this.label13);
this.Controls.Add(this.textBox9);
this.Controls.Add(this.vScrollBar12);
this.Controls.Add(this.label7);
this.Controls.Add(this.textBox10);
this.Controls.Add(this.vScrollBar13);
this.Controls.Add(this.label15);
this.Controls.Add(this.textBox4);
this.Controls.Add(this.label14);
this.Controls.Add(this.label13);
this.Controls.Add(this.label14);
this.Controls.Add(this.label13);
this.Controls.Add(this.label12);
this.Controls.Add(this.label11);
this.Controls.Add(this.label10);
this.Controls.Add(this.label9);
this.Controls.Add(this.vScrollBar9);
this.Controls.Add(this.vScrollBar8);
this.Controls.Add(this.vScrollBar7);
this.Controls.Add(this.textBox3);
this.Controls.Add(this.label8);
this.Controls.Add(this.vScrollBar4);
this.Controls.Add(this.vScrollBar6);
this.Controls.Add(this.label6);
this.Controls.Add(this.label5);
this.Controls.Add(this.vScrollBar5);
this.Controls.Add(this.checkBox1);
this.Controls.Add(this.textBox1);
this.Controls.Add(this.label3);
this.Controls.Add(this.label2);
this.Controls.Add(this.vScrollBar3);
this.Controls.Add(this.vScrollBar2);
this.Controls.Add(this.button2);
this.Controls.Add(this.button1);
this.Location = new System.Drawing.Point(0, 540);
this.Name = "Form1";
this.Text = "FFbluescreen";
this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form1_FormClosing);
thisShown += new System.EventHandler(this.Form1_Shown);
(System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();
}
#endregion
private System.Windows.Forms.Button button1;
private System.Windows.Forms.VScrollBar vScrollBar1;
private System.Windows.Forms.VScrollBar vScrollBar2;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.VScrollBar vScrollBar3;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.CheckBox checkBox1;
private System.Windows.Forms.VScrollBar vScrollBar5;
As you can see in the source code, I didn't find a way to move the console window to another position (to make sure that it doesn't overlap the video...
output and the GUI of this program).
Moving the video output is no problem, there are even two ways how to do it. You can use the window_x and window_y options in the FFmpeg command line, or you can use this code:

```csharp
Process[] allProcesses = Process.GetProcessesByName("ffmpeg"); // this is unnecessary, if you already know the
IntPtr ffmpegHandle = allProcesses.First();                      // process handle because you have started the process
WinAPI.MoveWindow(ffmpegHandle.MainWindowHandle, 960, 0, 960, 540, true);   // move the video output window
public class WinAPI
{
  [DllImport("user32.dll")] public static extern bool MoveWindow(IntPtr hWnd, int X, int Y, int nWidth, int nHeight, bool bRepaint);
}
```

If you know how the programmatically move the console window, please let me know.

Note: If you want to find out to which process a window belongs, you can use the ProcessExplorer:
https://docs.microsoft.com/de-de/sysinternals/downloads/process-explorer

According to ProcessExplorer, both FFmpeg windows (console and video output) belong to the "ffmpeg" process (it's not the "conhost" process!). There is only one "ffmpeg" process running. I don't know how to get the handle of the console window.

But there is an easy workaround for moving the console window:
Open a console window and move it to the right of the desktop, then right click in the title line, then choose "layout" and then set the "Window position" and uncheck the box "Let system position window". This adjustment is only required one time and now the console window will always appear at the same position.

Note: I did try to scale the foreground video dynamically before the colorkey filter, but that didn't work. In general, Most FFmpeg filters don't support changing the size of a video stream while it is running. In some cases it works (for example if no other filters are between "scale" and "overlay"), but in many other cases it doesn't work.

My results of using this bluescreening program:
By far the most important thing is to set the colorkey parameters correctly: color and similarity.
The despill filter is not as important and the possible improvements are quite small. The "modify alpha" option is useless and should be deactivated. In many cases it totally destroys the background image.
Recommended settings for despill filter: type = blue, blue = -1, alpha = false, all other options = 0. The most important parameter is "brightness".
2.57 Datascope

The "datascope" filter can be used to measure the RGB color components of the bluescreen. In this example the input video comes from the HDMI to USB converter:

```
ffmpeg -f dshow -video_size 1920x1080 -framerate 5 -pixel_format yuyv422 -i video="USB Video" -lavfi format=rgb24,scale=64:20,datascope=mode=color2 -f sdl -
```

Note: The default size seems to be 1280x720 pixels.

This is an analyzer for one pixel in the center of the field of view. The input video comes from the HDMI to USB converter:

```
ffmpeg -f dshow -video_size 1920x1080 -framerate 5 -pixel_format yuyv422 -i video="USB Video" -lavfi format=rgb24,crop=1:1,datascope=s=20x36:mode=color2,scale=iw*10:ih*10:flags=gauss -f sdl -
```

Note: One block consisting of three hex digits has the size 20x36 pixels, which is in this example enlarged by a factor 10 to 200x360 pixels.
I wanted to create a special effect, adding jitter to the video lines. Like in this video at 3:40  [https://www.youtube.com/watch?v=A9D_PlfpBH4](https://www.youtube.com/watch?v=A9D_PlfpBH4)

This is the first version:
```
ffmpeg -f lavfi -i testsrc2=size=vga -vf format=gray,geq=lum='lum(X-5+10*random(0),Y)' -t 3 -y out.mp4
```

The problem is that this is a pixel-wise jitter. Each pixel gets it's own jitter value. That's not what I want. I want that all pixels in the same line get the same random jitter value. This should be possible by setting a seed value for the random generator. The seed value must be a function of N (frame number) and Y (line number). This is my next (unsuccessful) test:
```
ffmpeg -f lavfi -i testsrc2=size=vga -vf format=gray,geq=lum='st(0,mod(0.001*(N+Y),1));lum(X-5+10*random(0),Y)' -t 3 -y out.mp4
```

The random() function uses the variable 0 to save it's seed value. But it seems that it's impossible to write a seed value for the random function. I don't understand why it doesn't work.

Finally I replaced the random() function by a selfmade workaround. It's not a perfect random function, but good enough for this purpose:
```
ffmpeg -f lavfi -i testsrc2=size=vga -vf format=gray,geq=lum='st(0,mod(PI*(N+Y*(Y-N+PI)),1));lum(X-5+10*ld(0),Y)' -t 3 -y out.mp4
```
2.59  Vertical jitter effect

This is a simulation of vertical jitter, like in a defect film projector.

```
set "IN=test.mp4"            :: Input video
set "J=0.05"                 :: Maximum amplitude of vertical jitter as fraction of image height
set "F=15"                   :: Maximum speed of vertical jitter in pixels from one frame to the next frame
set "OUT=out.mp4"            :: Output video

ffmpeg -i %IN% -lavfi
crop=w=(1-%J%)*iw:h=(1-%J%)*ih:y='st(0,clip(l0+2*F*(random(l)-0.5),0,%J*ih))' -y %OUT%
```

This is an interesting example because it shows that a variable can be saved with st(0, expr) and that it keeps its value from one frame to the next. It's currently not possible to share variables between expressions.
2.60    **Deblock filter**

This filter removes unwanted blocking artefacts from low-quality input images or videos.

2.61    **Gradfun filter**

This filter removes unwanted banding artefacts that appear in backgrounds with a brightness gradient, especially in the sky towards the horizon.

```plaintext
set "IN=MVI_2562.mov" :: Input video
set "OUT=output.mp4" :: Output video
ffmpeg -i %IN% -vf gradfun=3.5:8 -y %OUT%
```

The first parameter is the strength, this is the maximum amount the filter will change any one pixel. Allowed values are from 0.51 to 64, the default value is 1.2.

The second parameter is the radius, which defines the neighborhood to fit the gradient to. Accepted values are from 8 to 32, the default is 16.

Don't use this filter before lossy compression.

2.62    **Dilation filter**

This filter replaces each pixel by the brightest pixel in the 3x3 neighborhood. It's very useful if you have fisheye images of the night sky (taken with Canon 6D, height 3648 pixels) and want to scale them down to height 1200 pixels (for projection in the planetarium). Scaling down would remove the fainter stars, because each pixel in the resulting image would be the average of 3x3 pixels in the original image. You can avoid this by using the dilation filter prior to scaling down.
2.63 Correct the radial distortion of (fisheye-) lenses

This can be done with the "lenscorrection" filter, which has the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cx</td>
<td>Relative x coordinate of the center of distortion, in range [0...1], default is 0.5</td>
</tr>
<tr>
<td>cy</td>
<td>Relative y coordinate of the center of distortion, in range [0...1], default is 0.5</td>
</tr>
<tr>
<td>k1</td>
<td>Coefficient of the quadratic correction term, in range [-1...1]. 0 means no correction, default is 0.</td>
</tr>
<tr>
<td>k2</td>
<td>Coefficient of the ^4 correction term, in range [-1...1]. 0 means no correction, default is 0.</td>
</tr>
</tbody>
</table>

The formula that generates the correction is:

\[ r_{src} = r_{tgt} \times (1 + k1 \times \left(\frac{r_{tgt}}{r_0}\right)^2 + k2 \times \left(\frac{r_{tgt}}{r_0}\right)^4) \]

where \( r_0 \) is half of the image diagonal and \( r_{src} \) and \( r_{tgt} \) are the distances from the focal point in the source and target images, respectively.

For fisheye images, it's a little bit more complicated because the coefficients \( k1, k2 \) aren't given for half of the image diagonal.

Let \( w \) and \( h \) be the dimensions of the rectangular input image.

The square of the ratio of the diagonal of the rectangular image to the diagonal of the circular fisheye image is:

\[ \text{ratio} = \frac{w^2 + h^2}{h^2} \]

Let \( c1 \) and \( c2 \) be the given coefficients for the fisheye lens. Then the coefficients \( k1 \) and \( k2 \) can be calculated as follows:

\[ k1 = c1 \times \text{ratio} \]

\[ k2 = c2 \times \text{ratio}^2 \]

See also [http://www.paulbourke.net/dome/fisheycorrect/](http://www.paulbourke.net/dome/fisheycorrect/)
This is an example for the Entaniya M12 280° lens, from Paul Bourke’s website:

\[ y = 0.5229 \times -0.043 \times^2 + 0.0253 \times^3 - 0.0109 \times^4 \]

\( y \) is the radial coordinate in the image plane in the [0...1] range, and \( x \) is the field angle in radians. The maximum \( x \) value is \( \text{FOV} \times \frac{\pi}{360°} \).

Let's assume we have a perfect fisheye image and we want to simulate how this image would look like, if it was taken through the not-so-perfect fisheye lens.

```plaintext
set "IN=1200.png"             :: Input image (fisheye test image)
set "S=1200"                  :: Size of input image
set "FOV=280"                 :: Field of view of fisheye lens in degrees
set "A=0.5229"                :: First order coefficient, for a perfectly linear fisheye lens this is \( \frac{360}{\text{FOV}/\pi} \)
set "B=-0.043"                :: Second order coefficient
set "C=0.0253"                :: Third order coefficient
set "D=-0.0109"               :: Fourth order coefficient

rem Step 1:
rem Apply the fisheye distortion Example for Entaniya M12 280° lens
rem \( y = A \times x + B \times x^2 + C \times x^3 + D \times x^4 \)
rem where \( x \) is in the [0...\text{FOV}/\pi/360] range

rem Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,geq='st(1,hypot(X-%S%/2,Y-%S%/2)/%S%*PI*%FOV/180);st(2,atan2(X-%S%/2,Y-%S%/2));st(3,%A%*ld(1)+%B%*pow(ld(1),2)+%C%*pow(ld(1),3)+%D%*pow(ld(1),4));%S%/2+0.5*ld(3)*%S%/2*sin(ld(2))' -frames 1 -y xmap.pgm

rem Create the ymap file
ffmpeg -i xmap.pgm -vf transpose -y ymap.pgm

rem Apply the remap filter to the video
ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -y out1.png

pause
```

Note: For all rotational-symmetric problems the ymap file can simply be generated by transposing the xmap file.
The inverse problem is much more complicated. Given is an image that was taken through a not-so-perfect fisheye lens, and we want to transform this image into an undistorted fisheye image, with other words: We want to linearize it.

In this case we need the inverse of the above $4^{th}$ degree function, which is very complicated to derive. The trick is to use the root(expr, max) function. But please be warned that this is an extremely slow solution, because it requires to find the root of an expression for each pixel.

```plaintext
set "IN=1200.png" :: Input image (fisheye test image)
set "S=1200" :: Size of input image
set "FOV=280" :: Field of view of fisheye lens in degrees
set "A=0.5229" :: First order coefficient, for a perfectly linear fisheye lens this is (360/FOV/PI)
set "B=-0.043" :: Second order coefficient
set "C=0.0253" :: Third order coefficient
set "D=-0.0109" :: Fourth order coefficient

rem Step 2:
rem Apply the inverse function to out1.png, and then the result should be same as the original image

rem Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,geq='st(1,hypot(X-%S%/2,Y-%S%/2)/%S
%*2);st(2,atan2(X-%S%/2,Y-%S%/2));st(3,root(-1d(1)+%A%*ld(0)+%B%*pow(ld(0),2)+%C%*pow(ld(0),3)+%D%*pow(ld(0),4),%FOV
%/360*PI));%S%/2+0.5+1d(3)*%S%/PI/%FOV%*180*sin(ld(2))' -frames 1 -y xmap.pgm

rem Create the ymap file
ffmpeg -i xmap.pgm -vf transpose -y ymap.pgm

rem Apply the remap filter to the video
ffmpeg -i out1.png -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -y out2.png

pause
```
2.64 V360 filter for rotation of equirectangular 360° videos

This video filter converts equirectangular 360° panoramic videos between various formats, and it can also rotate them.

The default rotation order is yaw --> pitch --> roll, but can be changed by setting the "rorder" parameter. Positive yaw moves the line of sight towards the right, positive pitch moves the line of sight up, positive roll rotates the image clockwise (or rotates the observer's head counter-clockwise).

```
set "IN=test1.mp4" :: Input video

ffmpeg -ss 10 -i %IN% -vf v360=yaw=0:output=e -frames 1 -y t_original.jpg
ffmpeg -ss 10 -i %IN% -vf v360=yaw=90:output=e -frames 1 -y t_yaw90.jpg
ffmpeg -ss 10 -i %IN% -vf v360=pitch=90:output=e -frames 1 -y t_pitch90.jpg
ffmpeg -ss 10 -i %IN% -vf v360=roll=90:output=e -frames 1 -y t_roll90.jpg
ffmpeg -ss 10 -i %IN% -vf v360=yaw=90:pitch=90:output=e -frames 1 -y t_yaw90_pitch90.jpg
ffmpeg -ss 10 -i %IN% -vf v360=yaw=90:roll=90:output=e -frames 1 -y t_yaw90_roll90.jpg
ffmpeg -ss 10 -i %IN% -vf v360=pitch=90:roll=90:output=e -frames 1 -y t_pitch90_roll90.jpg
```
Parameters of the v360 filter:

<table>
<thead>
<tr>
<th>input, output</th>
<th>e, equirect</th>
<th>Equirectangular projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>c3x2, c6x1, c1x6</td>
<td>Three different cubemap projection</td>
<td></td>
</tr>
<tr>
<td>eac</td>
<td>Equi-angular cubemap</td>
<td></td>
</tr>
<tr>
<td>flat, gnomonic, rectilinear</td>
<td>Regular video projection</td>
<td></td>
</tr>
<tr>
<td>dfisheye</td>
<td>Dual fisheye projection</td>
<td></td>
</tr>
<tr>
<td>barrel, fb, barrelsplit</td>
<td>Facebook's 360 projection</td>
<td></td>
</tr>
<tr>
<td>sg</td>
<td>Stereographic projection</td>
<td></td>
</tr>
<tr>
<td>mercator</td>
<td>Mercator projection</td>
<td></td>
</tr>
<tr>
<td>ball</td>
<td>Ball projection, this means the 360° content of the input video is shown as a reflection on a mirror sphere. Similar to single-fisheye with 360° field of view, but has a different mapping function: With ball projection all points with 90° distance from the center point are mapped to the circle with 70.7% radius, however with 360° single fisheye projection they are mapped to the circle with 50% radius. In both cases the point with 180° distance from the center point is mapped to the edge of the circle.</td>
<td></td>
</tr>
<tr>
<td>hammer</td>
<td>Hammer-Aitoff map projection</td>
<td></td>
</tr>
<tr>
<td>sinusoidal</td>
<td>Sinusoidal map projection projection</td>
<td></td>
</tr>
<tr>
<td>fisheye</td>
<td>Single fisheye projection</td>
<td></td>
</tr>
<tr>
<td>pannini</td>
<td>Pannini projection (output only)</td>
<td></td>
</tr>
<tr>
<td>cylindrical</td>
<td>Cylindrical projection</td>
<td></td>
</tr>
<tr>
<td>perspective</td>
<td>Perspective projection, this is like watching a sphere from big distance (output only). The output image is mirrored and the center of the input image is not in the center of the output image. Workaround for correct output: v360=e:perspective:pitch=90:v_flip=1</td>
<td></td>
</tr>
<tr>
<td>tetrahedron</td>
<td>Tetrahedron projection</td>
<td></td>
</tr>
<tr>
<td>octahedron</td>
<td>Octahedron projection</td>
<td></td>
</tr>
<tr>
<td>tsp</td>
<td>Truncated square pyramid projection</td>
<td></td>
</tr>
<tr>
<td>he, hequirect</td>
<td>Half equirectangular projection</td>
<td></td>
</tr>
<tr>
<td>equisolid</td>
<td>Equisolid projection</td>
<td></td>
</tr>
<tr>
<td>og</td>
<td>Orthographic projection</td>
<td></td>
</tr>
<tr>
<td>interp</td>
<td>near, nearest</td>
<td>Nearest neighbour interpolation</td>
</tr>
<tr>
<td>------------</td>
<td>---------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>line, linear</td>
<td>Bilinear interpolation, this is the default</td>
<td></td>
</tr>
<tr>
<td>cube, cubic</td>
<td>Bicubic interpolation</td>
<td></td>
</tr>
<tr>
<td>lanc, lanczos</td>
<td>Lanczos interpolation</td>
<td></td>
</tr>
<tr>
<td>sp16, spline16</td>
<td>Spline16 interpolation</td>
<td></td>
</tr>
<tr>
<td>gauss, gaussian</td>
<td>Gaussian interpolation</td>
<td></td>
</tr>
<tr>
<td>mitchell</td>
<td>Mitchell interpolation</td>
<td></td>
</tr>
</tbody>
</table>

| w, h | in pixels | Width and height of the output video, default size depends on output format |
| yaw, pitch, roll | in degrees | Rotation angles |
| rorder | 'ypr', 'ypr', 'pyr', 'pry', 'ryp', 'rpy' | Set the rotation order, default is 'ypr' |
| h_flip, v_flip | 0, 1 | Flip the output horizontally or vertically |
| d_flip | | Flip the output back/forward |
| ih_flip, iv_flip | | Flip the input horizontally or vertically |
| in_trans | | Transpose the input |
| out_trans | | Transpose the output |
| h_fov, v_fov, d_fov | in degrees | Set the horizontal, vertical or diagonal field of view for output |
| ih_fov, iv_fov, id_fov | in degrees | Set the horizontal, vertical or diagonal field of view for input |

Undocumented feature of the v360 filter: The top left pixel of the input video is mapped to all those pixels in the output video, which get no input data. If you want to give the unused area a specific color, you can just fill the top left pixel of the input video with this color:

```
-vf drawbox=w=1:h=1:color=green,v360=...
```

Note: Even if the input pixel format is rgb24, the output format is gbrp which is a planar pixel format.
2.65 Equirectangular images of the night sky

Equirectangular images of the night sky can be found here:

http://paulbourke.net/miscellaneous/astronomy/
http://paulbourke.net/dome/stellariumsphere/
http://paulbourke.net/dome/celestiasphere/
https://svs.gsfc.nasa.gov/vis/a000000/a003500/a003572/
https://sci.esa.int/web/gaia/-/60196-gaia-s-sky-in-colour-equirectangular-projection
2.66 Remap a fisheye video to an equirectangular video

In this example the xmap and ymap files for the remap filter are created by FFmpeg (no C# code required). The size of the equirectangular video is defined by the user and can be different from 2:1.

```
set "IN=110_0001.mp4" :: Input video
set "SQ=2880" :: Size of square input video
set "SR=1440" :: Radius that is actually used from the source video, must be SQ/2 or smaller
set "PW=1920" :: Width of panorama video
set "PH=550" :: Height of panorama video
set "OUT=out.mp4" :: Output video

rem Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='%SQ%/2-Y*%SR%/%PH%*sin(X*2*PI/%PW%)' -frames 1 -y xmap.pgm
rem Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='%SQ%/2-Y*%SR%/%PH%*cos(X*2*PI/%PW%)' -frames 1 -y ymap.pgm
rem Apply the remap filter to the video
ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -c:v mpeg4 -q:v 2 -y %OUT%
pause
```

If the fisheye lens has more than 180° field of view, but you want only 180° visible in the panorama, set the SR variable to a value smaller than SQ/2.

A lot of informations about fisheye projections can be found on Paul Bourke's website: [www.paulbourke.net/dome/](http://www.paulbourke.net/dome/)

More informations about the remap filter can be found here: [https://trac.ffmpeg.org/wiki/RemapFilter](https://trac.ffmpeg.org/wiki/RemapFilter)

The color of unmapped pixels can be specified with the "fill" option, the default color is black.
Fisheye input (from Kodak Pixpro SP360 camera): 

Panorama output: 

The VLC player won't recognize the output video as a spherical equirectangular video, because some special metadata is missing. This metadata can't be inserted with FFmpeg, but it can be done with the "Spatial Media Metadata Injector":

https://github.com/google/spatial-media/releases/tag/v2.1
In this example the fisheye's field of view can be set to any value up to 360°, and the width/height ratio of the equirectangular output video is always 2:1. The lower part is filled with black if the fisheye has less than 360° field of view.

```plaintext
set "IN=IMG_077.jpg" :: Fisheye input image or video, must be square
set "SQ=3648" :: Size of square fisheye input image
set "FOV=220" :: Fisheye field of view in degrees
set "Q=2" :: Size divider for output image, use 1 for best quality,
  :: or a bigger value for faster computing
set /a "H=%SQ%/%Q%" :: Height of equirectangular image
set /a "W=2*%H%" :: Width of equirectangular image is always twice the height
set /a "A=%H%*%FOV%/360" :: Height of equirectangular image that is actually filled with data,
  :: the lower part of the output image remains black
set "OUT=out.jpg" :: Equirectangular output image or video

rem  Create the xmap file for remapping from fisheye to equirectangular
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray16le,^
  geq='%SQ%/2*(1-Y/%A%*sin(X*2*PI/%W%))' -frames 1 -y xmap1.pgm

rem  Create the ymap file for remapping from fisheye to equirectangular
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray16le,^
  geq='%SQ%/2*(1-Y/%A%*cos(X*2*PI/%W%))' -frames 1 -y ymap1.pgm

rem  Remap from fisheye to equirectangular
ffmpeg -i %IN% -i xmap1.pgm -i ymap1.pgm -filter_complex "format=pix_fmts=rgb24,remap" -y %OUT%
pause
```
For a square 180° single-fisheye video the conversion to an equirectangular video can also be done with the V360 filter. The second hemisphere is filled with a user-defined color. This example is obsolete, please use the next example.

```
set "IN=in.mp4"               :: Fisheye input video (square, camera pointing upwards)
set "OUT=out.mp4"             :: Equirectangular output video

ffmpeg -i %IN% -lavfi "pad=w=2*iw:color=darkgreen,v360=input=dfisheye:output=e:pitch=90" -y %OUT%
pause
```

Square single-fisheye images or videos with any field of view can be converted to equirectangular images or videos:

```
set "IN=1200.png"             :: Input image or video
set "FOV=180"                 :: Input field of view in degrees
set "C=green"                 :: Color for filling unused area
set "OUT=out.png"             :: Equirectangular output image or video

ffmpeg -i %IN% -vf drawbox=w=1:h=1:color=%C%,v360=input=fisheye:id_fov=%FOV%:output=equirect:pitch=-90 -y %OUT%
pause
```

If required, the lower part of the equirectangular output can be cut off with the crop filter.
2.67 Remap an equirectangular video to a fisheye video

The field of view can be set between 1 and 360 degrees. The sky is in the center of the fisheye video, and the ground is at the circular edge.

The input video must have 2:1 width/height ratio.

set "IN=test1.mp4" :: Input video
set "H=960" :: Height of equirectangular input video
set "S=1080" :: Size of square fisheye output video
set "FOV=220" :: Set the field of view in degrees
set "OUT=fish.mp4" :: Output video

rem Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^-
geq='%H%*(0.9999+atan2(X-%S%/2,Y-%S%/2)/PI)' -frames 1 -y xmap.pgm

rem Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^-
geq='%H%/360*%FOV%*(hypot((2*X/%S%)-1,(2*Y/%S%)-1))' -frames 1 -y ymap.pgm

rem Apply the remap filter to the video
ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -q:v 2 -y %OUT%

The same thing can also be done with the v360 filter:

set "IN=equirectangular.png" :: Input image or video
set "FOV=220" :: Output field of view in degrees
set "OUT=fish.png" :: Output image or video

ffmpeg -i %IN% -vf v360=input=equirect:output=fisheye:h_fov=%FOV%:v_fov=%FOV%:pitch=90 -y %OUT%

pause
2.68    Remap an equirectangular video to a "Little planet" video

Fisheye projection is used. The ground is in the center of the video, and the sky is at the circular edge. The input video must have 2:1 width/height ratio.

```plaintext
set "IN=test3.mp4" :: Equirectangular input video
set "H=960" :: Height of input video (width = 2 * height)
set "S=1080" :: Size of square little planet output video
set "OUT=out.mp4" :: Output video

rem  Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^ 
   geq='%H%*(0.9999+atan2(Y-%S%/2,X-%S%/2)/PI)' -frames 1 -y xmap.pgm

rem  Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^ 
   geq='%H%*(1-hypot((2*X/%S%)-1,(2*Y/%S%)-1))' -frames 1 -y ymap.pgm

rem  Apply the remap filter to the video
ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap=fill=green" -q:v 2 -y %OUT%
```

The values in the xmap and ymap files can't be negative. If a value is larger than the size of the input image, this pixel is painted with the color that's specified by the "fill" option.
If you want the sky in the center and the ground at the circular edge, use these remap functions instead:

```plaintext
ffmpeg -f lavfi -i nullsrc=size=S%xS% -vf format=pix_fmts=gray16le,^ 
geq='H%*(0.9999*atan2(Y-%S%/2,X-%S%/2)/PI)' -frames 1 -y xmap.pgm

ffmpeg -f lavfi -i nullsrc=size=S%xS% -vf format=pix_fmts=gray16le,^ 
geq='H%*(hypot((2*X/%S%)-1,(2*Y/%S%)-1))' -frames 1 -y ymap.pgm
```

The same thing can also be done with the v360 filter:

```plaintext
set "IN=test1.png"            :: Input image or video
set "FOV=360"                 :: Output field of view in degrees
set "OUT=littleplanet.png"    :: Output image or video

ffmpeg -i %IN% -vf v360=input=equirect:output=fisheye:h_fov=%FOV%:v_fov=%FOV%:pitch=-90 -y %OUT%
```

pause
2.69  Remap an equirectangular video to a "Mirror sphere" video

Similar to "Little planet", but using a different projection. The 360° world is shown as a reflection on a mirror sphere. The ground is in the center of the video, and the sky is at the circular edge. The input video must have 2:1 width/height ratio.

<table>
<thead>
<tr>
<th>Set</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN=equirectangular_test.png</td>
<td>Equirectangular input video</td>
</tr>
<tr>
<td>H=1200</td>
<td>Height of input video (width = 2 * height)</td>
</tr>
<tr>
<td>S=900</td>
<td>Size of square mirror sphere output video</td>
</tr>
<tr>
<td>OUT=mirror.png</td>
<td>Output video</td>
</tr>
</tbody>
</table>

```bash
rem  Create the xmap file
ffmpeg -flavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^
geq='%H%*(0.9999+atan2(Y-%S%/2,X-%S%/2)/PI)' -frames 1 -y xmap.pgm

rem  Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^
geq='%H%*(1-2/PI*asin(hypot((2*X/%S%)-1,(2*Y/%S%)-1))))' -frames 1 -y ymap.pgm

rem  Apply the remap filter to the video
ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -q:v 2 -y %OUT%
```

pause
If you want the sky in the center and the ground at the circular edge, use these remap functions instead:

```bash
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^
geq='\%H%*(0.9999*atan2(Y-%S%/2,X-%S%/2)/PI)' -frames 1 -y xmap.pgm
```

```bash
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^
geq='\%H%*(2/PI*asin(hypot((2*X/%S%)-1,(2*Y/%S%)-1)))' -frames 1 -y ymap.pgm
```

The same thing can also be done with the "ball" output format of the v360 filter:

```bash
set "IN=test1.png"                 :: Equirectangular input image or video
set "OUT=mirror.png"               :: Output image or video
ffmpeg -i %IN% -lavfi "v360=input=e:output=ball:pitch=90" -q:v 2 -y mirror.png
```

Pitch=90 is for the sky in the center, pitch=-90 is for the ground in the center.
2.70   Shifting the viewing direction in a fisheye image or video

When you want to create a timelapse of many fisheye images, it may happen that one of the images isn't aligned correctly because the viewing direction of the camera was off. With normal (non-fisheye) images that isn't a big problem, because you can simply re-align the image by shifting it in x and y directions. However for fisheye images things are much more complicated. The required procedure is as follows:

1. Remap the fisheye image to an equirectangular 360° image. The lower part of the image remains black.
2. Apply two rotations to this equirectangular image.
3. Remap the equirectangular image back to a fisheye image.

```
set "IN=IMG_077.jpg"        :: Input image or video
set "S=3648"                :: Size of square fisheye input image
set "FOV=180"               :: Fisheye field of view in degrees
set "X=15"                  :: Rotation angle around X axis
set "Y=0"                   :: Rotation angle around Y axis
set "Q=5"                   :: Size divider for the intermediate equirectangular image,
                             ::  use 1 for best quality, or a bigger value for faster computing
set /a "H=%S%/%Q%"          :: Height of equirectangular image
set /a "W=2*%H%"            :: Width of equirectangular image is always twice the height
set /a "A=%H%*%FOV%/360"    :: Height of equirectangular image that is actually filled with data, the rest remains black
set "OUT=out.jpg"           :: Output image or video

rem  Create the xmap file for remapping from fisheye to equirectangular
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray16le,^
  geq='%S%/2*(1-Y/%A%*sin(X*2*PI/%W%))' -frames 1 -y xmap1.pgm

rem  Create the ymap file for remapping from fisheye to equirectangular
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray16le,^
  geq='%S%/2*(1-Y/%A%*cos(X*2*PI/%W%))' -frames 1 -y ymap1.pgm
```
rem Create the xmap file for remapping from equirectangular to fisheye

ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^ 
geq='%H%*(0.9999+atan2(X-%S%/2,Y-%S%/2)/PI)' -frames 1 -y xmap2.pgm

rem Create the ymap file for remapping from equirectangular to fisheye

ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^ 
geq='%H%/360*%FOV%*(hypot((2*X/%S%)-1,(2*Y/%S%)-1))' -frames 1 -y ymap2.pgm

rem Remap from fisheye to equirectangular, apply the rotations, then remap back to fisheye

ffmpeg -i %IN% -i xmap1.pgm -i ymap1.pgm -i xmap2.pgm -i ymap2.pgm -filter_complex "format=pix_fmts=rgb24,remap,v360=pitch=%Y%:roll=%X%:output=e[5];[5][3][4]remap" -y %OUT%

pause

The same thing can also be done with the v360 filter. In this example the top left pixel of the input image or video is set to a specific color with the "drawbox" filter. This color is used for all those pixels in the output file, that aren't mapped to a pixel in the input file. Please note that this is an undocumented feature of the v360 filter and it's not guaranteed that it works in all cases.

set "IN=1200.png" :: Input image or video
set "FOV=180" :: Field of view in degrees
set "PITCH=0" :: Rotation angle around X axis
set "YAW=30" :: Rotation angle around Y axis
set "C=green" :: Color for filling unused area
set "OUT=out.png" :: Output image or video

ffmpeg -i %IN% -vf drawbox=w=1:h=1:color=%C%,v360=input=fisheye:ih_fov=%FOV%:iv_fov=%FOV%:output=fisheye:h_fov=%FOV%:v_fov=%FOV%:yaw=%YAW%:pitch=%PITCH% -y %OUT%

pause

Note for "drawbox" filter: This filter doesn't support RGB formats!
The v360 filter does have the "alpha_mask" option. If this option is set, all unused pixels in the output file are set to maximum transparency, so that the overlay filter can be used for filling this area with a color. This example does exactly the same thing as the previous example. Decide yourself which one is easier or faster:

```
set "IN=1200.png"           :: Input image or video
set "FOV=180"               :: Field of view in degrees
set "PITCH=0"               :: Rotation angle around X axis
set "YAW=30"                :: Rotation angle around Y axis
set "C=green"               :: Color for filling unused area
set "OUT=out.png"           :: Output image or video

ffmpeg -i %IN% -f lavfi -i color=%C%:s=1200x1200 -filter_complex v360=input=fisheye:ih_fov=%FOV%:iv_fov=%FOV% %:output=fisheye:h_fov=%FOV%:v_fov=%FOV%:yaw=%YAW%:pitch=%PITCH%:alpha_mask=1[a],1[a] overlay -frames 1 -y %OUT%
pause
```

Note: If the input is a video, remove the -frames 1 option.

See also [www.paulbourke.net/dome/fishtilt/](http://www.paulbourke.net/dome/fishtilt/)
2.71 Stitching together double-fisheye videos

The result is an equirectangular panorama video.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN=double_fisheye.jpg</td>
<td>Input video or picture</td>
</tr>
<tr>
<td>X1=198</td>
<td>X coordinate of center of left fisheye image</td>
</tr>
<tr>
<td>Y1=210</td>
<td>Y coordinate of center of left fisheye image</td>
</tr>
<tr>
<td>X2=595</td>
<td>X coordinate of center of right fisheye image</td>
</tr>
<tr>
<td>Y2=210</td>
<td>Y coordinate of center of right fisheye image</td>
</tr>
<tr>
<td>SR=192</td>
<td>Radius that is actually used from the source video</td>
</tr>
<tr>
<td>PW=1920</td>
<td>Width of panorama video</td>
</tr>
<tr>
<td>PH=960</td>
<td>Height of panorama video</td>
</tr>
<tr>
<td>OUT=out.jpg</td>
<td>Output video or picture</td>
</tr>
</tbody>
</table>

rem Create the xmap file

```bash
grep -f lavfi -i nullsrc=size=%PW%\x%PH% -vf format=pix_fmts=gray16le,geq='if(lt(Y,%PH%/2),%X1%-Y*2*%SR%/%PH%*sin(X*2*PI/%PW%),%X2%+(%PH%-Y)*2*%SR%/%PH%*sin(X*2*PI/%PW%))' -frames 1 -y xmap.pgm
```

rem Create the ymap file

```bash
grep -f lavfi -i nullsrc=size=%PW%\x%PH% -vf format=pix_fmts=gray16le,geq='if(lt(Y,%PH%/2),%Y1%-Y*2*%SR%/%PH%*cos(X*2*PI/%PW%),%Y2%-(%PH%-Y)*2*%SR%/%PH%*cos(X*2*PI/%PW%))' -frames 1 -y ymap.pgm
```

rem Apply the remap filter to the video

```bash
grep -f lavfi -i IN -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -q:v 2 -y OUT
```

The parameters X1, Y1, X2, Y2 and SR must be carefully adjusted (by try and error) to get a good stitching result. They depend on the size of the source video or picture. Use these values as a starting point: X1=width/4, Y1=height/2, X2=width*3/4, Y2=height/2, SR=height/2. The following table shows how the parameters affect the stitching.

Note: The same thing can also be done with the V360 filter, see the next chapter.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Result when decreasing the parameter</th>
<th>Result when increasing the parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>+---------------------------------+</td>
<td>upper half from left fisheye</td>
</tr>
<tr>
<td></td>
<td></td>
<td>← ↑ → ↓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lower half from right fisheye</td>
</tr>
<tr>
<td>Y1</td>
<td>+---------------------------------+</td>
<td>upper half from left fisheye</td>
</tr>
<tr>
<td></td>
<td></td>
<td>↑ → ↓ ← ↑</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lower half from right fisheye</td>
</tr>
<tr>
<td>X2</td>
<td>+---------------------------------+</td>
<td>upper half from left fisheye</td>
</tr>
<tr>
<td></td>
<td></td>
<td>← ↑ ← ↓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lower half from right fisheye</td>
</tr>
<tr>
<td>Y2</td>
<td>+---------------------------------+</td>
<td>upper half from left fisheye</td>
</tr>
<tr>
<td></td>
<td></td>
<td>↑ → ↓ ↑</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lower half from right fisheye</td>
</tr>
<tr>
<td>SR</td>
<td>+---------------------------------+</td>
<td>upper half from left fisheye</td>
</tr>
<tr>
<td></td>
<td></td>
<td>↑ ↑ ↑ ↑</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lower half from right fisheye</td>
</tr>
</tbody>
</table>
2.72 Remove stitching artefacts

When double-fisheye images are stitched together to an equirectangular image, it's possible that stitching artefacts are visible as two vertical lines where the luminance from the two images doesn't fit together. These artefacts can be removed by applying a suitable luminance gradient at one or both sides of the border. This example applies the gradient to the left side of two vertical borders:

| set "IN=fli0z.png"      :: Input image          |
| set "B1=250"            :: Right side of first vertical border, left side is at B1-1 |
| set "B2=750"            :: Right side of second vertical border, left side is at B2-1 |
| set "W=25"              :: Width of interpolation area |

ffmpeg -i %IN% -vf "geq=cb_expr=cb(X,Y)':cr_expr=cr(X,Y)':lum_expr=clip(lum(X,Y)+between(X,%B1%-1-%W%,%B1%-1)*lerp(0,lum(%B1%,Y)-lum(%B1%-1,Y),(X-%B1%-1+%W%)/%W%),0,255)+between(X,%B2%-1-%W%,%B2%-1)*lerp(0,lum(%B2%,Y)-lum(%B2%-1,Y),(X-%B2%-1+%W%)/%W%),0,255)',format=rgb24" -y out.png

How it works:

In the area of width W to the left side of the vertical border, a ramp is added to the luminance. The amplitude of this ramp equals the difference of the luminance values left and right of the border.
You have to know in advance where exactly the vertical borders are.
Same as previous example, but now applying the gradient to the left side of the first border and to the right side of the second border:

```bash
set "IN=fli0z.png" :: Input image
set "B1=250" :: Right side of first vertical border, left side is at B1-1
set "B2=750" :: Right side of second vertical border, left side is at B2-1
set "W=25" :: Width of interpolation area

ffmpeg -i %IN% -vf "geq=cb_expr='cb(X,Y)':cr_expr='cr(X,Y)':lum_expr='clip(lum(X,Y)+
between(X,\%B1\%-1-%W%,\%B1\%-1)*lerp(0,lum(\%B1%,Y)-lum(\%B1%-1,Y),(X-%B1%-1+%W%)/%W%),
between(X,%B2%-1-%W%,%B2%-1)*lerp(0,lum(%B2%,Y)-lum(%B2%-1,Y),(X-%B2%-1+%W%)/%W%)+
between(X,\%B1%,\%B1%+%W%)*lerp(lum(%B1%-1,Y)-lum(%B1%,Y),0,(X-%B1%/\%W%)),0,255)',format=rgb24" -y out.png
```

Same as previous examples, but now applying half of the gradient to the left side and the other half to the right side of both borders:

```bash
set "IN=fli0z.png" :: Input image
set "B1=250" :: Right side of first vertical border, left side is at B1-1
set "B2=750" :: Right side of second vertical border, left side is at B2-1
set "W=25" :: Half width of interpolation area

ffmpeg -i %IN% -vf "geq=cb_expr='cb(X,Y)':cr_expr='cr(X,Y)':lum_expr='clip(lum(X,Y)+0.5*(
between(X,\%B1\%-1-%W%,\%B1%-1)*lerp(0,lum(\%B1%,Y)-lum(\%B1%-1,Y),(X-%B1%-1+%W%)/%W%),
between(X,%B2%-1-%W%,%B2%-1)*lerp(0,lum(%B2%,Y)-lum(%B2%-1,Y),(X-%B2%-1+%W%)/%W%)+
between(X,\%B1%,\%B1%+%W%)*lerp(lum(%B1%-1,Y)-lum(%B1%,Y),0,(X-%B1%/\%W%))+
between(X,%B2%,%B2%+%W%)*lerp(lum(%B2%-1,Y)-lum(%B2%,Y),0,(X-%B2%/\%W%)),0,255)',format=rgb24" -y out.png
```

Remove the line feeds from the command line, which were only inserted for clarity.

Please note that workarounds with geq filter are quite slow.
This is an example for merging two overlapping fisheye videos, realized with the "maskedmerge" filter:

```
set "IN=double_fisheye.mp4" :: Input video
set "H=640" :: Height of input video
set "FOV=191.5" :: Horizontal and vertical field of view of the fisheye lenses in degrees
set "C=11.5" :: Width of interpolation band in degrees, must be smaller or equal than (FOV-180°)
set "T=10" :: Duration in seconds
set "OUT=out.mp4" :: Output video

rem Create the mergemap file
ffmpeg -f lavfi -i nullsrc=size=%H%x%H% -vf "format=gray8,geq='clip(128-128/%C%*(180-%FOV%/(%H%/2)*hypot(X-%H%/2,Y-%H%/2)),0,255)',v360=input=fisheye:output=e:ih_fov=%FOV%:iv_fov=%FOV%" -frames 1 -y mergemap.png

rem Merge the two fisheye images from the double-fisheye input video
ffmpeg -i %IN% -i mergemap.png -lavfi "[0]format=rgb24,split[a][b];
[a]crop=ih:iw/2:0:0,v360=input=fisheye:output=e:ih_fov=%FOV%:iv_fov=%FOV%;[c];
[b]crop=ih:iw/2:iw/2:0,v360=input=fisheye:output=e:yaw=180:ih_fov=%FOV%:iv_fov=%FOV%;[d];[1]format=gbrp[e];[c][d]
[e]maskedmerge" -t %T% -y %OUT%
```

Tested with this input video, downloaded in 1280x640 size: [https://www.youtube.com/watch?v=70Wd7Ex54jE](https://www.youtube.com/watch?v=70Wd7Ex54jE)

Note: The FOV variable must be set to the correct field of view of the fisheye lenses. The procedure for finding the best value for "FOV" is as follows: Set "C" to a very small value (for example 0.5 degrees), then find the best FOV value by try and error, then set "C" to a larger value, for example 10 degrees.

Note: The "maskedmerge" filter expects the mergemap in the same pixel format as it processes the first two inputs, and these are (in this case) automatically converted to the planar gbrp pixel format. This is hard to find out, because it's not well documented. That's why the mergemap must be converted to gbrp pixel format as well.

Note: Pixel formats can be checked in the filter chain by inserting the "showinfo" filter. Another method for checking where Ffmpeg did auto-insert format conversions is to use "-v verbose" or (for even more informations) "-v debug". But it's quite hard to find the relevant informations in the long listing.
For comparison, this is the same as the previous example, but it's just hard stitching the two fisheye videos together, without any merging. Tested with the same input video as the previous example.

```bash
set "IN=double_fisheye.mp4"   :: Input video
set "FOV=191.5"               :: Field of view of the fisheye lenses, over full image height, find the best value by try and error
set "T=10"                    :: Duration in seconds
set "OUT=out.mp4"             :: Output video

ffmpeg -i %IN% -vf "v360=input=dfisheye:output=e:ih_fov=%FOV%:iv_fov=%FOV%" -t %T% -y %OUT%
```

The following example is for converting a dual-fisheye video from a Ricoh Theta camera to an equirectangular video.

The problem with this input video is that the size is 1920x1080, which is not a 2:1 aspect ratio as it should be. The input video has a black border at the bottom which must be cropped away, so that the height is reduced to 960.

```bash
set "IN=theta.mp4"            :: Input video
set "H=960"                   :: Half of the image width = height of input image after cropping
set "FOV=204"                 :: Horizontal and vertical field of view of the fisheye lenses in degrees
set "C=10"                    :: Width of interpolation band in degrees, must be smaller or equal than (FOV-180°)
set "OUT=out.mp4"             :: Output video

rem Create the mergemap file

ffmpeg -f lavfi -i nullsrc=size=%H%x%H% -vf "format=gray8,geq='clip(128-128/%C%*(180-%FOV%/(%H%/2)*hypot(X-%H%/2,Y-%H%/2)),0,255)' ,v360=input=fisheye:output=e:ih_fov=%FOV%:iv_fov=%FOV%" -frames 1 -y mergemap.png

rem Merge the two fisheye images from the double-fisheye input video

ffmpeg -i %IN% -i mergemap.png -lavfi "[0]crop=h=%H%:y=0,format=rgb24,split[a][b]; [a]crop=iw/2:0:0,v360=input=fisheye:output=e:ih_fov=%FOV%:iv_fov=%FOV%[c]; [b]crop=iw/2:iw/2:0,v360=input=fisheye:output=e:yaw=180:ih_fov=%FOV%:iv_fov=%FOV%[d];[1]format=gbrp[e];[c][d][e]maskedmerge" -y %OUT%
```

pause
If you want to rotate another part of the video into the center, insert one more v360 filter after "maskedmerge" and use the rorder/yaw/pitch/roll rotation options:

```... [1]maskedmerge,v360=input=e:output=e:rorder=rpy:roll=-95:pitch=-18" -y %OUT%```
Preprocessing a flat video for fulldome projection

If a flat video is to be shown in a fulldome planetarium with a fisheye projector, some preprocessing is required. The video is downscaled to a smaller size, padded with large black borders to equirectangular 2:1 format, rotated with the v360 filter, and then given out in 180° fisheye output.

```plaintext
set "IN=pk14.mp4"             :: Input video
set "UP=35"                   :: Up-looking angle in degrees (center of the rectangular video)
set "W=480"                   :: Width of input video after downscaling, this is for 16:9 aspect ratio
set "H=270"                   :: Height of input video after downscaling, this is for 16:9 aspect ratio
set "S=1200"                  :: Size of square fisheye output video
set "OUT=out.mp4"             :: Output video

ffmpeg -i %IN% -lavfi "scale=%W%:%H%,pad='2*%S%':%S%:-1:-1,format=pix_fmts=rgb24,v360=input=equirect:output=fisheye:h_fov=180:v_fov=180:pitch='90-%UP%'", -y %OUT%
```

It's also possible to use the flat video directly as input for the v360 filter. This has the problem that the unused area is filled with a random color (coming from the top left pixel of the input video). As a workaround, this pixel is filled with black before using the v360 filter:

```plaintext
set "IN=pk14.mp4"             :: Input video
set "UP=30"                   :: Up-looking angle in degrees (center of the rectangular video)
set "H=64"                    :: Horizontal field of view, this is for 16:9 aspect ratio
set "V=36"                    :: Vertical field of view, this is for 16:9 aspect ratio
set "OUT=out.mp4"             :: Output video

ffmpeg -i %IN% -vf drawbox=w=1:h=1:color=black,v360=input=flat:ih_fov=%H%:iv_fov=%V:output=fisheye:h_fov=180:v_fov=180:pitch='90-%UP%'", -y %OUT%
```
With sufficient computing power live processing is possible. Just drag and drop the input video over the icon of this batch file:

```
set "UP=30" :: Up-looking angle in degrees (center of the rectangular video)
set "H=64" :: Horizontal field of view, this is for 16:9 aspect ratio
set "V=36" :: Vertical field of view, this is for 16:9 aspect ratio
ffmpeg -re -i %1 -vf drawbox=w=1:h=1:color=black,v360=input=flat:ih_fov=%H%:iv_fov=%V
%:output=fisheye:h_fov=180:v_fov=180:pitch='90-%UP%'-window_fullscreen 1 -fs sdl2 -
```

Please note that the sdl2 output doesn't play audio. The Windows taskbar remains visible in full-screen mode. You can hide it as follows: Make a right click on the taskbar, click on "properties" and then select "automatically hide taskbar".

This is an example for live processing and passing the output to FFplay. Just drag and drop the input video over the icon of this batch file. FFplay has the advantage that it does also play audio, and the Windows taskbar is automatically hidden:

```
set "UP=30" :: Up-looking angle in degrees (center of the rectangular video)
set "H=64" :: Horizontal field of view, this is for 16:9 aspect ratio
set "V=36" :: Vertical field of view, this is for 16:9 aspect ratio
ffmpeg -re -i %1 -vf drawbox=w=1:h=1:color=black,v360=input=flat:ih_fov=%H%:iv_fov=%V
%:output=fisheye:h_fov=180:v_fov=180:pitch='90-%UP%'-q:v 2 -c:v mpeg4 -f nut - | c:\ffmpeg \ffplay -fs -autoexit -
```

The -fs option means full screen, and -autoexit means that FFplay closes automatically when the end of the video has been reached.
2.74 Rotating earth or planet

```plaintext
set "IN=Earth_eq.jpg" :: Equirectangular image of earth or planet surface, for example from:
                        ::  https://de.wikipedia.org/wiki/Datei:Nasa_land_ocean_ice_8192.jpg
set "BG=Starfield.jpg" :: Background image
set "P=-50" :: Pitch angle
set "R=30" :: Roll angle
set "S=-0.005" :: Rotation speed, 1.0 means one full revolution per frame
set "D=200" :: Diameter of planet
set "X=900" :: X position of planet
set "Y=450" :: y position of planet
set "T=10" :: Length of output video

ffmpeg -loop 1 -i %BG% -loop 1 -i %IN% -lavfi "[1]scroll=h=%S%,v360=e:perspective:pitch=%P%:roll=%R%
d_flip=1:alpha_mask=1, scale=%D%:%D% [a], [0] [a] overlay=x=%X%:y=%Y%" -t %T% -y out.mp4

pause
```
2.75 Tunnel effect

The idea is to horizontally stack many images together, so that we get a very wide image. In this image, connect the upper edge to the lower edge, so that we get a long cylindrical tube. Now fly through this tube with the camera.

```plaintext
set "IN=image%%3d.jpg" :: Input filenames
set "N=26" :: Number of images
set "T=3" :: Time in seconds for scrolling from one image to the next image
set /a "D=%T%*(%N%-2)" :: Duration (Warning: /a supports only integer arithmetic!)
set "IH=400" :: Height of input images
set "PW=1200" :: Width of output video
set "PH=800" :: Height of output video
set "A=1200" :: Distance from viewing point to image plane
set "E=10" :: Radius of central black dot
set "FPS=30" :: Output framerate
set "OUT=tunnel.mp4" :: Output filename

rem Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='st(0,hypot(%PW%/2-X,%PH%/2-Y));%A%*(1-%E /ld(0))' -frames 1 -y xmap.pgm

rem Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='%IH%*(0.5-atan2(%PH%/2-Y,%PW%/2-X)/(2*PI))' -frames 1 -y ymap.pgm

rem Create the tunnel video
ffmpeg -framerate 1/%T% -start_number 2 -i %IN% -framerate 1/%T% -start_number 1 -i %IN% -framerate 1/%T% -start_number 0 -i %IN% -i xmap.pgm -i ymap.pgm -filter_complex [0][1][2]hstack=inputs=3,fps=%FPS%,crop=w=2*iw/3:x='iw/3*(1-mod(t,%T %)/%T%)',format=pix_fmts=rgb24[5];[5][3][4]remap -t %D% -y %OUT%
```

pause
An alternative approach is to project the images on a cone instead of a cylinder. Only the xmap file must be changed:

```bash
set "C=200"               :: Distance from image plane to the vanishing point
rem Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='st(0,hypot(%PW%/2-%X,%PH%/2-%Y));%A*%C
%*(ld(0)-%E%)/(ld(0)*%C%+%A%*%E%)*sqrt(1+%E%*%E%/%C%/%C%)' -frames 1 -y xmap.pgm
```

Note: For $C \to \infty$ the formula is the same as the cylinder case.

A drawback of the above projections is the discontinuity where the upper edge touches the lower edge, which is visible as a straight line. This can be avoided by duplicating the input image, so that the same image appears twice around the cylinder. There is no visible discontinuity because the bottom edges of both images touch each other, and the upper edges as well. Only the ymap file must be changed:

```bash
rem Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='%IH%*(abs(atan2(%PW%/2-%X,%PH%/2-%Y)/PI))-0.5' -frames 1 -y ymap.pgm
```
This is the tunnel effect with an additional spiral effect:

```
set "IN=image%%3d.jpg"    :: Input filenames
set "N=6"                 :: Number of images
set "T=3"                 :: Time in seconds for scrolling from one image to the next image
set /a "D=%T%*(%N%-2)"    :: Duration (Warning: /a supports only integer arithmetic!)
set "IH=400"              :: Height of input images
set "PW=1200"             :: Width of output video
set "PH=800"              :: Height of output video
set "A=1200"              :: Distance from viewing point to image plane
set "E=10"                :: Radius of central black dot
set "S=500"               :: Spiral effect, number of pixels in radial direction for a 360° rotation
set "FPS=30"              :: Output framerate
set "OUT=tunnel.mp4"      :: Output filename

rem  Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='st(0,hypot(%PW%/2-X,%PH%/2-Y));%A%*(1-%E /ld(0))' -frames 1 -y xmap.pgm

rem  Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='%IH%*(mod(hypot(%PW%/2-X,%PH%/2-Y)/%S%+0.5-atan2(%PH%/2-Y,%PW%/2-X)/(2*PI),1))' -frames 1 -y ymap.pgm

rem  Create the tunnel video
ffmpeg -framerate 1/%T% -start_number 2 -i %IN% -framerate 1/%T% -start_number 1 -i %IN% -framerate 1/%T% -start_number 0 -i %IN% -i xmap.pgm -i ymap.pgm -filter_complex [0][1][2]hstack=inputs=3,fps=%FPS%,crop=w=2*iw/3:x='iw/3*(1-mod(t,%T %)/%T%)' ,format=pix_fmts=rgb24[5],[5][3][4]remap -t %D% -y %OUT%
```

pause
How the mathematics of this filter works:

A is the distance from the viewing point to the output image plane.

E is the radius of the central area in the output image, which will remain black.

R is the radial coordinate in the output image.

X is the horizontal coordinate in the input image.

C is a parameter that defines how far the vanishing point is behind the output image plane.

In the special case of a cylindrical tube (C → ∞) the formula simplifies to:  \[ X = A \left(1 - \frac{E}{R}\right) \]

You must make sure that for the maximum possible value of R the resulting X value doesn't exceed the width of the input image.
2.76 Black hole simulation with remap filter

FFmpeg's remap filter can be used to simulate the light deviation near black holes. When a beam of light passes near a black hole, it will be deviated by angle alpha (in Radians):

\[
\alpha = \frac{2 \cdot rs}{r - rs}
\]

where \(rs\) is the Schwarzschild radius of the black hole, and \(r\) is the closest distance between the beam and the center of the black hole.

Assuming we have a 180° fisheye image, we can express the light deviation in pixels:

\[
c = \frac{\text{height}}{\pi} \cdot \frac{2 \cdot rs}{r - rs}
\]

The values for the PGM files (which are required for the remap filter) can be calculated with these formulas:

\[
r = \sqrt{(x - xc)^2 + (y - yc)^2}
\]
\[
c = \frac{\text{shape}}{(r - rs)}
\]

where \(shape\) is a constant that defines the "strength" of the distortion

if \(r > rs\):

\[
x_{\text{remap}} = x - c \cdot (x - xc)
\]
\[
y_{\text{remap}} = y - c \cdot (y - yc)
\]

if \(r < rs\):

\[
x_{\text{remap}} = 0
\]
\[
y_{\text{remap}} = 0
\]

where \(xc, yc\) are the pixel coordinates of the center of the black hole, \(x, y\) are the pixel coordinates in the source video and \(r\) is the distance between the source pixel and the center of the black hole.
This is the batch file for applying the black-hole-effect to a video:

```bash
set "IN=MVI_2562.mov"       :: Input video
set "OUT=output.mp4"        :: Output video

ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -c:v mpeg4 -q:v 2 -y %OUT%
pause
```

It's also possible to simulate moving black holes. To do this you need many xmap and ymap files (one for each frame), and loop through them.

```bash
set "IN=MVI_2562.mov"       :: Input video
set "OUT=output.mp4"        :: Output video

ffmpeg -i %IN% -framerate 30 -i xmap%%4d.pgm -framerate 30 -i ymap%%4d.pgm -lavfi "format=pix_fmts=rgb24,remap" -c:v mpeg4 -q:v 2 -y %OUT%
pause
```
This is a C# program for creating the xmap and ymap files for a moving black hole:
using System;
using System.IO;
using System.Globalization;
namespace Moving_Black_Hole
{
class Program
{
static void Main(string[] args)
{
int width = 1920;
// Width of images
int height = 1080;
// Height of images
double shape = 10;
// Strength factor of black hole effect
int count = 250;
// Number of frames
int framerate = 25; // Framerate for conversion of frame number to time
double ignore = 0.1; // Central fraction of radius that's not used for mapping, can be deactivated by setting to 0
// Use this parameter to hide the small white ball on the invisible tripod.
// Example: If the black hole radius is 50 pixels and the small ball has 5 pixels radius,
// set ignore to 0.1
string path = @"F:\Wormhole_2020\";
// Path for writing the xmap and ymap files
int[,] nxyr = new int[11, 4]

// This array contains the interpolation data: n, x, y, r
// n is the frame number beginning with 0,
//
the last entry must not be smaller than the number of frames
// x and y are the center coordinates of the black hole
// r is the black hole's radius

{
{
{
{
{
{
{
{
{
{
{
{

0,
25,
50,
75,
100,
125,
150,
175,
200,
225,
250,

1735,
1723,
1605,
1440,
1266,
1113,
1007,
966,
964,
964,
964,

527,
529,
526,
526,
523,
523,
522,
522,
522,
522,
522,

50 },
50 },
50 },
50 },
50 },
50 },
50 },
70 },
90 },
1 },
1 }

};
int xc, yc, radius, dx, dy, xb, yb;

156


CultureInfo invC = CultureInfo.InvariantCulture;
TextWriter pos = File.CreateText(path + "positions.cmd");  // Write positions.cmd file
for (int i = 0; i < nxyr.GetLength(0) - 1; i++)
{
    int t0 = nxyr[i, 0] / framerate;
    int t1 = nxyr[i + 1, 0] / framerate;
    int x0 = nxyr[i, 1];
    int x1 = nxyr[i + 1, 1];
    int y0 = nxyr[i, 2];
    int y1 = nxyr[i + 1, 2];
    int r0 = nxyr[i, 3];
    int r1 = nxyr[i + 1, 3];
    pos.WriteLine(t0.ToString("F2", invC) + ":" + t1.ToString("F2", invC) + 
                  ", overlay x 'lerp(" + (x0 - r0).ToString(invC).PadLeft(4) + 
                  ", " + (x1 - r1).ToString(invC).PadLeft(4) + ", t-" + i.ToString() + 
                  ")', overlay y 'lerp(" + (y0 - r0).ToString(invC).PadLeft(4) + 
                  ", " + (y1 - r1).ToString(invC).PadLeft(4) + ", t-" + i.ToString() + 
                  ")', scale w 'lerp(" + (2 * r0 + 1).ToString(invC).PadLeft(4) + 
                  ", " + (2 * r1 + 1).ToString(invC).PadLeft(4) + ", t-" + i.ToString() + 
                  ")', scale h 'lerp(" + (2 * r0 + 1).ToString(invC).PadLeft(4) + 
                  ", " + (2 * r1 + 1).ToString(invC).PadLeft(4) + ", t-" + i.ToString() + 
                  ")');
}
pos.Close();

for (int n = 0; n < count; n++)
{
    int i = 0;
    while ((n < nxyr[i, 0]) || (n > nxyr[i + 1, 0]))
        i++;
    xc = nxyr[i, 1] + (nxyr[i + 1, 1] - nxyr[i, 1]) * (n - nxyr[i, 0]) / (nxyr[i + 1, 0] - nxyr[i, 0]);
    yc = nxyr[i, 2] + (nxyr[i + 1, 2] - nxyr[i, 2]) * (n - nxyr[i, 0]) / (nxyr[i + 1, 0] - nxyr[i, 0]);
    radius = nxyr[i, 3] + (nxyr[i + 1, 3] - nxyr[i, 3]) * (n - nxyr[i, 0]) / (nxyr[i + 1, 0] - nxyr[i, 0]);
    Console.WriteLine("Writing mapping files for frame " + (n + 1).ToString() + " of " + count.ToString() + 
    " x: " + xc.ToString() + " y: " + yc.ToString() + " r: " + radius.ToString());

TextWriter xmap = File.CreateText(path + "xmap" + n.ToString("0000") + ".pgm");
xmap.Write("P2\n");
xmap.Write("# Xmap file for FFmpeg remap \n");
xmap.Write(width.ToString() + " " + height.ToString() + " \n");
xmap.Write("65535\n");
TextWriter ymap = File.CreateText(path + "ymap" + n.ToString("0000") + ".pgm");
ymap.Write("P2\n");
ymap.Write("# Ymap file for FFmpeg remap \n");
ymap.Write(width.ToString() + " " + height.ToString() + " \n");
ymap.Write("65535\n");

for (int y = 0; y < height; y++)
{
    dy = y - yc;
    for (int x = 0; x < width; x++)
    {
        dx = x - xc;
        double r = Math.Sqrt(dx * dx + dy * dy);
        if (r > radius) // outer area of black hole
        {
            double c = shape / (r - radius);
            if (c > 1.0 - ignore) c += 2 * ignore;
            xb = x - (int)(1.0 + dx * c);
            yb = y - (int)(1.0 + dy * c);
            if (xb < 0) xb = 0;
            if (yb < 0) yb = 0;
            if (xb >= width) xb = width - 1;
            if (yb >= height) yb = height - 1;
        }
        else
        {
            xb = 65535; // inner area of black hole is declared as "unmapped"
            yb = 65535;
        }
        xmap.Write(xb + " ");
        ymap.Write(yb + " ");
    }
}
xmap.WriteLine("\n");
ymap.WriteLine("\n");
xmap.Close();
ymap.Close();
Console.WriteLine("All done!");
}
Example of a simulated black hole:
Black hole simulation with FFmpeg, no C# code required:

```plaintext
set "W=2448"               :: Width of image
set "H=2448"               :: Height of image
set "CX=2000"              :: X center of distortion
set "CY=1200"              :: Y center of distortion
set "RS=50"                :: Schwarzschild radius
set "SH=0.50"              :: Shape parameter

rem  Create the xmap file
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray16le,geq='st(0,X-%CX%);st(1,hypot(ld(0),%CY%-Y));st(2,X-(ld(0)*%SH%*2*%RS%/(ld(1)-%RS%)));if(lt(%RS%,ld(1)),clip(ld(2),0,%W%),0)' -frames 1 -y xmap.pgm
rem  Create the ymap file
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray16le,geq='st(0,Y-%CY%);st(1,hypot(%CX%-X,ld(0)));st(2,Y-(ld(0)*%SH%*2*%RS%/(ld(1)-%RS%)));if(lt(%RS%,ld(1)),clip(ld(2),0,%H%),0)' -frames 1 -y ymap.pgm
rem  Apply the displace filter to the image
ffmpeg -i test3.mp4 -i xmap.pgm -i ymap.pgm -lavfi format=pix_fmts=rgb24,remap -frames 1 -y out.jpg
rem  Alternatively all can be written in one command line:
ffmpeg -i test3.mp4 -f lavfi -i nullsrc=size=%W%x%H% -f lavfi -i nullsrc=size=%W%x%H% -lavfi [0]format=pix_fmts=rgb24[v];[1]format=pix_fmts=gray16le,geq='st(0,X-%CX%);st(1,hypot(ld(0),%CY%-Y));st(2,X-(ld(0)*%SH %2*%RS%/(ld(1)-%RS%)));if(lt(%RS%,ld(1)),clip(ld(2),0,%W%),0)';[2]format=pix_fmts=gray16le,geq='st(0,Y-%CY %);st(1,hypot(%CX%-X,ld(0)));st(2,Y-(ld(0)*%SH%2*%RS%/(ld(1)-%RS%)));if(lt(%RS%,ld(1)),clip(ld(2),0,%H%),0)' [v][x][y]remap -frames 1 -y out.jpg
```
This example is for a moving black hole, no C# code required (but unfortunately this is extremely slow):

```
set "IN=test3.mp4"     :: Input video
set "W=2448"           :: Width of video
set "H=2448"           :: Height of video
set "CX0=2000"         :: X center of distortion, T=0
set "CY0=1200"         :: Y center of distortion, T=0
set "CX1=1900"         :: X center of distortion, T=1
set "CY1=1500"         :: Y center of distortion, T=1
set "CX2=1800"         :: X center of distortion, T=2
set "CY2=1800"         :: Y center of distortion, T=2
set "CX3=1000"         :: X center of distortion, T=3
set "CY3=2000"         :: Y center of distortion, T=3
set "RS=50"            :: Schwarzschild radius
set "SH=0.50"          :: Shape parameter
set "OUT=out.mp4"      :: Output video
```

```
ffmpeg -i %IN% -f lavfi -i nullsrc=size=%W%x%H% -f lavfi -i nullsrc=size=%W%x%H% -lavfi ^
[0]format=pix_fmts=rgb24[v];^  
[1]format=pix_fmts=gray16le,geq='^  
[2]format=pix_fmts=gray16le,geq='^  
[l0](0,between(T+0.001,0,1)*lerp(%CX0%,%CX1%,T)+between(T+0.001,1,2)*lerp(%CX1%,%CX2%,T-1)+between(T+0.001,2,3)*lerp(%CX2%,%CX3%,T-2)));^  
[l1](1,between(T+0.001,0,1)*lerp(%CY0%,%CY1%,T)+between(T+0.001,1,2)*lerp(%CY1%,%CY2%,T-1)+between(T+0.001,2,3)*lerp(%CY2%,%CY3%,T-2)));^  
[l2](2,X-ld(0));^  
[l3](3,hypot(ld(1),X-(ld(2)*%SH%*2*%RS%/(ld(3)-%RS%))));^  
[l4](4,Y-hypot(ld(0)-X,ld(2)));^  
[l5](x,y)remap -t 3 -y %OUT%
```

"T+0.001" is a workaround to avoid the problem that at the segment borders two "between" expressions become simultaneously true.

This method is extremely slow because this expression must be evaluated four times for each pixel, although it would be sufficient to evaluate it only one time per frame:

```
[l1](1,between(T+0.001,0,1)*lerp(%CY0%,%CY1%,T)+between(T+0.001,1,2)*lerp(%CY1%,%CY2%,T-1)+between(T+0.001,2,3)*lerp(%CY2%,%CY3%,T-2));
```

Recommended workaround: Calculate many xmap and ymap files in advance by C# code.
2.77 Wormhole simulation

A wormhole is a hypothetical window to another place in space or time, or even in another universe.
For more informations please see https://en.wikipedia.org/wiki/Wormhole
Short story from Rudy Rucker: "The Last Einstein-Rosen Bridge” http://www.rudyrucker.com/transrealbooks/completetopics/#_Toc14

A wormhole can be simulated in a video as follows:

- In the outer area the light rays are distorted in the same way as when passing near a black hole. This can be simulated with the remap filter.
- In the inner area, another video is inserted as a 360° "little planet" video (or even better a mirror-sphere video).
This is a batch file for wormhole simulation. The xmap0000 and ymap0000 files for the black hole are created in advance by C# code.

```
set "IN=main.mov"                :: Main input video
set "LP=test1.mp4"               :: Equirectangular video for little planet
set "H=960"                      :: Height of equirectangular input video
set "S=1080"                     :: Size of square little planet output video
set "P=0"                        :: Pitch angle
set "Y=90"                       :: Yaw angle
set "R=-90"                      :: Roll angle
set "LPD=100"                    :: Little planet diameter
set "LPX=1500"                   :: X Position of center of little planet
set "LPY=1000"                   :: Y Position of center of little planet
set "T=8"                        :: Length of output video

rem Step 1: Convert the equirectangular video to a little planet video

rem  Create the xmap and ymap files

ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^geq='H*(0.9999+atan2(Y-%S%/2,X-%S%/2)/PI)' -frames 1 -y xmap.pgm

ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=pix_fmts=gray16le,^geq='H*(1-hypot((2*X/%S%)-1,(2*Y/%S%)-1))' -frames 1 -y ymap.pgm

rem  Apply the remap filter to the video

ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -filter_complex "[0][2][3]remap[4];[1]scale=%LPD%:%LPD%,format=argb,geq=a=255*lt(hypot((2*X/W)-1,(2*Y/H)-1),1):r='r(X,Y)':g='g(X,Y)':b='b(X,Y)'[5];[4][5]overlay=x=%LPX%-LPD%/2:y=%LPY%-LPD%/2" -q:v 2 -t %T% -y out.mp4

rem Step 2: Apply the black hole effect to the main video and then overlay the little planet video over the black hole

ffmpeg -i %IN% -i lp.mp4 -i xmap0000.pgm -i ymap0000.pgm -filter_complex "[0][2]remap[4];[1]scale=%LPD%:%LPD%,format=argb,geq=a=255*lt(hypot((2*X/W)-1,(2*Y/H)-1),1):r='r(X,Y)':g='g(X,Y)':b='b(X,Y)'[5];[4][5]overlay=x=%LPX%-LPD%/2:y=%LPY%-LPD%/2" -q:v 2 -t %T% -y out.mp4
```

pause
The same thing can be done much easier with the v360 filter and the alpha_mask option:

```
set "IN=main.mov"             :: Main input video
set "LP=test1.mp4"            :: Equirectangular video for mirror-sphere
set "H=960"                   :: Height of equirectangular input video
set "S=1080"                  :: Size of square mirror-sphere output video
set "P=30"                    :: Pitch angle
set "Y=0"                     :: Yaw angle
set "R=0"                     :: Roll angle
set "LPD=102"                 :: Mirror-sphere diameter
set "LPX=1800"                :: X Position of center of mirror-sphere
set "LPY=1000"                :: Y Position of center of mirror-sphere
set "T=8"                     :: Length of output video

rem  Make only the mirror-sphere video
rem ffmpeg -i %LP% -vf v360=output=ball:pitch=%P%:yaw=%Y%:roll=%R% -q:v 2 -t %T% -y lp.mp4

ffmpeg -i %IN% -i xmap0000.pgm -i ymap0000.pgm -i %LP% -filter_complex "[0][1][2]remap[4][3]v360=output=ball:pitch=%P%:yaw=%Y%:roll=%R%:alpha_mask=1, scale=%LPD%:%LPD%[5][4]overlay=x=%LPX%-%LPD%/2:y=%LPY%-%LPD%/2" -q:v 2 -t %T% -y out.mp4

pause
```
2.78 Simulation of a moving wormhole

If the wormhole shall move in the field of view, two things must move:

1. The black hole distortion must move. This requires many unique xmap and ymap files for each frame. These files are created by a C# program.
2. The inserted mirror-sphere video must move. This can be realized with sendcmd and overlay filters.

Step 1:
In the main video it's recommended to have a small object at that position in space where the wormhole shall later be inserted. This can for example be a small white ball (about 8mm diameter) on an almost invisible tripod (which can be built with 0.3mm diameter carbon fibers).

The x,y coordinates of this object must be measured in each frame. There are two methods how to measure the x,y coordinates: Either extract many images from the video and measure the coordinates manually (this is described in step 2), or preferably extract the coordinates automatically with FFprobe and find_rect filter. In this case continue with step 3.

Step 2:
Extract a suitable number of frames from the main video:

```
set "IN=in.mp4" :: Input video
set "STEP=1" :: Step width (number of frames)
set "OUT=image%%4d.jpg" :: Output images filename
ffmpeg -i %IN% -vf framestep=%STEP% -start_number 0 -y %OUT%
```

Measure the x,y coordinates of the small object in each frame and enter the positions in the "measured.csv" file. Set the "offset" variable in the C# program to 0. Continue with step 4.
Step 3:

Create a small 40x40 pixel grayscale image of the object and then automatically extract the x,y coordinates from the main video:

```
set "IN=in.mp4"               :: Input video
set "OBJ=needle.pgm"          :: Image of the object, must be gray8
set "TH=0.4"                  :: Threshold, 0.01 = only exact matches, 0.99 = almost everything matches
set "XMIN=900"                :: Minimum x position of the object's top left corner
set "XMAX=1900"               :: Maximum x position of the object's top left corner
set "YMIN=490"                :: Minimum y position of the object's top left corner
set "YMAX=510"                :: Maximum y position of the object's top left corner

ffprobe -f lavfi movie=%IN%,find_rect=object=%OBJ%:threshold=%TH%:xmin=%XMIN%:xmax=%XMAX%:ymin=%YMIN%:ymax=%YMAX%
-show_entries frame=pkt_pts_time:frame_tags=lavfi.rect.x,lavfi.rect.y -of csv=p=0 l> measured.csv
```

Pause

Note: To speed up the algorithm, make the object image as small as possible (40x40 pixels) and specify a search window with the xmin, xmax, ymin, ymax options.

This is the resulting logfile. If no coordinates are written in a line, then no object was found for the specified threshold. In this case you can try a larger threshold value, or you have to enter the coordinates manually.

```
0.000000
0.040000
0.080000
0.120000,45,1
0.160000,45,1
0.200000,45,1
0.240000,45,1
...
```

Note: These coordinates are for the top left corner of the 40x40 pixels search window. Set the "offset" variable in the C# program to 20.
Step 4:

Calculate the xmap and ymap files for all frames. This is done by a C# program:

```csharp
using System;
using System.IO;
using System.Globalization;

namespace Moving_Black_Hole
{
    class Program
    {
        static void Main(string[] args)
        {
            int width = 1920; // Width of images
            int height = 1080; // Height of images
            double shape = 10; // Strength factor of black hole effect
            double ignore = 0.1; // Central fraction of radius that's not used for mapping, can be deactivated by setting to 0
            // Use this parameter to hide the small white ball on the invisible tripod.
            // Example: If the black hole radius is 50 pixels and the small ball has 5 pixels radius,
            //  set ignore to 0.1
            int offset = 20; // This is half the size if the search window and will be added to the x,y positions
            string path = @"F:\Wormhole_2020"; // Path for reading the input file and writing the xmap and ymap files
            int dx, dy, xb, yb;
            string[] lines = File.ReadAllLines(path + "measured.csv");
            double[] t = new double[lines.Length];
            int[] x = new int[lines.Length];
            int[] y = new int[lines.Length];
            double[] r = new double[lines.Length];
            for (int n = 0; n < lines.Length; n++)
            {
                string[] entries = lines[n].Split(new char[] {','});
                t[n] = System.Convert.ToDouble(entries[0], CultureInfo.InvariantCulture);
                x[n] = offset + System.Convert.ToInt32(entries[1]);
                y[n] = offset + System.Convert.ToInt32(entries[2]);
                r[n] = 50;
                if (n > 150 && n <= 200) // specify the variable radius of the wormhole
                    r[n] = 50 + (n - 150) * 0.8;
                if (n > 200 && n <= 225)
                    r[n] = 90 - (n - 200) * 3.56;
                if (n > 225)
                    r[n] = 1;
            }

            TextWriter pos = File.CreateText(path + "positions.cmd"); // Write positions.cmd file
        }
    }
}
```
for (int n = 0; n < lines.Length; n++)
{
    pos.WriteLine(t[n].ToString("F2", CultureInfo.InvariantCulture) +
        " overlay x " + (x[n] - r[n]).ToString("F2", CultureInfo.InvariantCulture).PadLeft(4) +
        " overlay y " + (y[n] - r[n]).ToString("F2", CultureInfo.InvariantCulture).PadLeft(4) +
        " scale w " + (2 * r[n]).ToString("F2", CultureInfo.InvariantCulture).PadLeft(3) +
        " scale h " + (2 * r[n]).ToString("F2", CultureInfo.InvariantCulture).PadLeft(3) + ");
}
pos.Close();

for (int n = 0; n < lines.Length; n++)
{
    int xc = x[n];
    int yc = y[n];
    double radius = r[n];
    Console.WriteLine("Writing mapping files for frame " + (n + 1).ToString() + " of " + lines.Length.ToString() +
        " x: " + xc.ToString() + " y: " + yc.ToString() + " r: " + radius.ToString(CultureInfo.InvariantCulture));
    TextWriter xmap = File.CreateText(path + "xmap" + n.ToString("0000") + ".pgm");
    xmap.Write("P2
");
    xmap.Write("# Xmap file for FFmpeg remap 
");
    xmap.Write(width.ToString() + " " + height.ToString() + " \n");
    xmap.Write("65535\n");
    TextWriter ymap = File.CreateText(path + "ymap" + n.ToString("0000") + ".pgm");
    ymap.Write("P2\n");
    ymap.Write("# Ymap file for FFmpeg remap \n");
    ymap.Write(width.ToString() + " " + height.ToString() + " \n");
    ymap.Write("65535\n");
    for (int yy = 0; yy < height; yy++)
    {
        dy = yy - yc;
        for (int xx = 0; xx < width; xx++)
        {
            dx = xx - xc;
            double rr = Math.Sqrt(dx * dx + dy * dy);
            if (rr > radius) // outer area of black hole
            {
                double c = shape / (rr - radius);
                if (c > 1.0 - ignore) c *= 2 * ignore;
                xb = xx - (int)(1.0 + dx * c);
                yb = yy - (int)(1.0 + dy * c);
                if (xb < 0) xb = 0;
                if (yb < 0) yb = 0;
                if (xb >= width) xb = width - 1;
                if (yb >= height) yb = height - 1;
            }
        }
    }
Step 5:
The file positions.cmd was also automatically created by the C# program (many lines omitted here):

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Overlay X</th>
<th>Overlay Y</th>
<th>Scale Width</th>
<th>Scale Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>1665.00</td>
<td>454.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>0.04</td>
<td>1665.00</td>
<td>454.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>0.08</td>
<td>1665.00</td>
<td>454.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>6.00</td>
<td>939.00</td>
<td>449.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>6.04</td>
<td>935.20</td>
<td>448.20</td>
<td>101.60</td>
<td>101.60</td>
</tr>
<tr>
<td>6.08</td>
<td>932.40</td>
<td>447.40</td>
<td>103.20</td>
<td>103.20</td>
</tr>
<tr>
<td>6.12</td>
<td>928.60</td>
<td>446.60</td>
<td>104.80</td>
<td>104.80</td>
</tr>
<tr>
<td>6.16</td>
<td>925.80</td>
<td>445.80</td>
<td>106.40</td>
<td>106.40</td>
</tr>
<tr>
<td>6.20</td>
<td>922.00</td>
<td>445.00</td>
<td>108.00</td>
<td>108.00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Step 6: Run this batch file to create the final moving wormhole video:

```plaintext
set "IN=in.mp4"               :: Main input video
set "LP=WL4149.mp4"           :: Equirectangular video for mirror-sphere
set "P=0"                     :: Pitch angle
set "Y=0"                     :: Yaw angle
set "R=60"                    :: Roll angle
set "R1=0.00"                 :: Rotation speed before v360 filter, 1.0 means one revolution per frame
set "R2=0.00"                 :: Rotation speed after v360 filter, 1.0 means one revolution per second
set "V=9"                     :: Time when wormhole vanishes
set "LUT=lut2.cube"           :: Look-Up-Table
set "T=12"                    :: Length of output video

rem  Make only the mirror-sphere video
rem ffmpeg -i %LP% -vf v360=output=ball:pitch=%P%:yaw=%Y%:roll=%R% -t %T% -y lp.mp4

ffmpeg -i %IN% -ss 9 -i %LP% -start_number 0 -i xmap%4d.pgm -start_number 0 -i ymap%4d.pgm -lavfi "[0]sendcmd='%V% streamselect map 1',split[4][5]:[4][2][3]remap=fill=red,sendcmd=f=positions.cmd[6][1] fps=25,v360=e:e:pitch=0,scroll=h=%R1%,v360=output=ball:pitch=%P%:yaw=%Y%:roll=%R %:alpha_mask=1,rotate='%R2%*2*PI*t':c=black@0.0, scale=w=10:h=10:eval=frame,lut3d=%LUT%[7][6][7] overlay=x=0:y=0:format=rgb[8][8][5]streamselect=map=0,format=yuv420p" -t %T% -y out.mp4

pause
```

overlay=format=rgb is strongly required, because the default format yuv420 allows only to set the x,y coordinates in multiples of 2.

"remap=fill=red" is used here only to make alignment errors visible, if the overlay isn't exactly at the same position as the black hole distortion. Normally there should no red pixels be visible. After this check you can replace it by "remap=fill=black".

"fps=25" is used here because the mirror-sphere video has a different framerate (30fps), which resulted in jerky scaling of this video.

It's also possible to let the inner area of the wormhole rotate as a function of time. Two different rotations are applied in this example. The first rotation is using the scroll filter (applied to an equirectangular video, before the v360 filter) and the other is using the rotate filter after the v360 filter.

Note for scroll filter: scroll=h=1.0 means one full horizontal revolution per frame. So you have to know the framerate to set the rotation speed.
2.79 Sendcmd and commands

- sendcmd has many pitfalls and can drive you crazy!
- The sendcmd filter sends commands to another filter. For example in the previous chapter sendcmd was used to send the x and y coordinates to the overlay filter. The commands are defined in the file positions.cmd, or could also be defined in the command line.
- Normally the sendcmd filter is inserted in the filter chain somewhere before the target filter. A problem arises when the target filter has more than one input (for example overlay has two inputs). This doesn't work, because sendcmd accepts only one input. In this case sendcmd must be inserted somewhere earlier in the filter chain, where only one input exists.
- It's important that sendcmd is inserted at a position in the filter chain that has sufficient duration. For example, if the overlaid video is shorter than the main video, and if sendcmd is inserted in the input of the shorter video, that would give unexpected behaviour, because when the shorter video has ended, sendcmd will get the wrong time (which stays then constant), and will send wrong commands to the other filters based on the wrong time. Always insert sendcmd at the longest input.
- It's also possible to have more than one sendcmd in the filter chain, for example at both inputs.
- It's also possible to insert sendcmd after the target filter, for example at the end of the filter chain. The drawback of this method is that the changes become effective with one frame delay.
- All arguments of the sendcmd target filter must be initialized with valid values, even if these values are never used because sendcmd does always overwrite them.
- It's also possible to evaluate an expression in sendcmd and send the result to the target filter. To enable expression evaluation the [expr] flag must be used instead of the default [enter] flag. Inside the expression the "TI" variable can be used, which runs from 0 to 1 in the interval.
- If a line in the *.cmd file begins with a "#" character then it's a comment. Empty lines are allowed as well.
- The *.cmd file must contain at least one command. It's not allowed if it contains only comments.
- If the filter chain contains multiple filters of the same type, they must be given unique names, for example "v360@1", "v360@2".
- (I have not tested this with FFmpeg. It doesn't work with FFplay) There is another (undocumented) way how to send commands. In the same console window where FFmpeg is running, type "c" or "C" and immediately (without a space character) let the command follow, for example:
  Cdrawtext 15.3 reinit 'x=752:y=480'<enter>
  In this example "15.3" is the time. You can use "all" instead of the filter/class instance if you want to send the command to all filters that can receive it. Instead of pressing <enter> you can also send "\n". Found here:
This is an example of a rotating earth, where the rotation axis rapidly changes (this is physically impossible) and the observer's viewing point changes:

```plaintext
set "IN=Earth_eq.jpg" :: Equirectangular input image of earth
:: from https://de.wikipedia.org/wiki/Datei:Nasa_land_ocean_ice_8192.jpg
set "BG=Starfield.jpg" :: Background image 1920x1080
set "R1=0.01" :: Rotation speed, 1.0 means one revolution per frame
set "D=400" :: Diameter of earth
set "X=760" :: X position of earth
set "Y=340" :: y position of earth
set "T=30" :: Length of output video

ffmpeg -loop 1 -i %BG% -loop 1 -i %IN% -lavfi \
[1]
sendcmd=v360@1=e:e:pitch=0:yaw=0:roll=0,scroll=h=
%R1%,v360@2=e:e:pitch=0:yaw=0:roll=0,v360=e:perspective:pitch=90:v_flip=1:alpha_mask=1,
scale=%D%:%D%[a],[0][a]
overlay=x=%X%:y=%Y% -t %T% -y out.mp4
pause

rem This is a workaround for faulty "perspective" output: v360=e:perspective:pitch=90:v_flip=1
rem Normally "pitch=90:v_flip=1" should be unnecessary.

sendcmd.cmd
# set the initial conditions
0.00 v360@1 pitch 0;
0.00 v360@1 yaw 0;
0.00 v360@1 roll 0;
0.00 v360@2 pitch 140;
0.00 v360@2 yaw 0;
0.00 v360@2 roll 180;

# from t = 8s to 9s change the rotation axis of the earth by 60° from the north pole to Cairo in Egypt
# Latitude 30° north, Longitude 30° east
8.00-9.001 [expr] v360@1 pitch 'lerp(0,-60,TI)';
8.00-9.001 [expr] v360@1 yaw 'lerp(0,30,TI)';

# from t = 14s to 15s change the viewing point, so that the observer is on the rotation axis:
```

172
14.00-15.00 [expr] v360@2 pitch 'lerp(140,90, TI)';

# from t = 18s to 21s change the rotation axis of the earth from Cairo to New York
# Latitude 41° north, Longitude 74° west

18.00-21.00 [expr] v360@1 pitch 'lerp(-60,-49, TI)';
18.00-21.00 [expr] v360@1 yaw 'lerp(30,-74, TI)';
A few examples for sendcmd and single / double quotes:

<table>
<thead>
<tr>
<th>Command</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| `ffmpeg -i in1.mp4 -i in2.mp4 -lavfi
"[0]sendcmd='3.0 streamselect map 1'[a];[a][1]streamselect=inputs=2:map=0" out.mp4` | sendcmd at the beginning of the filter chain, double quotes for whole filter chain. Works fine, but would give unexpected results if the first input is shorter than the second input! |
| `ffmpeg -i in1.mp4 -i in2.mp4 -lavfi
"[0][1]sendcmd='3.0 streamselect map 1',streamselect=inputs=2:map=0" out.mp4` | This doesn't work because sendcmd accepts only one input                     |
| `ffmpeg -i in1.mp4 -i in2.mp4 -lavfi
sendcmd='3.0 streamselect map 1',streamselect=inputs=2:map=0 out.mp4`     | This is the example from the streamselect documentation, doesn't work under Windows. |
| `ffmpeg -i in1.mp4 -i in2.mp4 -lavfi
sendcmd="3.0 streamselect map 1",streamselect=inputs=2:map=0 out.mp4`    | Single quotes replaced by double quotes, works under Windows.                |
| `ffmpeg -i in1.mp4 -i in2.mp4 -lavfi
"sendcmd='3.0 streamselect map 1',streamselect=inputs=2:map=0" out.mp4`  | Double quotes added for the whole filter chain, single quotes for sendcmd argument, works under Windows. |
| `ffmpeg -i in1.mp4 -i in2.mp4 -lavfi
"[0][1]streamselect@my=inputs=2:map=0,sendcmd='3.0 streamselect@my map 1'" out.mp4` | [0][1] added, sendcmd at the end of the filter chain, commands become effective with one frame delay. Double quotes for filter chain, single quotes for sendcmd argument |
| `ffmpeg -i in1.mp4 -i in2.mp4 -lavfi
"[0][1]streamselect@my=inputs=2:map=0,sendcmd"3.0 streamselect@my map 1" out.mp4` | [0][1] added, sendcmd at the end of the filter chain, commands become effective with one frame delay. No quotes for filter chain, double quotes for sendcmd argument |

Note about double quotes around the filter chain:

In Windows it's not required to put the whole filter chain in double quotes, but it seems these double quotes are required on a Mac. Not tested myself.

If a filter chain contains the "|" character, it must be encapsulated in "" double quotes.

A note from Moritz Barsnick in the FFmpeg user list, April 5, 2020:

"Under Windows, "'" is not a command line quotation character. If parts of the command line need to be quoted in order to be collated, you need to use the double quotation mark '"'. The single quotation mark is passed directly to ffmpeg on Windows, making the filter argument unparsable. [...] I believe a large part of ffmpeg's examples in doc and wiki are "wrong" in this matter, and could or should be changed to also work on Windows, where possible."
## 2.80 Sending commands with ZMQ

This is an example of a command line with the "zmq" filter, which receives messages that were sent from somewhere else. It's possible to send messages to all filters and all options that accept commands (as indicated in the documentation of the filter). These are the same options as for the "sendcmd" filter.

```plaintext
ffplay -dumpgraph 1 -f lavfi "color=s=200x200:c=red[l];color=s=200x200:c=blue[r];nullsrc=s=400x200,zmq[bg];[bg][l]overlay[bg+l];[bg+l][r]overlay@my=x=200"
pause
```

Note: "-dumpgraph" is not required in this example. This option is only available in FFplay and not in FFmpeg. It draws a graphical representation of the filtergraph in the console output.

Note: This example is copied from the documentation of the zmq filter, but the sizes were changed because it didn't work with the original smaller sizes. Seems to be a bug in FFplay.

Note: the zmq filter has the option "bind_address" or "b" which is by default set to "tcp://*:5555". You can change this value to your needs, but don't forget to escape any ':' signs.

Note: Don't send messages to those options that don't support commands. This could lead to malfunction.

For details about ZMQ (or ZeroMQ), see [https://zeromq.org/](https://zeromq.org/)

The commands can be sent from a command line in another console window by using the zmqsend.exe tool.

Copy the files zmqsend.exe and avutil-56.dll in the same directory where you have ffmpeg.exe (In this example I'm using the folder `c:\ffmpeg`). Open a second console window and type this command:

```plaintext
echo overlay@my x 150 | c:\ffmpeg\zmqsend
```

Question: What can be done if this error message appears?

```
[Parsed_zmq_3 @ 000001d92beb4180] Could not bind ZMQ socket to address 'tcp://*:5555': Address in use
[lavfi @ 000001d92beabb80] Error initializing filter 'zmq'
```
It's also possible to send ZMQ commands from a C# program with the NetMQ library, as in this short example:

```csharp
using NetMQ;
using NetMQ.Sockets;
using System;
using System.Windows.Forms;

namespace netmq
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            using (var client = new RequestSocket())
            {
                client.Connect("tcp://localhost:5555");
                // this works, but I don't understand it:
                // what does "tcp://" mean?
                // what does "localhost:" mean?
                // what does "5555" mean?
                // If you can explain it, please let me know

                client.SendFrame("overlay@my x 150");
                var message = client.ReceiveFrameString();
                Console.WriteLine("Received {0}", message);
            }
        }
    }
}
```

Note: Use Nuget to add the NetMQ package, and add the two using directives at the beginning.

Please note that not everything that's possible is also useful. For example, a message can be sent to the "width" option of the "scale" filter for changing the size of a video stream. But changing the size mid-stream isn't supported by many other filters (for example "eq", "colorkey" and "despill"). In some cases it works (for example "scale" immediately followed by "overlay"), but in most other cases it fails.
## 2.81 Comparison: sendcmd / zmq

<table>
<thead>
<tr>
<th></th>
<th>sendcmd</th>
<th>zmq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which options can be controlled?</td>
<td>All those that have support for commands.</td>
<td>All those that have support for commands.</td>
</tr>
<tr>
<td>When can an option be changed?</td>
<td>Options can be changed at exact times, but these times must be known in advance, because they must be written in the *.cmd file or in the command line.</td>
<td>Options can be changed in real time while FFmpeg is running, but the timing is not as exact as with sendcmd.</td>
</tr>
<tr>
<td>Timing accuracy</td>
<td>exact</td>
<td>about 100-200ms, as a rough estimate</td>
</tr>
<tr>
<td>Is it possible to decide in real time to change or not to change an option?</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Possible sources where data comes from</td>
<td>• *.cmd file</td>
<td>• from a second console window via &quot;zmqsend&quot; tool</td>
</tr>
<tr>
<td></td>
<td>• FFmpeg command line, as argument of sendcmd filter</td>
<td>• from a batch file via &quot;zmqsend&quot; tool</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• zeromq library which is available for many programming languages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• for example NetMQ library for C#</td>
</tr>
<tr>
<td>Is it possible to change an option for each frame, using an expression?</td>
<td>yes, with [expr] option</td>
<td>No, not with exact timing for each frame. However it's possible if exact timing for each frame isn't required</td>
</tr>
</tbody>
</table>
2.82 FFmpeg and C# Programming

Start a FFplay process with console window and video window:

```csharp
Process p;
ProcessStartInfo startInfo = new ProcessStartInfo();
startInfo.FileName = "ffplay";
startInfo.Arguments = "-f lavfi testsrc2=s=vga";
p = Process.Start(startInfo);
```

Note: This assumes that you have set the PATH environment variable accordingly, so that FFplay is found. Otherwise you can use `startInfo.WorkingDirectory = ...`

Start a FFplay process without console window:

```csharp
Process p;
ProcessStartInfo startInfo = new ProcessStartInfo();
startInfo.UseShellExecute = false;
startInfo.CreateNoWindow = true;
startInfo.FileName = "ffplay";
startInfo.Arguments = "-f lavfi testsrc2=s=vga";
p = Process.Start(startInfo);
```

See also: [https://stackoverflow.com/questions/19756860/executing-an-external-program-via-c-sharp-without-showing-the-console](https://stackoverflow.com/questions/19756860/executing-an-external-program-via-c-sharp-without-showing-the-console)

Stop the FFplay process:

```csharp
p.kill();
```

Get a filename from a command line argument, this works also if you start the C# program by dropping a file on the program's icon:

```csharp
private void Form1_Shown(object sender, EventArgs e)
{
    string[] arguments = Environment.GetCommandLineArgs(); // get the filename by command line argument,
    if (arguments.Length == 2) // this works also for drag-and-drop
        myFilename = arguments[1];
}
```
C# Sample program for real-time zmq adjustment of brightness and contrast with scrollbars:

```csharp
using NetMQ;
using NetMQ.Sockets;
using System;
using System.Diagnostics;
using System.Globalization;
using System.Windows.Forms;
namespace netmq
{
    public partial class Form1 : Form
    {
        Process p;
        RequestSocket client;

        public Form1()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e) // Start the FFplay process
        {
            if (p == null)
            {
                ProcessStartInfo startInfo = new ProcessStartInfo();
                startInfo.UseShellExecute = false;
                startInfo.CreateNoWindow = true;
                startInfo.FileName = "ffplay";
                startInfo.Arguments = "-top 0 -left 0 -f lavfi \"testsrc2=s=960x540,zmq,eq@my\"";
                richTextBox1.AppendText(startInfo.Arguments);
                p = Process.Start(startInfo);
            }
        }

        private void button3_Click(object sender, EventArgs e) // Stop the FFplay process
        {
            if ((p != null) && !p.HasExited)
            {
                p.Kill();
                p = null;
            }
        }

        private void Form1_Shown(object sender, EventArgs e) // Create and connect the client for zmq
        {
            client = new RequestSocket();
            client.Connect("tcp://localhost:5555"); // This works, but I don't understand it:
            // What does "tcp://" mean?
        }
    }
}
```
// What does "localhost:" mean?
// Why 5555?
// If you can explain it, please let me know

private void hScrollBar1_Scroll(object sender, ScrollEventArgs e) // Scrollbar for brightness adjustment
{
    client.SendFrame("eq@my brightness " + ((double)hScrollBar1.Value * 0.02).ToString(CultureInfo.InvariantCulture));
    var message = client.ReceiveFrameString();
    Console.WriteLine("Received {0}", message);
}

private void hScrollBar2_Scroll(object sender, ScrollEventArgs e) // Scrollbar for contrast adjustment
{
    client.SendFrame("eq@my contrast " + ((double)hScrollBar2.Value * 0.1).ToString(CultureInfo.InvariantCulture));
    var message = client.ReceiveFrameString();
    Console.WriteLine("Received {0}", message);
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e) // Cleanup
{
    if ((p != null) && !p.HasExited)
    {
        p.Kill();
    }
}

Note: In the command line for FFplay it’s not necessary to specify any values for contrast and brightness. Just writing "eq" or "eq@my" is sufficient.

It might be a problem that the code is blocking if the zmq message can’t be sent. In this case it’s better to specify a 500ms timeout for receiving the reply:

private void vScrollBar1_Scroll(object sender, ScrollEventArgs e)
{
    if ((p != null) && !p.HasExited)
    {
        string message;
        client.SendFrame("overlay@my y " + y);
        client.TryReceiveFrameString(TimeSpan.FromMilliseconds(500), out message);
    }
}

See also my example in chapter "Real-time bluescreening".
2.83 Video stabilization

Videos can be stabilized in a one-pass process with "deshake" filter or in a two-pass process with "vidstabdetect" and "vidstabtransform" filters.

```
set "IN=C1000650.MOV" :: Input video
set "OUT=C0650_stab.MOV" :: Output video
rem Stabilize the video
ffmpeg -i %IN% -vf vidstabdetect -y dummy.mov
del dummy.mov
ffmpeg -i %IN% -vf vidstabtransform -y %OUT%
pause
```

This is the same thing, but with 10-bit DNxHD (Digital Nonlinear Extensible High Definition) codec for importing in the free DaVinci Resolve version:

```
set "IN=C1000645.MOV" :: Input video
set "OUT=C0645_stab.MOV" :: Output video
rem Stabilize the video
ffmpeg -i %IN% -vf vidstabdetect -y dummy.mov
del dummy.mov
ffmpeg -i %IN% -vf vidstabtransform -map_metadata 0 -pix_fmt yuv422p10le -c:v dnxhd -profile:v 4 -c:a pcm_s24le -color_range pc -movflags write_colr -y %OUT%
pause
```
2.84 Stabilization of 360° Videos

360° videos can’t be stabilized the same way as normal (flat) videos. A normal video can be stabilized by following one point and applying x and y shifts. In a 360° video, two points must be followed and rotations in three axes (yaw, pitch, roll) must be applied.

Let’s begin by making a nice 360° test video. First get a fulldome test pattern and create an equirectangular test image:

```bash
set "IN=1200.png"                   :: Test pattern from http://www.paulbourke.net/dome/testpattern/1200.png
ffmpeg -i %IN% -i %IN% -lavfi 
  '[0]transpose=1[left];[1]transpose=2,negate[right];[left]
  [right]hstack,v360=input=dfisheye:output=e:pitch=90" -y equirectangular_test.png
```

Now use this image for creating a test video with rotations around different axes:

```bash
set "IN=equirectangular_test.png"   :: Equirectangular input image
ffmpeg -loop 1 -i %IN% -lavfi sendcmd=f=sendcmd.cmd,v360=e:e:pitch=0:yaw=0:roll=0,drawtext=text=''::x=(w-text_w)/2:y=0.7*h;fontsize=160;fontcolor=red:boxcolor=yellow:box=1:boxborderw=20 -t 9 -y test.mp4
```

The rotations are defined in the file "sendcmd.cmd":

```
0.0-1.0 drawtext reinit 'text=';
0.0-1.0 v360 pitch 0;
0.0-1.0 v360 yaw 0;
0.0-1.0 v360 roll 0;

1.0-2.0 drawtext reinit 'text=PITCH';
1.0-2.0 [expr] v360 pitch 'lerp(0,45, TI)';
1.0-2.0 v360 yaw 0;
1.0-2.0 v360 roll 0;

2.0-3.0 drawtext reinit 'text=YAW';
```
2.0-3.0 v360 pitch 45;
2.0-3.0 [expr] v360 yaw 'lerp(0,45,TI)';
2.0-3.0 v360 roll 0;

3.0-4.0 drawtext reinit 'text=PITCH + YAW';
3.0-4.0 [expr] v360 pitch 'lerp(45,90,TI)';
3.0-4.0 [expr] v360 yaw 'lerp(45,90,TI)';
3.0-4.0 v360 roll 0;

4.0-5.0 drawtext reinit 'text=ROLL';
4.0-5.0 v360 pitch 90;
4.0-5.0 v360 yaw 90;
4.0-5.0 [expr] v360 roll 'lerp(0,45,TI)';

5.0-6.0 drawtext reinit 'text=PITCH + ROLL';
5.0-6.0 [expr] v360 pitch 'lerp(90,135,TI)';
5.0-6.0 v360 yaw 90;
5.0-6.0 [expr] v360 roll 'lerp(45,90,TI)';

6.0-7.0 drawtext reinit 'text=YAW + ROLL';
6.0-7.0 v360 pitch 135;
6.0-7.0 [expr] v360 yaw 'lerp(90,135,TI)';
6.0-7.0 [expr] v360 roll 'lerp(90,135,TI)';

7.0-8.0 drawtext reinit 'text=PITCH + YAW + ROLL';
7.0-8.0 [expr] v360 pitch 'lerp(135,180,TI)';
7.0-8.0 [expr] v360 yaw 'lerp(135,180,TI)';
7.0-8.0 [expr] v360 roll 'lerp(135,180,TI)';

8.0-9.0 drawtext reinit 'text=';
8.0-9.0 v360 pitch 180;
8.0-9.0 v360 yaw 180;
8.0-9.0 v360 roll 180;
How large is the image shift in this test video, from one image to the next? The image height is 1200 pixels and that's 180 degrees. So the image scale is 0.15° per pixel. The rotation speed is 45° per second. So the image shift at the default 25fps framerate is 1.8° per frame or 12 pixel per frame.

Let's check this and extract small grayscale images from the center of the test video (images Axxx.png), and also from a point 90° right of the center (images Bxxx.png):

```
set "IN=test.mp4"             :: Equirectangular input image
set "B=100"                   :: Image size in pixels
set "T=10"                    :: Duration in seconds

ffmpeg -i %IN% -vf crop=w=%B%:h=%B%,format=gray8 -start_number 0 -t %T% -y a%%04d.png
ffmpeg -i %IN% -vf v360=input=e:output=e:yaw=90,crop=w=%B%:h=%B%,format=gray8 -start_number 0 -t %T% -y B%%04d.png
```

Note: The duration must only be specified if you want to analyze only the beginning of the video, for fast testing. If you want to analyze the whole video, just set the "T" variable to a value longer than the video.

Let's have a look which movements we can see in the small images:

```
<table>
<thead>
<tr>
<th>Time [s]</th>
<th>Frames</th>
<th>Rotation</th>
<th>Movement in image A (center of equirectangular video)</th>
<th>Movement in image B (90° right of center of equirectangular video)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-1</td>
<td>0-25</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>1-2</td>
<td>25-50</td>
<td>pitch</td>
<td>down</td>
<td>counter-clockwise rotation</td>
</tr>
<tr>
<td>2-3</td>
<td>50-75</td>
<td>yaw</td>
<td>left</td>
<td>down and left</td>
</tr>
<tr>
<td>3-4</td>
<td>75-100</td>
<td>pitch + yaw</td>
<td>down and left</td>
<td>first down and left, then down</td>
</tr>
<tr>
<td>4-5</td>
<td>100-125</td>
<td>roll</td>
<td>counter-clockwise rotation</td>
<td>up</td>
</tr>
<tr>
<td>5-6</td>
<td>125-150</td>
<td>pitch + roll</td>
<td>first down and right, then right</td>
<td>up and right</td>
</tr>
<tr>
<td>6-7</td>
<td>150-175</td>
<td>yaw + roll</td>
<td>first up, then up and left</td>
<td>first up, then up and left</td>
</tr>
<tr>
<td>7-8</td>
<td>175-200</td>
<td>pitch + yaw + roll</td>
<td>first up, then up and left</td>
<td>first up, then up and left</td>
</tr>
<tr>
<td>8-9</td>
<td>200-224</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>
```

184
That means we can measure the image shift in the small images in x and y direction and then calculate the rotation angles as follows:

\[
\begin{align*}
\text{Yaw\_angle\_in\_radians} &= -c \times \text{horizontal\_shift\_in\_image\_A} \\
\text{Pitch\_angle\_in\_radians} &= c \times \text{vertical\_shift\_in\_image\_A} \\
\text{Roll\_angle\_in\_radians} &= -c \times \text{vertical\_shift\_in\_image\_B}
\end{align*}
\]

where \(c\) is a constant, \(c = \pi / \text{equirectangular\_image\_height}\)

Before the A and B images can be analyzed for movement, you should make sure that they contain useful details. For example, they should not contain the blue sky which has no fine details. Also, when you have for example a video of a mountainbiker, make sure that the images contain only the background and not the bike or the driver (under the assumption that you want to stabilize on the background).

This batch file draws two red rectangles around the small windows, so that you can check if these windows contain useful details:

```bash
set "IN=test.mp4"             :: Equirectangular input image
set "B=100"                   :: Box size in pixels
ffmpeg -i %IN% -vf
drawbox=x=iw/2-%B%/2:y=ih/2-%B%/2:w=%B%/2:h=%B%/2:color=red:thickness=5,drawbox=x=3/4*iw-%B%/2:y=ih/2-%B%/2:w=%B%/2:h=%B%/2:color=red:thickness=5 -y dummy.mp4
```

If you find that the small windows contain unsuitable content, then use the V360 filter to rotate the video and then repeat the process.

The image shifts in x and y direction in the A and B images are now analyzed by this C# program, which produces a file "stabilize.cmd" which can later be used as input for FFmpeg's sendcmd filter.

```csharp
using System;
using System.Drawing;
using System.Globalization;
using System.IO;
using System.Windows.Forms;

namespace v360stabilizer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
```
}

InitializeComponent();

private void button1_Click(object sender, EventArgs e)
{
int shift_max = 25;
// the shift range is from -shift_max to +shift_max
int n_max = 224;
// index of the last image, the total number of images is n_max + 1
int image_height = 1200;
// height of equirectangular video
string path = @"C:\Users\astro\Desktop\stabilize360\";
// this is the path to the input images A0000.png and B0000.png
double fps = 25;
// framerate
double c = Math.PI / image_height;
// constant for calculating the angles in radians
CultureInfo invC = CultureInfo.InvariantCulture;
TextWriter cmd = File.CreateText(path + "stabilize.cmd");
double yaw = 0;
double pitch = 0;
double roll = 0;
double delta_yaw;
double delta_pitch;
double delta_roll;
for (int n = 0;
{
int minimum
int best_ax
int best_ay
int best_bx
int best_by

n < n_max; n++)
=
=
=
=
=

int.MaxValue;
0;
0;
0;
0;

Bitmap bitmap1 = new Bitmap(path + "A" + n.ToString("0000") + ".png");
int size = bitmap1.Width;
int[,] image1 = new int[size, size];
int wsize = size - 1 - (2 * shift_max);

// size of the compare window

pictureBox1.Image = bitmap1;
pictureBox1.Refresh();
for (int x = 0; x < size; x++)
for (int y = 0; y < size; y++)
image1[x, y] = (int)bitmap1.GetPixel(x, y).R;
Bitmap bitmap2 = new Bitmap(path + "A" + (n + 1).ToString("0000") + ".png");
int[,] image2 = new int[size, size];
pictureBox2.Image = bitmap2;
pictureBox2.Refresh();
for (int x = 0; x < size; x++)
for (int y = 0; y < size; y++)
image2[x, y] = (int)bitmap2.GetPixel(x, y).R;

186


for (int shift_x = -shift_max; shift_x <= shift_max; shift_x++)  // loop shift_x from -shift_max to +shift_max
{
    for (int shift_y = -shift_max; shift_y <= shift_max; shift_y++)  // loop shift_y from -shift_max to +shift_max
    {
        int sum = 0;
        for (int x = 0; x < wsize; x++)
        {
            for (int y = 0; y < wsize; y++)
            {
                int p1 = image1[shift_max + x + shift_x, shift_max + y + shift_y];
                int p2 = image2[shift_max + x, shift_max + y];
                sum += Math.Abs(p1 - p2);  // sum up the absolute differences
            }
        }
        if (sum < minimum)  // found a better minimum?
        {
            minimum = sum;
            best_ax = shift_x;
            best_ay = shift_y;
        }
    }
}
// now do the same thing for the B images
minimum = int.MaxValue;
bitmap1 = new Bitmap(path + "B" + n.ToString("0000") + ".png");
size = bitmap1.Width;
image1 = new int[size, size];
wsize = size - 1 - (2 * shift_max);  // size of the compare window
pictureBox3.Image = bitmap1;
pictureBox3.Refresh();
for (int x = 0; x < size; x++)
    for (int y = 0; y < size; y++)
        image1[x, y] = (int)bitmap1.GetPixel(x, y).R;
bitmap2 = new Bitmap(path + "B" + (n + 1).ToString("0000") + ".png");
image2 = new int[size, size];
pictureBox4.Image = bitmap2;
pictureBox4.Refresh();
for (int x = 0; x < size; x++)
    for (int y = 0; y < size; y++)
        image2[x, y] = (int)bitmap2.GetPixel(x, y).R;
for (int shift_x = -shift_max; shift_x <= shift_max; shift_x++)  // loop shift_x from -shift_max to +shift_max
{
for (int shift_y = -shift_max; shift_y <= shift_max; shift_y++)  // loop shift_y from -shift_max to +shift_max
{
    int sum = 0;
    for (int x = 0; x < wsize; x++)
    {
        for (int y = 0; y < wsize; y++)
        {
            int p1 = image1[shift_max + x + shift_x, shift_max + y + shift_y];
            int p2 = image2[shift_max + x, shift_max + y];
            sum += Math.Abs(p1 - p2);  // sum up the absolute differences
        }
    }
    if (sum < minimum)  // found a better minimum?
    {
        minimum = sum;
        best_bx = shift_x;
        best_by = shift_y;
    }
}

delta_pitch = c * best_ax;    // angles are in radians
delta_yaw = -c * best_ax;
delta_roll = -c * best_by;

// This is the rotation matrix with the delta angles delta_yaw, delta_pitch and delta_roll
// These are the difference angles between two frames

double r11 = Math.Cos(delta_yaw) * Math.Cos(delta_pitch);
double r12 = Math.Sin(delta_yaw) * Math.Cos(delta_pitch);
double r13 = -Math.Sin(delta_pitch);
double r23 = Math.Cos(delta_pitch) * Math.Sin(delta_roll);
double r33 = Math.Cos(delta_pitch) * Math.Cos(delta_roll);

// This is the rotation matrix with the accumulated angles yaw, pitch and roll

double s11 = Math.Cos(yaw) * Math.Cos(pitch);
double s12 = Math.Sin(yaw) * Math.Cos(pitch);
double s13 = -Math.Sin(pitch);
double s23 = Math.Cos(pitch) * Math.Sin(roll);
double s33 = Math.Cos(pitch) * Math.Cos(roll);
// multiply the matrices r * s = t

double t11 = r11 * s11 + r21 * s12 + r31 * s13;
// four of the matrix elements are not required
// double t21 = r11 * s21 + r21 * s22 + r31 * s23;
// double t31 = r11 * s31 + r21 * s32 + r31 * s33;
double t12 = r12 * s11 + r22 * s12 + r32 * s13;
// double t22 = r12 * s21 + r22 * s22 + r32 * s23;
// double t32 = r12 * s31 + r22 * s32 + r32 * s33;
double t13 = r13 * s11 + r23 * s12 + r33 * s13;
// double t23 = r13 * s21 + r23 * s22 + r33 * s23;
double t33 = r13 * s31 + r23 * s32 + r33 * s33;

// calculate the rotation angles from the resulting matrix
if ((t13 == 1.0) || (t13 == -1.0))
{
    yaw = 0;
    double d = Math.Atan2(t12, t13);
    if (t13 == -1.0)
    {
        pitch = -Math.PI / 2;
        roll = -yaw + d;
    }
    else
    {
        pitch = Math.PI / 2;
        roll = yaw + d;
    }
}
else
{
    yaw = Math.Atan2(t12, t11);
    pitch = -Math.Asin(t13);
    roll = Math.Atan2(t23, t33);
}

// write the results in the rich textbox window
richTextBox1.AppendText("N: "+n.ToString("0000")+
" AX: "+best_ax.ToString("000")+
" AY: "+best_ay.ToString("000")+
" BX: "+best_bx.ToString("000")+
" BY: "+best_by.ToString("000")+
" PITCH: "+(pitch / Math.PI * 180.0).ToString("00.00")+
" YAW: "+(yaw / Math.PI * 180.0).ToString("00.00")+
" ROLL: "+(roll / Math.PI * 180.0).ToString("00.00")+"\n");

// write the results to the *.cmd file

cmd.WriteLine(((n + 1) / fps).ToString("F2", invC) +
"\n");
"v360 yaw " + (yaw / Math.PI * 180.0).ToString("00.00", invC) +
"v360 pitch " + (pitch / Math.PI * 180.0).ToString("00.00", invC) +
"v360 roll " + (roll / Math.PI * 180.0).ToString("00.00", invC) + ");
}
cmd.Close();
}

Form1.designer.cs

namespace v360stabilizer
{
    partial class Form1
    {
        /// <summary>
        /// Erforderliche Designervariable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        /// <param name="disposing">True, wenn verwaltete Ressourcen gelöscht werden sollen; andernfalls False.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Vom Windows Form-Designer generierter Code

        /// <summary>
        /// Erforderliche Methode für die Designerunterstützung.
        /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
        /// </summary>
        private void InitializeComponent()
        {
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.pictureBox2 = new System.Windows.Forms.PictureBox();
            this.richTextBox1 = new System.Windows.Forms.RichTextBox();
            this.button1 = new System.Windows.Forms.Button();
            this.pictureBox3 = new System.Windows.Forms.PictureBox();
            this.pictureBox4 = new System.Windows.Forms.PictureBox();
        }
    }
}
private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.PictureBox pictureBox2;
private System.Windows.Forms.RichTextBox richTextBox1;
private System.Windows.Forms.Button button1;
private System.Windows.Forms.PictureBox pictureBox3;
private System.Windows.Forms.PictureBox pictureBox4;
}
This is the output file stabilize.cmd (many lines omitted):

```
0.04 v360 yaw 00.00, v360 pitch 00.00, v360 roll 00.00;
0.08 v360 yaw 00.00, v360 pitch 00.00, v360 roll 00.00;
0.12 v360 yaw 00.00, v360 pitch 00.00, v360 roll 00.00;
...  
1.00 v360 yaw 00.00, v360 pitch 00.00, v360 roll 00.00;
1.04 v360 yaw 00.00, v360 pitch -01.80, v360 roll 00.00;
1.08 v360 yaw 00.00, v360 pitch -03.60, v360 roll 00.00;
...  
2.00 v360 yaw 00.00, v360 pitch -45.00, v360 roll 00.00;
...  
3.00 v360 yaw -54.59, v360 pitch -30.61, v360 roll 34.87;
...  
4.00 v360 yaw -89.94, v360 pitch -01.24, v360 roll 90.24;
...  
5.00 v360 yaw -89.91, v360 pitch -46.24, v360 roll 90.18;
...  
6.00 v360 yaw 174.14, v360 pitch -87.95, v360 roll -128.88;
...  
7.00 v360 yaw 17.59, v360 pitch -60.99, v360 roll 14.57;
...  
8.00 v360 yaw 03.27, v360 pitch -01.53, v360 roll -00.37;
...  
8.96 v360 yaw 03.27, v360 pitch -01.53, v360 roll -00.37;
```

You can see that the accumulated angle error is only a few degrees at the end of the video. In the input video the angles were 180°, 180°, 180° at the end of the video, which is equivalent to 0°, 0°, 0°.
Finally apply the corrections to the test video:

<table>
<thead>
<tr>
<th>set &quot;IN=test.mp4&quot;</th>
<th>:: Equirectangular input image</th>
</tr>
</thead>
<tbody>
<tr>
<td>set &quot;T=10&quot;</td>
<td>:: Duration in seconds</td>
</tr>
</tbody>
</table>

```bash
ffmpeg -i %IN% -lavfi sendcmd=f=stabilize.cmd,v360=e:e:pitch=0:yaw=0:roll=0 -t %T% -y out.mp4
```

pause

When you play this video, you see that most of the rotations are removed. Now I must produce a real 360° video on my mountainbike :-)  

Of course, there is room for improvements:

- Use more than two windows for detecting the image shifts. For example 6 windows front, rear, left, right, up, down.
- This would add redundancy and make the algorithm more robust.
- Automatically detect if a window contains no useful details and don't use this data
- Automatically detect if one window contains details that don't move in the same direction as the other windows. Some kind of median filtering.
- Use all R, G and B channels for detecting the image shifts.
2.85 Stabilization of 360° Videos, improved

This version has the following improvements:

- Extract color images and use R, G and B channels for image shift detection, all colors can be individually enabled or disabled.
- Use 6 images from each frame for rotation detection: front (XA), right (XB), back (XC), left (XD), up (XE) and down (XF), all 6 images can be individually enabled or disabled.
- Offset angles can be applied to the output file as well.
- Use median filtering for the rotation angles.
- Faster algorithm for finding the minimum of the sum of absolute differences.
- Set the time values in the *.cmd file in the middle between the time stamps of the frames.

These are the movements in the 6 image sequences:

<table>
<thead>
<tr>
<th>Movement in images</th>
<th>XA</th>
<th>XB</th>
<th>XC</th>
<th>XD</th>
<th>XE</th>
<th>XF</th>
</tr>
</thead>
<tbody>
<tr>
<td>pitch down</td>
<td>up</td>
<td>down</td>
<td>down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yaw left</td>
<td>left</td>
<td>left</td>
<td>left</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>roll up</td>
<td>down</td>
<td>left</td>
<td>right</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

There is a lot of redundancy: 12 independent ways to calculate pitch (4 for each color channel), and the same for yaw and roll as well.

This is the batch file for extracting the 6 image sequences from the input video:

```
set "IN=test.mp4"             :: Equirectangular input image
set "B=100"                   :: Image size in pixels
set "T=9"                     :: Duration in seconds

ffmpeg -i %IN% -vf crop=w=%B%:h=%B% -start_number 0 -t %T% -y XA%%04d.png
ffmpeg -i %IN% -vf v360=input=e:output=e:yaw=90,crop=w=%B%:h=%B% -start_number 0 -t %T% -y XB%%04d.png
```
This is the C# source code:

```csharp
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Globalization;
using System.Windows.Forms;

namespace v360stabilizer
{
    public partial class Form1 : Form
    {
        // global variables
        Bitmap bitmap1, bitmap2;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Shown(object sender, EventArgs e)
        {
            int size = 100;         // size of the quadratic images
            int shift_max = 25;     // the shift range is from -shift_max to +shift_max
            int n_max = 224;        // index of the last image, the total number of images is n_max + 1
            int image_height = 1200;   // height of equirectangular video
            double fps = 25;        // framerate
            double highpass = 1.00;  // use 1.0 for full corrections, in this case all errors are accumulating
            // use for example 0.95 if you want to preserve the initial angles
            // using 0 would disable all angle corrections
            Boolean[] enable = new Boolean[3];
            enable[0] = true;        // enable checking red channel At least one color channel must be enabled
            enable[1] = true;        // enable checking green channel
            enable[2] = false;       // enable checking blue channel
            bool enable_front = true; // enable checking the XA images At least two images which are 90° apart must be enabled
            bool enable_right = true; // enable checking the XB images for example enabling only XA and XB is sufficient,
            bool enable_back = true; // enable checking the XC images however enabling only XA and XC is wrong because they are 180° apart
            bool enable_left = true; // enable checking the XD images
            bool enable_up = true;   // enable checking theXE images
            bool enable_down = true; // enable checking the XF images
            int remove = 0;         // number of elements to remove from the beginning and end of the sorted lists (yaw, pitch, roll)
```
offset_yaw = 0;  // this will be applied to the output
offset_pitch = 0;  // this will be applied to the output
offset_roll = 0;  // this will be applied to the output

string path = @"C:\Users\astro\Desktop\stabilize360\";  // this is the path to the input images

CultureInfo invC = CultureInfo.InvariantCulture;
offset_yaw *= Math.PI / 180.0;
offset_pitch *= Math.PI / 180.0;
offset_roll *= Math.PI / 180.0;

double yaw = 0;
double pitch = 0;
double roll = 0;
double delta_yaw;
double delta_pitch;
double delta_roll;
double yaw_out;
double pitch_out;
double roll_out;
double r11, r12, r13, r21, r22, r23, r31, r32, r33;
double s11, s12, s13, s21, s22, s23, s31, s32, s33;
double t11, t12, t13, t21, t22, t23, t31, t32, t33;
List<int> list_yaw = new List<int>();
List<int> list_pitch = new List<int>();
List<int> list_roll = new List<int>();
Point shift;

for (int n = 0; n < n_max; n++)
{
    list_yaw.Clear();  // clear the lists
    list_pitch.Clear();
    list_roll.Clear();

    if (enable_front)
    {
        bitmap1 = new Bitmap(path + "XA" + n.ToString("0000") + ".png");
pictureBox1.Image = bitmap1;
        bitmap2 = new Bitmap(path + "XA" + (n + 1).ToString("0000") + ".png");
pictureBox2.Image = bitmap2;
        for (int c = 0; c < 3; c++)
        {
            if (enable[c] == true)
            {
                shift = shift_detect(bitmap1, bitmap2, shift_max, size, c);
                list_pitch.Add(shift.Y);
                list_yaw.Add(-shift.X);
            }
        }
    }
}
if (enable_right)
{
    bitmap1 = new Bitmap(path + "XB" + n.ToString("0000") + ".png");
    pictureBox3.Image = bitmap1;
    bitmap2 = new Bitmap(path + "XB" + (n + 1).ToString("0000") + ".png");
    pictureBox4.Image = bitmap2;
    for (int c = 0; c < 3; c++)
    {
        if (enable[c] == true)
        {
            shift = shift_detect(bitmap1, bitmap2, shift_max, size, c);
            list_yaw.Add(-shift.X);
            list_roll.Add(-shift.Y);
        }
    }
}

if (enable_back)
{
    bitmap1 = new Bitmap(path + "XC" + n.ToString("0000") + ".png");
    pictureBox5.Image = bitmap1;
    bitmap2 = new Bitmap(path + "XC" + (n + 1).ToString("0000") + ".png");
    pictureBox6.Image = bitmap2;
    for (int c = 0; c < 3; c++)
    {
        if (enable[c] == true)
        {
            shift = shift_detect(bitmap1, bitmap2, shift_max, size, c);
            list_pitch.Add(-shift.Y);
            list_yaw.Add(-shift.X);
        }
    }
}

if (enable_left)
{
    bitmap1 = new Bitmap(path + "XD" + n.ToString("0000") + ".png");
    pictureBox7.Image = bitmap1;
    bitmap2 = new Bitmap(path + "XD" + (n + 1).ToString("0000") + ".png");
    pictureBox8.Image = bitmap2;
    for (int c = 0; c < 3; c++)
    {
        if (enable[c] == true)
        {
            shift = shift_detect(bitmap1, bitmap2, shift_max, size, c);
            list_yaw.Add(-shift.X);
            list_roll.Add(shift.Y);
        }
    }
}
if (enable_up)
{
    bitmap1 = new Bitmap(path + "XE" + n.ToString("0000") + ".png");
pictureBox9.Image = bitmap1;
    bitmap2 = new Bitmap(path + "XE" + (n + 1).ToString("0000") + ".png");
pictureBox10.Image = bitmap2;
    for (int c = 0; c < 3; c++)
    {
        if (enable[c] == true)
        {
            shift = shift_detect(bitmap1, bitmap2, shift_max, size, c);
            list_pitch.Add(shift.Y);
            list_roll.Add(-shift.X);
        }
    }
}

if (enable_down)
{
    bitmap1 = new Bitmap(path + "XF" + n.ToString("0000") + ".png");
pictureBox11.Image = bitmap1;
    bitmap2 = new Bitmap(path + "XF" + (n + 1).ToString("0000") + ".png");
pictureBox12.Image = bitmap2;
    for (int c = 0; c < 3; c++)
    {
        if (enable[c] == true)
        {
            shift = shift_detect(bitmap1, bitmap2, shift_max, size, c);
            list_pitch.Add(shift.Y);
            list_roll.Add(shift.X);
        }
    }
}

this.Refresh();

// check if all lists contain at least one element
if ((list_yaw.Count - 2 * remove < 1) || (list_pitch.Count - 2 * remove < 1) || (list_roll.Count - 2 * remove < 1))
    richTextBox1.AppendText("Error: Not enough data! Enable more images or colors or use smaller value for remove.
");
else
{
    list_yaw.Sort(); // sort the lists
    list_pitch.Sort();
    list_roll.Sort();

    list_yaw.RemoveRange(0, remove); // remove some elements at the beginning
    list_yaw.RemoveRange(list_yaw.Count - remove, remove); // remove some elements at the end
    delta_yaw = 0;
    foreach (double d in list_yaw) // calculate the average of the remaining elements
delta_yaw += d;
delta_yaw = Math.PI / image_height * delta_yaw / list_yaw.Count;

list_pitch.RemoveRange(0, remove);  // remove some elements at the beginning
list_pitch.RemoveRange(list_pitch.Count - remove, remove);  // remove some elements at the end
delta_pitch = 0;
foreach (double d in list_pitch)   // calculate the average of the remaining elements
    delta_pitch += d;
delta_pitch = Math.PI / image_height * delta_pitch / list_pitch.Count;

list_roll.RemoveRange(0, remove);  // remove some elements at the beginning
list_roll.RemoveRange(list_roll.Count - remove, remove);  // remove some elements at the end
delta_roll = 0;
foreach (double d in list_roll)   // calculate the average of the remaining elements
    delta_roll += d;
delta_roll1 = Math.PI / image_height * delta_roll / list_roll.Count;

// this is the rotation matrix with the delta angles delta_yaw, delta_pitch and delta_roll
// these are the difference angles between two frames
r11 = Math.Cos(delta_yaw) * Math.Cos(delta_pitch);
     r12 = Math.Sin(delta_yaw) * Math.Cos(delta_pitch);  
     r13 = -Math.Sin(delta_pitch);

     r23 = Math.Cos(delta_pitch) * Math.Sin(delta_roll);

     r33 = Math.Cos(delta_pitch) * Math.Cos(delta_roll);

// This is the rotation matrix with the accumulated angles yaw, pitch and roll
s11 = Math.Cos(yaw) * Math.Cos(pitch);
     s12 = Math.Sin(yaw) * Math.Cos(pitch);  
     s13 = -Math.Sin(pitch);

     s23 = Math.Cos(pitch) * Math.Sin(roll);

     s33 = Math.Cos(pitch) * Math.Cos(roll);

// multiply the matrices t = r * s  (four of the matrix elements are not required)
t11 = r11 * s11 + r21 * s12 + r31 * s13;
t12 = r12 * s11 + r22 * s12 + r32 * s13;
t13 = r13 * s11 + r23 * s12 + r33 * s13;
t23 = r13 * s21 + r23 * s22 + r33 * s23;
t33 = r13 * s31 + r23 * s32 + r33 * s33;
if ((t13 == 1.0) || (t13 == -1.0))  // calculate the rotation angles from the resulting matrix
{
    yaw = 0;
    double d = Math.Atan2(t12, t13);
    if (t13 == -1.0)
    {
        pitch = -Math.PI / 2;
        roll = -yaw + d;
    }
    else
    {
        pitch = Math.PI / 2;
        roll = yaw + d;
    }
}
else
{
    yaw = Math.Atan2(t12, t11);
    pitch = -Math.Asin(t13);
    roll = Math.Atan2(t23, t33);
}
yaw *= highpass;   // apply the highpass value, which means the angles
pitch *= highpass;  // try to slowly drift back to the initial zero value
roll *= highpass;

// This is the rotation matrix with the accumulated angles yaw, pitch and roll
r11 = Math.Cos(yaw) * Math.Cos(pitch);
r12 = Math.Sin(yaw) * Math.Cos(pitch);
r13 = -Math.Sin(pitch);
r23 = Math.Cos(pitch) * Math.Sin(roll);
r33 = Math.Cos(pitch) * Math.Cos(roll);

// this is the rotation matrix with the offset angles offset_yaw, offset_pitch and offset_roll
s11 = Math.Cos(offset_yaw) * Math.Cos(offset_pitch);
s12 = Math.Sin(offset_yaw) * Math.Cos(offset_pitch);
s13 = -Math.Sin(offset_pitch);
s23 = Math.Cos(offset_pitch) * Math.Sin(offset_roll);
s33 = Math.Cos(offset_pitch) * Math.Cos(offset_roll);
// multiply the matrices \( t = r \cdot s \)  
(four of the matrix elements are not required)

t11 = r11 * s11 + r21 * s12 + r31 * s13;
t12 = r12 * s11 + r22 * s12 + r32 * s13;
t13 = r13 * s11 + r23 * s12 + r33 * s13;
t23 = r13 * s21 + r23 * s22 + r33 * s23;
t33 = r13 * s31 + r23 * s32 + r33 * s33;

if (((t13 == 1.0) || (t13 == -1.0)))  
// calculate the rotation angles from the resulting matrix
{
    yaw_out = 0;
    double d = Math.Atan2(t12, t13);
    if (t13 == -1.0)
    {
        pitch_out = -Math.PI / 2;
        roll_out = -yaw_out + d;
    }
    else
    {
        pitch_out = Math.PI / 2;
        roll_out = yaw_out + d;
    }
}
else
{
    yaw_out = Math.Atan2(t12, t11);
    pitch_out = -Math.Asin(t13);
    roll_out = Math.Atan2(t23, t33);
}

// note that the time in the output file is in the middle between the timestamps of the frames
richTextBox1.AppendText(((n + 0.5) / fps).ToString("F3", invC) + " v360 yaw " + (yaw_out / Math.PI * 180.0).ToString("00.00", invC) + " v360 pitch " + (pitch_out / Math.PI * 180.0).ToString("00.00", invC) + " v360 roll " + (roll_out / Math.PI * 180.0).ToString("00.00", invC) + ":\n");
richTextBox1.ScrollToCaret();

richTextBox1.SaveFile(path + filename, RichTextBoxStreamType.PlainText);  // save the *.cmd file

public Point shift_detect(Bitmap b1, Bitmap b2, int shift_max, int size, int color)
{
    Point p = new Point(0, 0);
    int minimum = int.MaxValue;
    int[,] image1 = new int[size, size];
    int[,] image2 = new int[size, size];
int wsize = size - 1 - (2 * shift_max); // size of the compare window
for (int x = 0; x < size; x++) // copy the bitmaps to arrays, for faster access
{
    for (int y = 0; y < size; y++)
    {
        switch (color)
        {
            case 0:
                image1[x, y] = (int)b1.GetPixel(x, y).R;
                image2[x, y] = (int)b2.GetPixel(x, y).R;
                break;
            case 1:
                image1[x, y] = (int)b1.GetPixel(x, y).G;
                image2[x, y] = (int)b2.GetPixel(x, y).G;
                break;
            case 2:
                image1[x, y] = (int)b1.GetPixel(x, y).B;
                image2[x, y] = (int)b2.GetPixel(x, y).B;
                break;
        }
    }
}
for (int shift_x = 0; shift_x <= shift_max; ) // loop shift_x from -shift_max to +shift_max, but in different order
{
    for (int shift_y = 0; shift_y <= shift_max; ) // loop shift_y from -shift_max to +shift_max, but in different order
    {
        int sum = 0;
        for (int x = 0; (x < wsize) && (sum < minimum); x++) // this loop is terminated early if the sum is already larger than the
        {
            for (int y = 0; y < wsize; y++)
            {
                int p1 = image1[shift_max + x + shift_x, shift_max + y + shift_y];
                int p2 = image2[shift_max + x, shift_max + y];
                sum += Math.Abs(p1 - p2); // sum up the absolute differences
            }
        }
        if (sum < minimum) // found a better minimum?
        {
            minimum = sum;
            p.X = shift_x;
            p.Y = shift_y;
        }
    }
}
if (shift_y <= 0) // shift_y is changed in the order 0, 1, -1, 2, -2, 3, -3, ...
    shift_y--;
    shift_y = -shift_y;
if (shift_x <= 0) // shift_x is changed in the order 0, 1, -1, 2, -2, 3, -3, ...
    shift_x--;
shift_x = -shift_x;
}
return (p);
}

Form1.designer.cs
namespace v360stabilizer
{
  partial class Form1
  {
    /// <summary>
    /// Erforderliche Designervariable.
    /// </summary>
    private System.ComponentModel.IContainer components = null;
    
    /// <summary>
    /// Verwendete Ressourcen bereinigen.
    /// </summary>
    /// <param name="disposing">True, wenn verwaltete Ressourcen gelöscht werden sollen; andernfalls False.</param>
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }
    
    #region Vom Windows Form-Designer generierter Code
    
    /// <summary>
    /// Erforderliche Methode für die Designerunterstützung.
    /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
    /// </summary>
    private void InitializeComponent()
    {
      this.richTextBox1 = new System.Windows.Forms.RichTextBox();
      this.pictureBox1 = new System.Windows.Forms.PictureBox();
      this.pictureBox2 = new System.Windows.Forms.PictureBox();
      this.pictureBox3 = new System.Windows.Forms.PictureBox();
      this.pictureBox4 = new System.Windows.Forms.PictureBox();
      this.pictureBox5 = new System.Windows.Forms.PictureBox();
      this.pictureBox6 = new System.Windows.Forms.PictureBox();
      this.pictureBox7 = new System.Windows.Forms.PictureBox();
      this.pictureBox8 = new System.Windows.Forms.PictureBox();
      this.pictureBox9 = new System.Windows.Forms.PictureBox();
      this.pictureBox10 = new System.Windows.Forms.PictureBox();
    }
    #endregion
  }
}
this.pictureBox11 = new System.Windows.Forms.PictureBox();
this.pictureBox12 = new System.Windows.Forms.PictureBox();
(System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox2)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox3)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox4)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox5)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox6)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox7)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox8)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox9)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox10)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox11)).BeginInit();
(System.ComponentModel.ISupportInitialize)(this.pictureBox12)).BeginInit();
this.SuspendLayout();
//
// richTextBox1
//
this.richTextBox1.Location = new System.Drawing.Point(12, 224);
this.richTextBox1.Name = "richTextBox1";
this.richTextBox1.Size = new System.Drawing.Size(630, 332);
this.richTextBox1.TabIndex = 2;
this.richTextBox1.Text = "";
//
// pictureBox1

this.pictureBox1.Location = new System.Drawing.Point(12, 12);
this.pictureBox1.Name = "pictureBox1";
this.pictureBox1.Size = new System.Drawing.Size(100, 100);
this.pictureBox1.TabIndex = 0;
this.pictureBox1.TabStop = false;
//
// pictureBox2

this.pictureBox2.Location = new System.Drawing.Point(12, 118);
this.pictureBox2.Name = "pictureBox2";
this.pictureBox2.Size = new System.Drawing.Size(100, 100);
this.pictureBox2.TabIndex = 1;
this.pictureBox2.TabStop = false;
//
// pictureBox3

this.pictureBox3.Location = new System.Drawing.Point(118, 12);
this.pictureBox3.Name = "pictureBox3";
this.pictureBox3.Size = new System.Drawing.Size(100, 100);
this.pictureBox3.TabIndex = 4;
this.pictureBox3.TabStop = false;
//
// pictureBox4

this.pictureBox4.Location = new System.Drawing.Point(118, 118);
this.pictureBox4.Name = "pictureBox4";
this.pictureBox4.Size = new System.Drawing.Size(100, 100);
this.pictureBox4.TabIndex = 5;
this.pictureBox4.TabStop = false;
//
// pictureBox5

this.pictureBox5.Location = new System.Drawing.Point(12, 12);
this.pictureBox5.Name = "pictureBox5";
this.pictureBox5.Size = new System.Drawing.Size(100, 100);
this.pictureBox5.TabIndex = 0;
this.pictureBox5.TabStop = false;
//
// pictureBox6

this.pictureBox6.Location = new System.Drawing.Point(118, 12);
this.pictureBox6.Name = "pictureBox6";
this.pictureBox6.Size = new System.Drawing.Size(100, 100);
this.pictureBox6.TabIndex = 4;
this.pictureBox6.TabStop = false;
//
// pictureBox7

this.pictureBox7.Location = new System.Drawing.Point(12, 118);
this.pictureBox7.Name = "pictureBox7";
this.pictureBox7.Size = new System.Drawing.Size(100, 100);
this.pictureBox7.TabIndex = 1;
this.pictureBox7.TabStop = false;
//
// pictureBox8

this.pictureBox8.Location = new System.Drawing.Point(118, 118);
this.pictureBox8.Name = "pictureBox8";
this.pictureBox8.Size = new System.Drawing.Size(100, 100);
this.pictureBox8.TabIndex = 5;
this.pictureBox8.TabStop = false;
//
// pictureBox9

this.pictureBox9.Location = new System.Drawing.Point(12, 12);
this.pictureBox9.Name = "pictureBox9";
this.pictureBox9.Size = new System.Drawing.Size(100, 100);
this.pictureBox9.TabIndex = 0;
this.pictureBox9.TabStop = false;
//
// pictureBox10

this.pictureBox10.Location = new System.Drawing.Point(118, 12);
this.pictureBox10.Name = "pictureBox10";
this.pictureBox10.Size = new System.Drawing.Size(100, 100);
this.pictureBox10.TabIndex = 4;
this.pictureBox10.TabStop = false;
//
// pictureBox11

this.pictureBox11.Location = new System.Drawing.Point(12, 118);
this.pictureBox11.Name = "pictureBox11";
this.pictureBox11.Size = new System.Drawing.Size(100, 100);
this.pictureBox11.TabIndex = 1;
this.pictureBox11.TabStop = false;
//
// pictureBox12

this.pictureBox12.Location = new System.Drawing.Point(118, 118);
this.pictureBox12.Name = "pictureBox12";
this.pictureBox12.Size = new System.Drawing.Size(100, 100);
this.pictureBox12.TabIndex = 5;
this.pictureBox12.TabStop = false;
//
this.pictureBox4.Location = new System.Drawing.Point(118, 118);
this.pictureBox4.Name = "pictureBox4";
this.pictureBox4.Size = new System.Drawing.Size(100, 100);
this.pictureBox4.TabIndex = 5;
this.pictureBox4.TabStop = false;

this.pictureBox5.Location = new System.Drawing.Point(224, 12);
this.pictureBox5.Name = "pictureBox5";
this.pictureBox5.Size = new System.Drawing.Size(100, 100);
this.pictureBox5.TabIndex = 9;
this.pictureBox5.TabStop = false;

this.pictureBox6.Location = new System.Drawing.Point(224, 118);
this.pictureBox6.Name = "pictureBox6";
this.pictureBox6.Size = new System.Drawing.Size(100, 100);
this.pictureBox6.TabIndex = 8;
this.pictureBox6.TabStop = false;

this.pictureBox7.Location = new System.Drawing.Point(330, 12);
this.pictureBox7.Name = "pictureBox7";
this.pictureBox7.Size = new System.Drawing.Size(100, 100);
this.pictureBox7.TabIndex = 7;
this.pictureBox7.TabStop = false;

this.pictureBox8.Location = new System.Drawing.Point(330, 118);
this.pictureBox8.Name = "pictureBox8";
this.pictureBox8.Size = new System.Drawing.Size(100, 100);
this.pictureBox8.TabIndex = 6;
this.pictureBox8.TabStop = false;

this.pictureBox9.Location = new System.Drawing.Point(436, 12);
this.pictureBox9.Name = "pictureBox9";
this.pictureBox9.Size = new System.Drawing.Size(100, 100);
this.pictureBox9.TabIndex = 13;
this.pictureBox9.TabStop = false;

this.pictureBox10.Location = new System.Drawing.Point(436, 118);
this.pictureBox10.Name = "pictureBox10";
this.pictureBox10.Size = new System.Drawing.Size(100, 100);
this.pictureBox10.TabIndex = 12;
this.pictureBox10.TabStop = false;
this.PictureBox10.Location = new System.Drawing.Point(436, 118);
this.PictureBox10.Name = "pictureBox10";
this.PictureBox10.Size = new System.Drawing.Size(100, 100);
this.PictureBox10.TabIndex = 12;
this.PictureBox10.TabStop = false;
// pictureBox11
// this.PictureBox11.Location = new System.Drawing.Point(542, 12);
this.PictureBox11.Name = "pictureBox11";
this.PictureBox11.Size = new System.Drawing.Size(100, 100);
this.PictureBox11.TabIndex = 11;
this.PictureBox11.TabStop = false;
// pictureBox12
// this.PictureBox12.Location = new System.Drawing.Point(542, 118);
this.PictureBox12.Name = "pictureBox12";
this.PictureBox12.Size = new System.Drawing.Size(100, 100);
this.PictureBox12.TabIndex = 10;
this.PictureBox12.TabStop = false;
// Form1
// this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.ClientSize = new System.Drawing.Size(656, 568);
this.Controls.Add(this.PictureBox9);
this.Controls.Add(this.PictureBox10);
this.Controls.Add(this.PictureBox11);
this.Controls.Add(this.PictureBox12);
this.Controls.Add(this.PictureBox5);
this.Controls.Add(this.PictureBox6);
this.Controls.Add(this.PictureBox7);
this.Controls.Add(this.PictureBox8);
this.Controls.Add(this.PictureBox4);
this.Controls.Add(this.PictureBox3);
this.Controls.Add(this.RichTextBox1);
this.Controls.Add(this.PictureBox2);
this.Controls.Add(this.PictureBox1);
this.Name = "Form1";
this.Text = "Form1";
this.Shown += new System.EventHandler(this.Form1_Shown);
// ((System.ComponentModel.ISupportInitialize)(this.PictureBox1)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.PictureBox2)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.PictureBox3)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.PictureBox4)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.PictureBox5)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.PictureBox6)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.PictureBox7)).EndInit();
This is the batch file for applying the rotations to the input video:

```batch
set "IN=test.mp4"             :: Equirectangular input image
set "T=9"                     :: Duration in seconds
ffmpeg -i %IN% -lavfi sendcmd=f=stabilize.cmd,v360=e:e:pitch=0:yaw=0:roll=0 -t %T% -y out.mp4
pause
```
Suppose you have a video in which a TV or computer screen is visible, and in postprocessing you want another video to be shown on that screen. Or you have a video in which a beamer projects an image on a wall, which is almost impossible to capture flicker-free in a video. It’s better to overlay the projected image in postprocessing.

The perspective filter can be used to remap a rectangular video into the distorted screen (which is an irregular quadrangle).

The coordinates of the corners of the screen are $x_0,y_0$ (top left), $x_1,y_1$ (top right), $x_2,y_2$ (bottom left) and $x_3,y_3$ (bottom right) and must be measured in the original video.

```
set "X0=500"       :: Top left corner
set "Y0=250"
set "X1=1250"      :: Top right corner
set "Y1=150"
set "X2=400"       :: Bottom left corner
set "Y2=750"
set "X3=1150"      :: Bottom right corner
set "Y3=850"
```

rem Make a color test video

```
ffmpeg -f lavfi -i testsrc2=s=hd1080 -t 5 -y video1.mp4
```

rem Make a black and white test video

```
ffmpeg -f lavfi -i testsrc2=s=hd1080 -vf eq=saturation=0 -t 5 -y video2.mp4
```

rem Embed the black and white video into the color video

```
ffmpeg -i video1.mp4 -i video2.mp4 -lavfi "[1]format=argb,pad=w=iw+2:h=ih+2:x=1:y=1:color=black@0.0,perspective=x0=%X0%:y0=%Y0%:x1=%X1%:y1=%Y1%:x2=%X2%:y2=%Y2%:x3=%X3%:y3=%Y3%:sense=1[2];[0][2]overlay" -q:v 2 -y out.mp4
```

pause
Before I discovered the perspective filter, I thought that I had to use the remap filter for this purpose, and I figured out the formulas myself. Here they are:

The coordinates of the point to be remapped are \(x, y\).
We draw a vertical line through point \(x, y\) and calculate the intersection points with the upper and lower edge of the quadrangle:

\[a = \frac{x - x_0}{x_1 - x_0}\] The parameter \(a\) describes where the line intersects the upper edge. For \(a=0\) it's at the top left corner, for \(a=1\) it's at the top right corner. For \(0<a<1\) the intersection point is somewhere between these two corners. But there are also cases possible \(a<0\) or \(a>1\) where the intersection point is outside the finite line segment.
The intersection point is \(x_4,y_4\)

\[x_4 = x\]
\[y_4 = y_0 + a \times (y_1 - y_0)\]

We do the same thing for the lower edge:

\[b = \frac{x - x_2}{x_3 - x_2}\]
\[x_5 = x\]
\[y_5 = y_2 + b \times (y_3 - y_2)\]

Parameter \(c\) describes where the point \(x,y\) lies on the line segment between points 4 and 5:

\[c = \frac{y - y_4}{y_5 - y_4}\]

Now we remap these points into a unit quadrat with the top left corner at 0,0:
Point 4 is at coordinates \(a,0\) and point 5 is at coordinates \(b,1\)
Point \(x,y\) is remapped to coordinates

\[x_{map} = (a + c \times (b - a))\]
\[y_{map} = c\]
2.87 Image warping with displace filter

set "W=751" :: Width of image
set "H=853" :: Height of image
set "CX=347" :: X center of distortion
set "CY=451" :: Y center of distortion
set "A=15" :: Maximum amplitude of displacement, positive displaces outwards and negative inwards, allowed range is [0..127], best values are below 20
set "D=30" :: Radius from center of distortion, where the maximum displacement occurs

rem Create the displace_x file
ffmpeg -flavfi -i nullsrc=size=WxH -vf format=pix_fmts=gray8,geq='st(0,2*A*D/(pow((CX-X),2)+pow((CY-Y),2)+D*D));128-ld(0)*(X-CX)' -frames 1 -y displace_x.pgm

rem Create the displace_y file
ffmpeg -f lavfi -i nullsrc=size=WxH -vf format=pix_fmts=gray8,geq='st(0,2*A*D/(pow((CX-X),2)+pow((CY-Y),2)+D*D));128-ld(0)*(Y-CY)' -frames 1 -y displace_y.pgm

rem Apply the displace filter to the image
ffmpeg -i me.jpg -i displace_x.pgm -i displace_y.pgm -lavfi format=pix_fmts=rgb24,displace -frames 1 -y bigger_nose.jpg

set "A=-15" :: Now let's try the other sign

rem Create the displace_x file
ffmpeg -f lavfi -i nullsrc=size=WxH -vf format=pix_fmts=gray8,geq='st(0,2*A*D/(pow((CX-X),2)+pow((CY-Y),2)+D*D));128-ld(0)*(X-CX)' -frames 1 -y displace_x.pgm

rem Create the displace_y file
ffmpeg -f lavfi -i nullsrc=size=WxH -vf format=pix_fmts=gray8,geq='st(0,2*A*D/(pow((CX-X),2)+pow((CY-Y),2)+D*D));128-ld(0)*(Y-CY)' -frames 1 -y displace_y.pgm

rem Apply the displace filter to the image
ffmpeg -i me.jpg -i displace_x.pgm -i displace_y.pgm -lavfi format=pix_fmts=rgb24,displace -frames 1 -y smaller_nose.jpg

pause
Here is the input image and the two output images:

It might be dangerous to use this kind of processing for images of women without prior asking them for permission :-)

The "displace" filter expects mapping files with relative values in the range [0..255], where 128 is the neutral value for no displacement. Larger displacements than 127 pixels aren't possible.

I recommend to set the format to rgb24 before using the displace filter.
This is an example for enlarging the eyes:

```
set "W=751"        :: Width of image
set "H=853"        :: Height of image
set "LX=256"       :: left eye x
set "LY=362"       :: left eye y
set "RX=445"       :: right eye x
set "RY=325"       :: right eye y
set "A=10"         :: Maximum amplitude of displacement, positive displaces outwards and negative inwards,
                     :: allowed range is [0..127], best values are below 20
set "D=25"         :: Radius from center of distortion, where the maximum displacement occurs

rem Create the displace_x file
ffmpeg -flavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray8,geq='st(0,2*A*D/(pow((%LX-X),2)+pow((%LY-Y),2)+
D*D));st(1,2*A*D/(pow((%RX-X),2)+pow((%RY-Y),2)+D*D));128-ld
(0)*(X-%LX)-ld(1)*(X-%RX)' -frames 1 -y displace_x.pgm

rem Create the displace_y file
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=pix_fmts=gray8,geq='st(0,2*A*D/(pow((%LX-X),2)+pow((%LY-Y),2)+
D*D));st(1,2*A*D/(pow((%RX-X),2)+pow((%RY-Y),2)+D*D));128-ld
(0)*(Y-%LY)-ld(1)*(Y-%RY)' -frames 1 -y displace_y.pgm

rem Apply the displace filter to the image or video
ffmpeg -i me.jpg -i displace_x.pgm -i displace_y.pgm -lavfi format=pix_fmts=rgb24,displace -frames 1 -y big_eyes.jpg
pause
```

If the output is a video, remove "-frames 1" in the last command line.
Here are the input and output images:

me.jpg

big_eyes.jpg
Mathematics for this distortion:

\[ d = \frac{2ADr}{r^2 + D^2} \]

\[ r = \sqrt{(x-cx)^2 + (y-cy)^2} \]

\[ \frac{d}{r} = \frac{2AD}{(x-cx)^2 + (y-cy)^2 + D^2} \]

\[ dx = (x-cx) \frac{d}{r} \quad dy = (y-cy) \frac{d}{r} \]

with \( d \) = displacement distance

A = maximum amplitude of displacement

r = distance from pixel x,y to center of distortion cx, cy

D = distance where the largest displacement occurs

cx, cy = coordinates of center of the distortion

dx, dy = displacement values
### 2.88 Noise reduction

FFmpeg has several filters for video noise reduction:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
<th>Notes and Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>atadenoise</td>
<td>Apply an Adaptive Temporal Averaging Denoiser to the video input</td>
<td>very fast, temporal only with no motion compensation; LGPL     Example: atadenoise=0a=0.2:1a=0.2:2a=0.2:0b=0.3:1b=0.3:2b=0.3</td>
</tr>
<tr>
<td>bm3d</td>
<td>Denoise frames using Block-Matching 3D algorithm</td>
<td>very very slow, currently implemented as spatial only, algorithm considered as one of the state of art denoisers; LGPL</td>
</tr>
<tr>
<td>chromanr</td>
<td>Reduce chrominance noise</td>
<td>This filter calculates the absolute difference of the Y components (contrary to the official documentation! ) of the current pixel and a neighbour pixel from a rectangular neighbourhood. Absolute differences are also calculated for the U and V components. A neighbour pixel is used for averaging, if the sum of all three absolute differences is lower than the threshold. Only the U and V components are averaged. The Y component remains unchanged. With the &quot;stepw&quot; and &quot;steph&quot; options it's possible to use only a subset of the neighbour pixels for averaging.</td>
</tr>
<tr>
<td>dctdnoiz</td>
<td>Denoise frames using 2D DCT (frequency domain filtering)</td>
<td>very very slow: spatial only, blurs too much; LGPL</td>
</tr>
<tr>
<td>fftdenoiz</td>
<td>Denoise frames using 3D FFT (frequency domain filtering)</td>
<td>slow, spatial and limited temporal, using Fast Fourier Transform, may have introduce ringing with bad settings; LGPL</td>
</tr>
<tr>
<td>hqdn3d</td>
<td>This is a high precision/quality 3d denoise filter. It aims to reduce image noise, producing smooth images and making still images really still. It should enhance compressibility.</td>
<td>fast, both spatial and temporal, does basically lowpass by destroying high frequencies, blurs with extreme settings; GPL  Example: hqdn3d=4:4:9:9</td>
</tr>
<tr>
<td>nlmeans</td>
<td>Denoise frames using Non-Local means algorithm</td>
<td>very slow, currently implemented as spatial only, algorithm considered as one of the state of art denoisers; LGPL</td>
</tr>
<tr>
<td>owdenoise</td>
<td>Apply Overcomplete Wavelet denoiser</td>
<td>very very very slow, spatial only, wavelet; LGPL  Example: owdenoise=ls=25</td>
</tr>
<tr>
<td>removegrain</td>
<td>Spatial denoiser for progressive video</td>
<td>fast, spatial only, limited usecase</td>
</tr>
<tr>
<td>vaguenoiser</td>
<td>Apply a wavelet based denoiser</td>
<td>slow, spatial only, limited usecase</td>
</tr>
<tr>
<td>tmix</td>
<td>Noise reduction by averaging up to 128 successive frames</td>
<td>Not suitable for moving objects. Example: tmix=frames=20</td>
</tr>
<tr>
<td>tmedian</td>
<td>Noise reduction by calculating the median out of up to 127 successive frames</td>
<td>Example: tmedian=radius=20</td>
</tr>
</tbody>
</table>

Special thanks to Paul B Mahol who posted most of these notes in the FFmpeg-user list on October 27, 2019
2.89 -filter_complex_script

The complex filtergraph can be loaded from an external script file. Line feeds and empty lines are allowed in the script file. This makes the script much more readable. The drawback is that you can't use variables as in a batch file.

2.90 Time delay within a filter chain

This is an example for a time delay within a filter chain:

```
ffmpeg -f lavfi -i testsrc=duration=10:size=vga -filter_complex split[a][b];[a]setpts=PTS-5/TB[c];[b][c]hstack=shortest=1 -y out.mp4
```

Hint: Subtracting a constant from PTS works fine. However if you try to add a constant to PTS (e.g. setpts=PTS+5/TB), this may lead to the problem that the true length of the output video isn't equal to the length in the metadata.

In this example the same thing is done with tpad and trim filters:

```
ffmpeg -f lavfi -i testsrc=duration=10:size=vga -filter_complex split[a][b];[a]tpad=start_duration=5[c];[b][c]hstack=shortest=1,trim=start=5,setpts=PTS-5/TB -y out.mp4
```
2.91 Recording one-time events with pre-trigger

Let's assume you want to record a video of a one-time event and you don't know when this event will happen. You get a trigger signal when the event starts, but in the video you also want to see 10 seconds before the trigger signal. This can be realized by capturing the camera signal continuously, delaying it by 10 seconds and then streaming it to a UDP address. This is the first process which is running continuously:

```plaintext
set "SIZE=640x480" :: Size of camera frames
set "FR=30" :: Framerate of camera
set "DELAY=10" :: Delay time in seconds
ffmpeg -f dshow -video_size %SIZE% -vcodec mjpeg -framerate %FR% -i video="BisonCam,NB Pro":audio="Mikrofon (Realtek(R) Audio)" -lavfi 
[0:0]tpad=%FR%*%DELAY%,format=yuv420p;[0:1]adelay=%DELAY%s" -q:v 2 -f mpegts udp://127.0.0.1:1234
```

Note: This doesn't work if you omit the audio channel.

Note: The argument of the "adelay" filter is normally in milliseconds, but you can also specify seconds if you add "s", as in this example.


The first process should be running for at least 10 seconds. Then the second process is started when the event occurs. It saves a 20 second video, which consists of 10 seconds before and 10 seconds after the trigger signal:

```plaintext
set "T=20" :: Duration in seconds
ffmpeg -f mpegts -i udp://127.0.0.1:1234 -t %T% -y out.mp4
```

For testing the stream you can use this batch file:

```plaintext
ffmpeg -f mpegts -i udp://127.0.0.1:1234 -f sdl2 -
```

See also https://trac.ffmpeg.org/wiki/Capture/Lightning (that's a complicated solution with external software)
2.92 Chroma subsampling, pixel formats of images or videos

When you make a video from many JPG images, all images must have the same pixel format. But sometimes they are different. For example I has many images that came from the camera with 4:2:2 pixel format. But I had to edit one of the images with IrfanView, it then it was saved with pixel format 4:2:0.

This example changes the pixel format of an image from 4:2:0 to 4:2:2:

```bash
ffmpeg -i IMG_044x.jpg -pix_fmt yuvj422p -q 0 IMG_044.jpg
```

Set the pixel format of a video to 4:4:4 and scale the video to 1920x1080:

```bash
ffmpeg -i input.MOV -pix_fmt yuv444p -s 1920x1080 out.mov
```

In a filter chain the format can be set as follows:

```bash
ffmpeg -i input.MOV -lavfi "format=pix_fmts=rgb24" -y out.mp4
```

Which is the same as:

```bash
ffmpeg -i input.MOV -lavfi "format=rgb24" -y out.mp4
```

<table>
<thead>
<tr>
<th>Chroma subsampling</th>
<th>8-bit format</th>
<th>10-bit format</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:4:4</td>
<td>yuv444p</td>
<td>yuv444p10le</td>
<td>Each of the three Y'CbCr components have the same sample rate, thus there is no chroma subsampling. This scheme is sometimes used in high-end film scanners and cinematic post production.</td>
</tr>
</tbody>
</table>
Note that "4:4:4" may instead be referring to R'G'B' color space, which implicitly also does not have any chroma subsampling.

<table>
<thead>
<tr>
<th>Format</th>
<th>Bit depth</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:2:2</td>
<td>yuv422p</td>
<td>yuv422p10le</td>
</tr>
<tr>
<td>4:2:0</td>
<td>yuv420p</td>
<td>yuv420p10le</td>
</tr>
</tbody>
</table>


RGB and gray pixel formats (this is only a subset of the available formats):

<table>
<thead>
<tr>
<th>Format</th>
<th>NB COMPONENTS</th>
<th>BITS_PER_PIXEL</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb24, bgr24</td>
<td>3</td>
<td>24</td>
<td>packed</td>
</tr>
<tr>
<td>gbrp</td>
<td>3</td>
<td>24</td>
<td>planar</td>
</tr>
<tr>
<td>gray</td>
<td>1</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>argb, rgba, abgr, bgra</td>
<td>4</td>
<td>32</td>
<td>packed</td>
</tr>
<tr>
<td>gray16be, gray16le</td>
<td>1</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>rgb48be, rgb48le, bgr48be, bgr48le</td>
<td>3</td>
<td>48</td>
<td>packed</td>
</tr>
<tr>
<td>brgp16be, brgp16le</td>
<td>3</td>
<td>48</td>
<td>planar</td>
</tr>
<tr>
<td>gray14be, gray14le</td>
<td>1</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

All pixel formats are described in the source code in the folder libavutil in file pixfmt.h.

You can also print out the list of all pixel formats with this command:

```bash
ffmpeg -pix_fmts
```
List of all pixel formats:

<table>
<thead>
<tr>
<th>FLAGS</th>
<th>NAME</th>
<th>NB_COMPONENTS</th>
<th>BITS_PER_PIXEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.....</td>
<td>yuv420p</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>I.....</td>
<td>yuv422p</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb24</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>I.....</td>
<td>bgr24</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>I.....</td>
<td>yuv422p</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>yuv444p</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>I.....</td>
<td>gray</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>I.....</td>
<td>monow</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I.....</td>
<td>monob</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I.....</td>
<td>pal8</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>I.....</td>
<td>yuvj420p</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>I.....</td>
<td>yuvj422p</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>yuvj444p</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>I.....</td>
<td>uyvy422</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>uyvy4411</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>I.....</td>
<td>bgr8</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>O.....</td>
<td>bgr4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb4_byte</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>O.....</td>
<td>rgb8</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>O.....</td>
<td>rgb4_byte</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>I.....</td>
<td>nv12</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>I.....</td>
<td>nv21</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>I.....</td>
<td>argb</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>I.....</td>
<td>rgba</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>I.....</td>
<td>abgr</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>I.....</td>
<td>gray16be</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>gray16le</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>yuv440p</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>yuv444p</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>yuva420p</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb48be</td>
<td>3</td>
<td>48</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb48le</td>
<td>3</td>
<td>48</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb565be</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb565le</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb555be</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>I.....</td>
<td>rgb555le</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>I.....</td>
<td>bgr565be</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>bgr565le</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>I.....</td>
<td>bgr555be</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>I.....</td>
<td>bgr555le</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>..H..</td>
<td>vaapi_moco</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>..H..</td>
<td>vaapi_idct</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>..H..</td>
<td>vaapi_vid</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Codec</td>
<td>Width</td>
<td>Height</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>yuv420p16le</td>
<td>3</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>yuv420p16be</td>
<td>3</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>yuv422p16le</td>
<td>3</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>yuv422p16be</td>
<td>3</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>yuv444p16le</td>
<td>3</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>yuv444p16be</td>
<td>3</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>dxva2_vld</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>rgb444le</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>rgb444be</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>bgr444le</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>bgr444be</td>
<td>3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>ya8</td>
<td>2</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>bgr48be</td>
<td>3</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>yuv420p9be</td>
<td>3</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>yuv420p9le</td>
<td>3</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>yuv420p10be</td>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>yuv420p10le</td>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>yuv422p10be</td>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>yuv422p10le</td>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>yuv444p9be</td>
<td>3</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>yuv444p9le</td>
<td>3</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>yuv444p10be</td>
<td>3</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>yuv444p10le</td>
<td>3</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>yuv422p9le</td>
<td>3</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>gbrp</td>
<td>3</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>gbrp9be</td>
<td>3</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>gbrp9le</td>
<td>3</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>gbrp10be</td>
<td>3</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>gbrp10le</td>
<td>3</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>gbrp16be</td>
<td>3</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>gbrp16le</td>
<td>3</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>yuva422p</td>
<td>4</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>yuva444p</td>
<td>4</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>yuva420p9be</td>
<td>4</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>yuva420p9le</td>
<td>4</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>yuva422p9be</td>
<td>4</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>yuva422p9le</td>
<td>4</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>yuva444p9be</td>
<td>4</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>yuva444p9le</td>
<td>4</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>yuva420p10be</td>
<td>4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>yuva420p10le</td>
<td>4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>yuva422p10be</td>
<td>4</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>yuva422p10le</td>
<td>4</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>yuva444p10be</td>
<td>4</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>yuva444p10le</td>
<td>4</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>yuva420p16be</td>
<td>4</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>yuva422p16be</td>
<td>4</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>yuva444p16be</td>
<td>4</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>vdpau</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>xyz12le</td>
<td>3</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>xyz12be</td>
<td>3</td>
<td>36</td>
<td></td>
</tr>
</tbody>
</table>

222
<table>
<thead>
<tr>
<th>Format</th>
<th>Channels</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>nv16</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>nv20le</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>nv20be</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>rgba64be</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>rgba64le</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>bgra64be</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>bgra64le</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>yvyu422</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>ya16be</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>ya16le</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>gbrap</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>gbrap16be</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>gbrap16le</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>qsv</td>
<td></td>
<td></td>
</tr>
<tr>
<td>smai</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d3d11va_vld</td>
<td></td>
<td></td>
</tr>
<tr>
<td>cuda</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0rgb</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>rgb0</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>0bgr</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>yuv420p12be</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>yuv420p12le</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>yuv420p14be</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>yuv420p14le</td>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>yuv422p12be</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>yuv422p12le</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>yuv422p14be</td>
<td>3</td>
<td>28</td>
</tr>
<tr>
<td>yuv422p14le</td>
<td>3</td>
<td>28</td>
</tr>
<tr>
<td>yuv44p12be</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>yuv44p12le</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>yuv44p14be</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>yuv44p14le</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>gbrp12be</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>gbrp12le</td>
<td>3</td>
<td>36</td>
</tr>
<tr>
<td>gbrp14be</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>gbrp14le</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>yuvj411p</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>bayer_bgr8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bayer_bgr8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bayer_bgr8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bayer_bgr8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bayer_bgr16le</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>bayer_bgr16be</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>bayer_rggb16le</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>bayer_rggb16be</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>bayer_gbrg16le</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>bayer_gbrg16be</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>bayer_gbrg16le</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>bayer_gbrg16be</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>xvmc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0rgb</td>
<td>3</td>
<td>64</td>
</tr>
<tr>
<td>yuv440p10be</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>yuv440p12le</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>yuv440p12be</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>ayuv64le</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Length</td>
<td>Width</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>ayuv64be</td>
<td>4</td>
<td>64</td>
</tr>
<tr>
<td>videotoolbox_vld</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p010le</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>p010be</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>gbrap12be</td>
<td>4</td>
<td>48</td>
</tr>
<tr>
<td>gbrap12le</td>
<td>4</td>
<td>48</td>
</tr>
<tr>
<td>gbrap10be</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>gbrap10le</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>mediaacodec</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gray12be</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>gray12le</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>gray10be</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>gray10le</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>p016le</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>p016be</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>d3d11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gray9be</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>gray9le</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>gbrpf32be</td>
<td>3</td>
<td>96</td>
</tr>
<tr>
<td>gbrpf32le</td>
<td>3</td>
<td>96</td>
</tr>
<tr>
<td>gbrapf32be</td>
<td>4</td>
<td>128</td>
</tr>
<tr>
<td>gbrapf32le</td>
<td>4</td>
<td>128</td>
</tr>
<tr>
<td>d3m_prime</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>opencl</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gray14be</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>gray14le</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>gray32be</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>gray32le</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>yuva422p12be</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>yuva422p12le</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>yuva444p12be</td>
<td>4</td>
<td>48</td>
</tr>
<tr>
<td>yuva444p12le</td>
<td>4</td>
<td>48</td>
</tr>
<tr>
<td>nv24</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>nv42</td>
<td>3</td>
<td>24</td>
</tr>
<tr>
<td>vulkan</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>y210be</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>y210le</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>x2rgb10le</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>x2rgb10be</td>
<td>3</td>
<td>30</td>
</tr>
</tbody>
</table>
Automatic format conversions can be disabled by the option "-noauto_conversion_filters". Use "-v verbose" for checking where FFmpeg did auto-insert format conversions. Search for the green lines in the console listing.

You can also add "graphmonitor=f=format". This output is shown as text overlaid to the video.

If you are using the "-noauto_conversion_filters" option, you must manually insert the required conversions in the filter chain.

Example without "-noauto_conversion_filters":

```
ffmpeg -v verbose -f lavfi -i testsrc2=s=svga:d=5,format=yuv422p10le -vf
lut3d="VLog_to_V709.cube" -pix_fmt yuv422p10le -c:v h264 -y out.mov
```

In this example you can see in the console listing that FFmpeg did auto-insert two format conversions: Before the lut3d filter from yuv422p10le to rgb48le, and after the lut3d filter from rgb48le to yuv422p10le.

The same example with "-noauto_conversion_filters":

```
ffmpeg -v verbose -f lavfi -i testsrc2=s=svga:d=5,format=yuv422p10le -vf
scale,format=rgb48le,lut3d="VLog_to_V709.cube",scale -noauto_conversion_filters -pix_fmt yuv422p10le -c:v h264 -y out.mov
```

As you can see, there are also two "scale" filters required. It's hard to understand why. In this case the second format conversion can be omitted because it's redundant with the following "-pix_fmt" option.

The following explanation was written by Gyan Doshi in the FFmpeg user list on September 14, 2020:

"Each filter presents a list of input formats they can work with and a list of output formats they can directly generate. The framework inspects adjacent filters and sets a compatible common format for the outputs and inputs when possible. If not, it sets one of the available output formats for the preceding filter and one from input formats for the following filter and inserts a scale filter to convert between those. This process is format negotiation. The format filter doesn't carry out the conversion itself - it inserts scale which in turn invokes libswscale. scale without any args defaults to the source W and H. But for pixel formats, its output format is constrained by the following format filter. That triggers a format conversion by libswscale."
### 2.94 Convert RGB to HSV or HSL

\[
\begin{align*}
\text{MAX} & := \max(R,G,B), \quad \text{MIN} := \min(R,G,B) \\
\text{H} & := 0 \quad \text{if} \quad \text{MAX} = \text{MIN} \\
\text{H} & := 60^\circ \times \left(0 + \frac{G - B}{\text{MAX} - \text{MIN}}\right) \quad \text{if} \quad \text{MAX} = \text{R} \\
\text{H} & := 60^\circ \times \left(2 + \frac{B - R}{\text{MAX} - \text{MIN}}\right) \quad \text{if} \quad \text{MAX} = \text{G} \\
\text{H} & := 60^\circ \times \left(4 + \frac{R - G}{\text{MAX} - \text{MIN}}\right) \quad \text{if} \quad \text{MAX} = \text{B} \\
\text{if} \quad \text{H} < 0 \quad \text{then} \quad \text{H} & := \text{H} + 360^\circ \\
S_{\text{HSV}} & := 0 \quad \text{if} \quad \text{MAX} = 0 \\
S_{\text{HSV}} & := \frac{\text{MAX} - \text{MIN}}{\text{MAX}} \quad \text{if} \quad \text{MAX} 
eq 0 \\
\text{V} & := \text{MAX} \\
S_{\text{HSL}} & := 0 \quad \text{if} \quad \text{MAX} = 0 \\
S_{\text{HSL}} & := 0 \quad \text{if} \quad \text{MIN} = 0 \\
S_{\text{HSL}} & := \frac{\text{MAX} - \text{MIN}}{1 - \mid \text{MAX} + \text{MIN} - 1 \mid} \quad \text{if} \quad \text{MAX} 
eq 0 \text{ and } \text{MIN} 
eq 0
\end{align*}
\]

Please note that this formula works only if the RGB values are in the [0..1] range. For the [0..255] range, replace "1" by "255".

\[
\text{L} := \frac{\text{MAX} + \text{MIN}}{2}
\]

This is a different method for calculating hue and chroma:

\[
\begin{align*}
\text{HUE} & := \text{atan2}(\sqrt{3} \times (G - B), 2 \times R - G - B) \\
\text{CHROMA} & := \sqrt{(3 \times (G - B)^2 + (2 \times R - G - B)^2)}
\end{align*}
\]

<table>
<thead>
<tr>
<th>Color</th>
<th>HUE</th>
<th>R</th>
<th>G</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>0°</td>
<td>255</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>orange</td>
<td>30°</td>
<td>255</td>
<td>128</td>
<td>0</td>
</tr>
<tr>
<td>yellow</td>
<td>60°</td>
<td>255</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>green-yellow</td>
<td>90°</td>
<td>128</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>green</td>
<td>120°</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>blue-green</td>
<td>150°</td>
<td>0</td>
<td>255</td>
<td>128</td>
</tr>
<tr>
<td>cyan</td>
<td>180°</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>green-blue</td>
<td>210°</td>
<td>0</td>
<td>128</td>
<td>255</td>
</tr>
<tr>
<td>blue</td>
<td>240°</td>
<td>0</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>violet</td>
<td>270°</td>
<td>128</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>magenta</td>
<td>300°</td>
<td>255</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>magenta</td>
<td>330°</td>
<td>255</td>
<td>0</td>
<td>128</td>
</tr>
<tr>
<td>white</td>
<td>360°</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>black</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
2.95 Convert HSV to RGB

Given is HUE \([0..360°]\) and \(S,V [0..1]\)

\[h_i := \lfloor\text{HUE} / 60°\rfloor\]
\[f := \text{HUE} / 60° - h_i\]
\[p := V \times (1 - S)\]
\[q := V \times (1 - S \times f)\]
\[t := V \times (1 - S \times (1 - f))\]

\((R,G,B) := (V,t,p)\) if \(h_i = 0\) (from red to yellow)
\((R,G,B) := (q,V,p)\) if \(h_i = 1\) (from yellow to green)
\((R,G,B) := (p,V,t)\) if \(h_i = 2\) (from green to cyan)
\((R,G,B) := (p,q,V)\) if \(h_i = 3\) (from cyan to blue)
\((R,G,B) := (t,p,V)\) if \(h_i = 4\) (from blue to magenta)
\((R,G,B) := (V,p,q)\) if \(h_i = 5\) (from magenta to red)
### Video Codecs

| `-c:v mpeg4` | This is the older MP4 codec. Search for "libxvid" in the documentation.  
See also [https://trac.ffmpeg.org/wiki/Encode/MPEG-4](https://trac.ffmpeg.org/wiki/Encode/MPEG-4)  
See also [http://www.ffmpeg.org/ffmpeg-codecs.html#libxvid](http://www.ffmpeg.org/ffmpeg-codecs.html#libxvid) |
| `-c:v libx264` | Newer codec with better compression than mpeg4, but it's possible that the videos don't play on older computers.  
See also [https://trac.ffmpeg.org/wiki/Encode/H.264](https://trac.ffmpeg.org/wiki/Encode/H.264)  
See also [http://www.ffmpeg.org/ffmpeg-codecs.html#libx264](http://www.ffmpeg.org/ffmpeg-codecs.html#libx264) |
See also [http://www.ffmpeg.org/ffmpeg-codecs.html#libx265](http://www.ffmpeg.org/ffmpeg-codecs.html#libx265) |
| `-c:v prores_ks` | Apple ProRes encoder, example:  
`ffmpeg -i input.MOV -vcodec prores_ks -pix_fmt yuva444p10le -profile:v 4444 -bits_per_mb 8000 -s 1920x1080 out.mov`  
See also [http://www.ffmpeg.org/ffmpeg-codecs.html#ProRes](http://www.ffmpeg.org/ffmpeg-codecs.html#ProRes)  
See also [https://trac.ffmpeg.org/wiki/Encode/VFX](https://trac.ffmpeg.org/wiki/Encode/VFX) |
| `-c:v dnxhd` | This codec is suitable for converting 10-bit videos from GH5S camera into a format that's readable by the free DaVoince Resolve software. There isn't much documentation available for this codec and its options. Example:  
`ffmpeg -i input.mov -map metadata 0 -pix_fmt yuv422p10le -c:v dnxhd -profile:v 4 -c:a pcm_s24le -color_range pc -movflags write_colr out.mov`  
See also [https://www.pixeltools.com/rate_control_paper.html](https://www.pixeltools.com/rate_control_paper.html)  
See also [https://trac.ffmpeg.org/wiki/Encode/VFX](https://trac.ffmpeg.org/wiki/Encode/VFX) |

The constant rate factor (CRF) can be set with the `-crf` parameter. Use this mode if you want to keep the best quality and don't care about the file size. The CRF range 0–51 for 8-bit x264 and 0-63 for 10-bit. 0 is lossless, 23 is the default, and 51 is worst quality possible.

This is in more detail explained here: [https://www.pixeltools.com/rate_control_paper.html](https://www.pixeltools.com/rate_control_paper.html)

Use a preset with the `-preset` parameter. Possible options are ultrafast, superfast, veryfast, faster, fast, medium (this is the default), slow, slower and veryslow. A preset is a collection of options that will provide a certain encoding speed to compression ratio. A slower preset will provide better compression. Use the slowest preset that you have patience for.
The -tune parameter can be set to these options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>film</td>
<td>use for high quality movie content; lowers deblocking</td>
</tr>
<tr>
<td>animation</td>
<td>good for cartoons; uses higher deblocking and more reference frames</td>
</tr>
<tr>
<td>grain</td>
<td>preserves the grain structure in old, grainy film material</td>
</tr>
<tr>
<td>stillimage</td>
<td>good for slideshow-like content</td>
</tr>
<tr>
<td>fastdecode</td>
<td>allows faster decoding by disabling certain filters</td>
</tr>
<tr>
<td>zerolatency</td>
<td>good for fast encoding and low-latency streaming</td>
</tr>
</tbody>
</table>

List all possible internal presets and tunes:

```bash
ffmpeg -hide_banner -f lavfi -i nullsrc -c:v libx264 -preset help -f mp4 -
```
2.97 Metadata

Global metadata can be saved in a text file as follows:

```
ffmpeg -i input.mp4 -f ffmetadata metadata.txt
```

If you also need the metadata from the video and audio streams (which may contain more informations), use this:

```
ffmpeg -i input.mp4 -c copy -map_metadata 0 -map_metadata:s:v 0:s:v -map_metadata:s:a 0:s:a -f ffmetadata metadata.txt
```

The metadata can be re-inserted into a file as follows:

```
ffmpeg -i input.mp4 -i metadata.txt -map_metadata 1 -codec copy output.mp4
```

Write metadata "title" to mp4 video without re-encoding:

```
ffmpeg -i input.mp4 -metadata title="This is the Title" -acodec copy -codec copy -copyts output.mp4
```

Unfortunately FFmpeg can't insert the metadata that is required for a spherical 360° video.

2.98 Video filters "copy" and "null"

These filters are only for testing, for example when you want to disable part of a filter chain.

The "null" filter does really nothing, the output is the same as the input.

The "copy" filter copies the old frame and deletes the old one. The output is the same as with the "null" filter.
2.99 Re-numbering images

Cameras do normally create images with 4-digit numbers. When the counter (in Canon cameras) overflows, the number changes from 9999 to 0001. That means the number 0000 is missing and the numbers aren't continuously increasing, as it's expected by FFmpeg. This problem can be solved with this sequence of console commands:

```
ren IMG_1*.* IMG_2*.*
ren IMG_0*.* IMG_1*.*
ren IMG_9*.* IMG_0*.*
```

The first two commands add 1000 to those numbers that begin with 0 or 1. The last command subtracts 9000 from those numbers that begin with 9. Now the images are in increasing order, but one image is still missing between images 0999 and 1001. This isn't a problem for FFmpeg because is does automatically search the next image. That means small gaps are allowed in the numbering. The maximum gap size is 5 by default. This can be changed with the parameter `-start_number_range`. 
## 2.100 Filenames for images

Image filenames can contain variables. Please note that in a Windows batch file the % character must be replaced by two %% characters.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>%nd</td>
<td>A sequential n-digit number</td>
<td>The start number can be specified with the -start_number option. The default value is 1.</td>
</tr>
<tr>
<td>%0nd</td>
<td>A sequential number that is 0-padded to n digits</td>
<td></td>
</tr>
<tr>
<td>%Y</td>
<td>Year</td>
<td>Only available if the “-strftime 1” option is used. Example for embedding date and time in the filename: Screenshot-%Y-%m-%d-%H-%M-%S.jpg</td>
</tr>
<tr>
<td>%m</td>
<td>Month</td>
<td></td>
</tr>
<tr>
<td>%d</td>
<td>Day</td>
<td></td>
</tr>
<tr>
<td>%H</td>
<td>Hours</td>
<td></td>
</tr>
<tr>
<td>%M</td>
<td>Minutes</td>
<td></td>
</tr>
<tr>
<td>%S</td>
<td>Seconds</td>
<td></td>
</tr>
<tr>
<td>%d</td>
<td>PTS of the frame (Presentation time stamp)</td>
<td>Only available if the “-frame_pts 1” option is used. Example for embedding the PTS in the filename: Screenshot-%d.jpg</td>
</tr>
</tbody>
</table>
2.101 Make spectrum images

This command line creates an image with a spectrum red - green - blue - red. In the middle between red and green is 50% red and 50% green, which is not a bright yellow.

```bash
ffmpeg -f lavfi -i nullsrc=s=768x256 -vf geq=r='lt(X,256)*clip(256-X,0,255)+clip(X-512,0,255)'+g='lt(X,256)*clip(X,0,255)+gte(X,256)*clip(512-X,0,255)'+b='lt(X,512)*clip(X-256,0,255)+gte(X,512)*clip(768-X,0,255)',oscilloscope=tw=1:s=1 -frames 1 -y spectrum.png
```

The same thing can also be done with the "gradients" video source:

```bash
ffmpeg -f lavfi -i gradients=s=768x256:c0=red:c1=lime:c2=blue:c3=red:nb_colors=4:x0=0:y0=0:x1=767:y1=0 -vf oscilloscope=tw=1:s=1 -frames 1 -y spectrum.png
```

Note: "green" is 0x008000, "lime" is 0x00FF00

This command line creates an image with a spectrum yellow - cyan - magenta - yellow:

```bash
ffmpeg -f lavfi -i gradients=s=768x256:c0=yellow:c1=cyan:c2=magenta:c3=yellow:nb_colors=4:x0=0:y0=0:x1=767:y1=0 -vf oscilloscope=tw=1:s=1 -frames 1 -y spectrum1.png
```

This command line creates an image with a spectrum red - yellow - green - cyan - blue - magenta. Because this is an optical spectrum, the last part from magenta to red is omitted. In the middle between red and green is 100% red and 100% green, which is bright yellow.

```bash
ffmpeg -f lavfi -i nullsrc=s=1280x256 -vf geq=r='clip(512-X,0,255)+clip(X-1024,0,255)'+g='lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255)'+b='lt(X,1024)*clip(X,256,0,255)+gte(X,512)*clip(1536-X,0,255)',oscilloscope=tw=1:s=1 -frames 1 -y spectrum2.png
```
The same thing can also be done with the "gradients" video source:

```bash
ffmpeg -f lavfi -i gradients=s=1280x256:c0=red:c1=yellow:c2=lime:c3=cyan:c4=blue:c5=magenta:nb_colors=6:x0=0:y0=0:x1=1279:y1=0 -vf oscilloscope=tw=1:s=1 -frames 1 -y spectrum2.png
```

Note: "green" is 0x008000, "lime" is 0x00FF00

This command line creates an image with a spectrum red - yellow - green - cyan - blue - magenta - red. In the middle between red and green is 100% red and 100% green, which is bright yellow.

```bash
ffmpeg -f lavfi -i nullsrc=s=1536x256 -vf geq=r='clip(512-X,0,255)+clip(X-1024,0,255)':g='lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255)':b='lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255)',oscilloscope=tw=1:s=1 -frames 1 -y spectrum3.png
```

Create a full saturation spectrum at the bottom with a gradient to black at the top:

```bash
ffmpeg -f lavfi -i nullsrc=s=1536x256:c0=red:c1=yellow:c2=lime:c3=cyan:c4=blue:c5=magenta:c6=red:nb_colors=7:x0=0:y0=0:x1=1535:y1=0 -vf oscilloscope=tw=1:s=1 -frames 1 -y spectrum4.png
```

```bash
ffmpeg -f lavfi -i nullsrc=s=1536x256 -vf geq=r='st(0,clip(512-X,0,255)+clip(X-1024,0,255));Y*ld(0)/255':g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));Y*ld(0)/255':b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));Y*ld(0)/255' -frames 1 -y spectrum4.png
```
Create a full saturation spectrum at the top with a gradient to black at the bottom:

```bash
ffmpeg -f lavfi -i nullsrc=s=1536x256 -vf geq=r='st(0,clip(512-X,0,255)+clip(X-1024,0,255));(255-Y)*1d(0)/255';
g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));(255-Y)*1d(0)/255';
b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));(255-Y)*1d(0)/255' -frames 1 -y spectrum5.png
```

Create a full saturation spectrum at the top with a gradient to white at the bottom:

```bash
ffmpeg -f lavfi -i nullsrc=s=1536x256 -vf geq=r='st(0,clip(512-X,0,255)+clip(X-1024,0,255));X+1d(0)*(1-Y/255)';
g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));X+1d(0)*(1-Y/255)';
b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));X+1d(0)*(1-Y/255)' -frames 1 -y spectrum6.png
```

Create a full saturation spectrum at the bottom with a gradient to white at the top:

```bash
ffmpeg -f lavfi -i nullsrc=s=1536x256 -vf geq=r='st(0,clip(512-X,0,255)+clip(X-1024,0,255));255+Y*(ld(0)/255-1)';
g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));255+Y*(ld(0)/255-1)';
b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));255+Y*(ld(0)/255-1)' -frames 1 -y spectrum7.png
```

Create a full saturation spectrum in the middle with a gradient to white at the top and a gradient to black at the bottom:

```bash
ffmpeg -f lavfi -i nullsrc=s=1536x512 -vf geq=r='st(0,clip(512-X,0,255)+clip(X-1024,0,255));if(lt(Y,256),255+Y*(ld(0)/255-1),(511-Y)*1d(0)/255)';
g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));if(lt(Y,256),255+Y*(ld(0)/255-1),(511-Y)*1d(0)/255)';
b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));if(lt(Y,256),255+Y*(ld(0)/255-1),(511-Y)*1d(0)/255)' -frames 1 -y spectrum8.png
```

```bash
pause
```
2.102  Redshift

It's possible to simulate a relativistic redshift, like in astronomy when a galaxy is moving away from the observer at a high speed.

In this case the colors must be modified as follows:

- black will remain black
- red is shifted towards black
- yellow is shifted towards red
- green is shifted towards yellow
- cyan is shifted towards green
- blue is shifted towards cyan
- magenta is shifted towards blue
- white is shifted towards yellow

This can't be realized by a hue rotation, because hue would rotate red towards magenta and blue (which physically makes no sense).

But this can be realized by a simple 3D look-up table which contains only 8 colors, which are the corners of the RGB cube. The table is saved in *.cube format. The order of the components in the *.cube file is red, green, blue.

redshift.cube

```
LUT 3D_SIZE 2
# black remains black
0 0 0
# red becomes black
0 0 0
# green becomes yellow
1 1 0
# yellow becomes red
1 0 0
# blue becomes cyan
0 1 1
# magenta becomes blue
0 0 1
# cyan becomes green
0 1 0
# white becomes yellow
1 1 0
```
This batch file will create a test spectrum (including black and white) and apply the above look-up table:

rem Create a test spectrum (including black and white):
ffmpeg -f lavfi -i nullsrc=s=1536x512 -vf geq=r='st(0,clip(512-X,0,255)+clip(X-
1024,0,255));if(lt(Y,256),255+Y*(ld(0)/255-1),(511-Y)*ld(0)/255)';g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-
X,0,255));if(lt(Y,256),255+Y*(ld(0)/255-1),(511-Y)*ld(0)/255)';b='st(0,lt(X,1024)*clip(X-
512,0,255)+gte(X,1024)*clip(1536-X,0,255));if(lt(Y,256),255+Y*(ld(0)/255-1),(511-Y)*ld(0)/255)'; -frames 1 -y
spectrum.png

rem Apply the look-up table and use vstack to show the output below the input:
ffmpeg -i spectrum.png -filter_complex split[a][b],[a]lut3d="redshift.cube"[c],[b][c]vstack -y out.png

pause

Some notes for redshift in astronomy:
delta lambda = lambda * v / c  z = (lambda_obs / lambda_emission) - 1  z = v / c  (approximation for v << c)
2.103 Make many JPG test images

```bash
set "N=6"         :: Number of frames
set "S=400x300"  :: Size
ffmpeg -f lavfi -i testsrc2=size=%S%:duration=%N%:rate=1 -y test%%3d.jpg
```

Note: The default start number is 1. If you want to begin the filenames with "test000.jpg", add "-start_number 0" before the output filename.

2.104 Make a grid video

```bash
set "G=10"         :: Grid size
set "T=9"          :: Grid thickness
set "C=white"      :: Grid color
set "B=black"      :: Background color
set "S=800x600"    :: Video size
set "D=10"         :: Duration in seconds

rem Make a grid video

ffmpeg -f lavfi -i color=%B%:s=%S% -vf drawgrid=w=%G%:h=%G%:t=%T%:c=%C% -t %D% -y grid.mp4
```

Note: If thickness is almost as large as grid size, the result becomes a grid of dots.
2.105  Make a chessboard video

```
ffmpeg -f lavfi -i color=black:s=vga -vf geq=lum='255*mod(floor(X/40)+floor(Y/40),2):cr=128' -t 5 -y out.mp4
```

2.106  Make a test video with audio

```
rem  Make a 6 seconds video with 1kHz tone
ffmpeg -f lavfi -i testsrc2=size=vga -f lavfi -i sine=1000 -t 6 -y video.mp4
```
2.107 Make a 10-bit test video with audio

```
rem Make a 6 seconds 10-bit video with 1kHz tone
ffmpeg -f lavfi -i testsrc2=size=hd1080,format=yuv422p10le -f lavfi -i sine=1000 -c:v h264 -t 6 -y out.mov
```

2.108 Test a filtergraph for 10-bit compatibility

Sometimes it's important to know if a filtergraph has 10-bit accuracy, or if there are hidden 8-bit conversions somewhere in the filtergraph. This can be tested with a test video which contains 1024 levels of gray in a 32x32 grid:

```
rem Make a 10-bit video with 32*32=1024 gray levels:
ffmpeg -f lavfi -i nullsrc=s=svga,format=yuv444p10le -vf geq=lum='X/W*32+32*floor(Y/H*32)':cr=512:cb=512 -crf 10 -c:v h264 -t 5 -y 10bit.mov

rem Now apply the filtergraph that you want to test:
.....

rem Now analyze the result by applying a strong contrast enhancement:
ffmpeg -i 10bit.mov -vf eq=contrast=32 -crf 10 -c:v h264 -y test.mov
```

Now check if the output video has 32 gray levels in X direction. If there was a hidden 8-bit conversion, then there would be only 8 levels.
2.109  Find an object in a video and hide it

The find_rect function can find a rectangular object in an image or video, and cover_rect can then replace it by another image or by interpolating the neighbor pixels.

```bash
ffmpeg -i test1.png -vf find_rect=test2.pgm,cover_rect out.png
```

Note: find_rect will find only one instance of the object in a frame.

If you already know the coordinates and size of the rectangular object to be covered, then instead of "find_rect" it's better to use these alternatives:

- If you want to interpolate the surrounding pixels, use the "delogo" filter.
- If you want to draw a rectangle with a uniform color, use the "drawbox" filter.
- If you want to replace the region by another image, use the "overlay" filter.

2.110  Find the coordinates of a moving object in a video

The find_rect filter can also be used for searching an object in a video and write the x,y coordinates to a file. This works only with FFprobe and not with FFmpeg:

```bash
set "IN=in.mp4" :: Input video
set "OBJ=needle.pgm" :: Image of the object, must be gray8
set "TH=0.8" :: Threshold, 0.01 = only exact matches, 0.99 = almost everything matches
set "XMIN=900" :: Minimum x position of the object's top left corner
set "XMAX=1900" :: Maximum x position of the object's top left corner
set "YMIN=490" :: Minimum y position of the object's top left corner
set "YMAX=510" :: Maximum y position of the object's top left corner
```
ffprobe -f lavfi movie=%IN%,find_rect=object=%OBJ%:threshold=%TH%:xmin=%XMIN%:xmax=%XMAX%:ymin=%YMIN%:ymax=%YMAX%-show_entries frame=pkt_pts_time:frame_tags=lavfi.rect.x,lavfi.rect.y -of csv 1> log.csv

Note: To speed up the algorithm, make the object image as small as possible and use a search window with the xmin, xmax, ymin, ymax options.

This is the resulting logfile. The coordinates are for the top left corner of the object image. If no object was found for the specified threshold, no coordinates are written:

frame,0.000000
frame,0.040000
frame,0.080000
frame,0.120000,45,1
frame,0.160000,45,1
frame,0.200000,45,1
frame,0.240000,45,1
...

If you want to omit the word "frame," in the log file, replace "csv" by "csv=p=0"

ffprobe -f lavfi movie=%IN%,find_rect=object=%OBJ%:threshold=%TH% -show_entries frame=pkt_pts_time:frame_tags=lavfi.rect.x,lavfi.rect.y -of csv=p=0 1> log.csv

Note: A list of variables that can be printed with "-show_entries" can be found in the file "ffprobe.xsd" which is in the source tree under doc/ffprobe.xsd

2.111 Detect black frames and replace them by previous frame

See https://video.stackexchange.com/q/23589/

See "blackframe" and "blackdetect" filters.
## Image formats

FFmpeg supports these image formats. This list is incomplete, a longer (but also incomplete) list can be found at [http://www.ffmpeg.org/general.html](http://www.ffmpeg.org/general.html)

<table>
<thead>
<tr>
<th>Image Format</th>
<th>compressed</th>
<th>lossless</th>
<th>16-Bit?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMP</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>very big file size</td>
</tr>
<tr>
<td>DNG</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>&quot;Adobe Digital Negative&quot; format, recommended for saving RAW images from cameras. Use Adobe DNG Converter to make these files. <strong>Warning:</strong> FFmpeg's DNG decoder doesn't work correctly with most images. See next chapter.</td>
</tr>
<tr>
<td>FITS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>Flexible Image Transport System, a popular image format in astronomy</td>
</tr>
<tr>
<td>GIF</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>this is obsolete, use PNG instead</td>
</tr>
<tr>
<td>JPG</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>recommended for 8-bit images if small file size is required</td>
</tr>
<tr>
<td>PAM</td>
<td>no</td>
<td>?</td>
<td>?</td>
<td>&quot;Portable Arbitrary Map&quot;</td>
</tr>
<tr>
<td>PGM</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>&quot;Portable Graymap&quot;, these files are required for the remap filter. FFmpeg can read binary PGM (P5) files and ASCII PGM (P2) files, but for output only binary PGM (P5) is supported. PGM files contain values in the range [0..65535]. Negative values aren't possible, but FFmpeg gives no warning if a negative number is found in a P2 file.</td>
</tr>
<tr>
<td>PGMYUV</td>
<td>no</td>
<td>?</td>
<td>?</td>
<td>This is a FFmpeg variant of the binary PGM format</td>
</tr>
<tr>
<td>PNG</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>recommended for lossless saving of 8-bit or 16-bit images</td>
</tr>
<tr>
<td>PPM</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>&quot;Portable Pixmap&quot;</td>
</tr>
<tr>
<td>TGA</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>this is obsolete, use PNG instead</td>
</tr>
<tr>
<td>TIFF</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

Show all supported pixel formats of the PNG encoder:

```
c:\ffmpeg\ffmpeg -h encoder=png
```
2.113 DNG Images

FFmpeg's DNG decoder has many problems and fails with most DNG images.

This is the best workaround I could find so far, for FFmpeg DNG (from Canon 6D via Adobe DNG converter V12.4) to JPG conversion:

```
ffmpeg -i IMG_3459.dng -vf
colorchannelmixer=rr=1.0:gg=0.5:bb=1.0,tonemap=linear:param=1000,tonemap=gamma:param=1.3,crop=x=96:y=60:w=iw-96:h=ih-60
-y out.jpg
```

Note: I think "colorchannelmixer=gg=0.5" is a dirty workaround for decoding the bayer_rggb16le pixel format. There are two green pixels in a macropixel, that's why the image appears too green and thus the green channel must be divided by 2.

Note: A problem of FFmpeg's DNG decoder is that the size of the image may be a little bit too large. If there are black borders at the top and left, you can remove them with the "crop" filter. In this example the size is for an image from a Canon 6D (5472x3648).

Note: Depending on the color temperature at which the picture was taken, the "rr" and "bb" parameters may have to be adjusted. If the image appears too reddish, make "rr" smaller than 1.0 and "bb" bigger than 1.0, and if it's too bluish, then the other way round. It should be possible to automatize this process using the metadata from the DNG image, but I don't know how to do this.

Recommended workarounds for DNG images:

- If 8-bit is sufficient, use IrfanView to convert the images to PNG or JPG.
- If 16-bit is required, use GIMP with Darktable plugin to convert the images to 16-bit PNG.

It was claimed that FFmpeg can correctly decode DNG images (from Adobe DNG converter V12.4). But I've never seen a working example.

Some more hints how it might be possible to decode a DNG image:

- Use zscale filter. Well, with which options, with which other filters and/or format conversions, in which order?
- Use the right input pixel format prior to calling zscale as it does not support bayer formats. Well, but which is the right input pixel format?
According to FFprobe, the input pixel format is bayer_rggb16le. How can the pixel format be set? With "-pix_fmt" before the input? Or after the input? Or with "-pixel_format"? Or with "format=" at the beginning of the filter chain? None of these four methods works.

- Set output trc. Well, what is "output trc"? Is "trc" a synonym for "transfer characteristic"? At which point in the filter chain must it be set? Before the zscale filter or in the zscale filter or after the zscale filter? To which value must it be set? How can it be specified that the input color space is sRGB?

I decided to give up at this point, because without proper documentation it's impossible to find a working example in reasonable time. There are too many open questions and the try-and-error approach is frustrating. I did already spend 20 hours on this problem. If anybody has found a working example, please let me know.


The "zimg" (zscale) library can be found here, but don't expect any useful documentation: [https://github.com/sekrit-twc/zimg](https://github.com/sekrit-twc/zimg)
2.114 Video sizes

This list contains only the most important video sizes. The full list is in the FFmpeg documentation.

<table>
<thead>
<tr>
<th>Size</th>
<th>Aspect ratio</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntsc</td>
<td>720x480</td>
<td>3:2</td>
</tr>
<tr>
<td>pal</td>
<td>720x576</td>
<td>5:4</td>
</tr>
<tr>
<td>vga</td>
<td>640x480</td>
<td>4:3</td>
</tr>
<tr>
<td>svga</td>
<td>800x600</td>
<td>4:3</td>
</tr>
<tr>
<td>xga</td>
<td>1024x768</td>
<td>4:3</td>
</tr>
<tr>
<td>uxga</td>
<td>1600x1200</td>
<td>4:3</td>
</tr>
<tr>
<td>wuxga</td>
<td>1920x1200</td>
<td>8:5 Beamer in the planetarium in the St. Andreasberg observatory</td>
</tr>
<tr>
<td>hd720</td>
<td>1280x720</td>
<td>16:9</td>
</tr>
<tr>
<td>hd1080</td>
<td>1920x1080</td>
<td>16:9</td>
</tr>
<tr>
<td>2k</td>
<td>2048x1080</td>
<td>256:135 = 17.066:9</td>
</tr>
<tr>
<td>2kdcii</td>
<td>2048x1080</td>
<td>256:135 = 17.066:9</td>
</tr>
<tr>
<td>2kflat</td>
<td>1998x1080</td>
<td>37:20 = 16.65:9 Cinema advertising</td>
</tr>
<tr>
<td>4k</td>
<td>4096x2160</td>
<td>256:135 = 17.066:9</td>
</tr>
<tr>
<td>4kdcii</td>
<td>4096x2160</td>
<td>256:135 = 17.066:9</td>
</tr>
<tr>
<td>4kflat</td>
<td>3996x2160</td>
<td>37:20 = 16.65:9</td>
</tr>
<tr>
<td>uhd2160</td>
<td>3840x2160</td>
<td>16:9</td>
</tr>
</tbody>
</table>
2.115 Editing videos from PanoView XDV360 fulldome camera

rem Editing videos from the chinese PanoView XDV360 fulldome camera

set "SIZE=1200" :: Video size (square)
set "QU=2" :: MP4 quality level, 0 is best quality, 2 is normal, 9 is strong compression
set "FPS=30" :: Output Framerate
set "IN=PanoView.mp4" :: Input video
set "START=0.5" :: Start time in seconds in the input video
set "LEN=4" :: Length in seconds in the input video
set "FADE=1" :: Fade-In and Fade-Out duration in seconds
set "FO=3.5" :: Start time of Fade-Out (Start + Length - Fade)
:: 185° field of view: 2168:2168:144:144
:: 190° field of view: 2224:2224:116:116
:: 195° field of view: 2280:2280:88:88
:: 200° field of view: 2336:2336:60:60
set "CON=1.0" :: Contrast in range [-1000 ... 1000], normal is 1.0
set "BR=0.0" :: Brightness in range [-1.0 ... 1.0], normal is 0.0
set "SA=1.0" :: Saturation in range [0.0 ... 3.0], normal is 1.0
set "GA=1.0" :: Gamma in range [0.1 ... 10.0], normal is 1.0
set "HUE=25" :: Color correction, negative towards red, positive towards blue
set "SOUND=birds.mp3" :: Audio input file
set "VOL=2.0" :: Audio volume, normal is 1.0
set "OUT=out.mp4" :: Output filename

ffmpeg -i %IN% -i Circle_3648.png -i %SOUND% -ss %START% -t %LEN% -filter_complex crop=%CR
% ,scale=3648:3648,overlay,hue=h=%HUE%,eq=contrast=%CON%:brightness=%BR%:saturation=%SA%:gamma=%GA%,fade=in:st=%START%:d=%FADE%,fade=out:st=%FO%:d=%FADE% -ac 2 -af volume=%VOL%,aresample=44100,afade=in:st=%START%:d=%FADE%,afade=out:st=%FO%:d=%FADE% -s %SIZE%x%SIZE% -c:v mpeg4 -q:v %QU% -y %OUT%

pause

The image "Circle_3648.png" has the size 3648x3648 and contains a circular mask. The color of the circle content is declared transparent, and the border is black. "-ac 2" expands the audio tracks to stereo, and "-af aresample=44100" increases the audio sampling rate to 44100.
2.116 Editing videos from the Kodak SP360_4K camera

The Kodak SP360_4K camera is better than the PanoView XDV360 because it has a slightly higher resolution and the videos are not compressed as much.

```plaintext
rem Manipulate a video from KODAK SP360_4K camera (size 2880x2880)

set "SIZE=1200" :: Video size (square)
set "QU=2" :: MP4 quality level, 0 is best quality, 2 is normal, 9 is strong compression
set "FPS=30" :: Output Framerate
set "INPUT=102_0001.MP4" :: Input video
set "START=5" :: Start time in seconds in the input video
set "LENGTH=28" :: Length in seconds in the input video
set "FADE=1" :: Fade-In and Fade-Out duration in seconds
set "FADEOUT=32" :: Start time of Fade-Out (Start + Length - Fade)
                                :: 185° field of view: 2528:2528:174:174
                                :: 190° field of view: 2592:2592:142:142
                                :: 195° field of view: 2664:2664:106:106
                                :: 200° field of view: 2728:2728:74:74

set "CONTRAST=1.0" :: Contrast in range [-1000 ... 1000], normal is 1.0
set "BRIGHT=0.0" :: Brightness in range [-1.0 ... 1.0], normal is 0.0
set "SATUR=1.0" :: Saturation in range [0.0 ... 3.0], normal is 1.0
set "GAMMA=1.0" :: Gamma in range [0.1 ... 10.0], normal is 1.0
set "HUE=0" :: Color correction, negative towards red, positive towards blue
set "SOUND=vogelstimmen.mp3" :: Audio file
set "VOL=0.4" :: Audio volume, normal is 1.0
set "OUTPUT=scene30.mp4" :: Output filename

ffmpeg -i %INPUT% -i Circle_3648.png -ss %START% -t %LENGTH% ^
-filter_complex crop=%CROP%,scale=3648:3648,overlay,hue=h=%HUE%,eq=contrast=%CONTRAST%:brightness=%BRIGHT%:^saturation=%SATUR%:gamma=%GAMMA%,fade=in:st=%START%:d=%FADE%,fade=out:st=%FADEOUT%:d=%FADE% ^
-af volume=%VOL%,aresample=44100,afade=in:st=%START%:d=%FADE%,afade=out:st=%FO%:d=%FADE% ^
-s %SIZE%x%SIZE% -c:v mpeg4 -q:v %QU% -y %OUT%
```

pause
2.117 Postprocessing of real time videos of the night sky

```plaintext
set "DARK=Dark.mov" :: Dark video
set "IN=sternschnuppe175.mov" :: Input video
set "AMP=0.06" :: 1 / Gain factor
set "D=0.3" :: Parameter for atadenoise filter
set "TMIX=25" :: Tmix number, more than 25 isn't required
set "BR=0.05" :: Small brightness increase after dark subtraction
set "NR=0.8" :: Noise reduction in maximum function
set "OUT=175.mp4" :: Output video

text Create a darkframe by averaging many frames from the dark video

ffmpeg -i %DARK% -vf "crop=2824:ih,pad=iw:2824:-1:-1,scale=1200:1200,curves=all='0/0 %AMP%/1 1/1',eq=saturation=0,tmix=frames=128" -frames 1 -y dark.png

text create a video with dark subtraction, strong contrast enhancement and denoise filter

ffmpeg -i %IN% -i dark.png -filter_complex "crop=2824:ih,pad=iw:2824:-1:-1,scale=1200:1200,curves=all='0/0 %AMP%/1 1/1',eq=saturation=0[a];[a][1]blend=subtract,eq=brightness=%BR%,split=2[b][c];[b]atadenoise=0a=%D%:1a=%D%:2a=%D%:0b=%D%:1b=%D%:2b=%D%:[c]tmix=%TMIX%[e];[d][e]blend=all_expr='max(%NR%*A,B)'" -q:v 1 -y %OUT%

Better version:

set "DARKVID=Dark.mov" :: Dark video
set "MAKEDARK=no" :: Make a darkframe yes / no
::
set "IN=CUT000534.mov" :: Input video
set "OUT=meteor534.mp4" :: Output video
::
set "BP_R=0.00" :: Black point red, positive value makes background darker
```

249
set "BP_G=0.00"   :: Black point green, positive value makes background darker
set "BP_B=0.00"   :: Black point blue, positive value makes background darker
::
set "WP=0.12"     :: White point
:: Make sure that all pixel values in the dark frame
:: are below this white point
::
set "SAT=0.0"     :: Saturation, normal = 1.0, set to 0 for monochrome
set "GAM=1.0"     :: Gamma, normal = 1.0
::
set "D=0.3"       :: Parameter for Atadenoise filter
set "TMIX=25"     :: Tmix count, more than 25 isn't required
::
set "CUT=12"      :: Duration that is cut off from the beginning
:: In the input video the duration from the beginning
:: to the first appearance of the meteor must be greater
:: than the duration of the output video
::
set "AUDIO=yes"   :: Copy audio yes / no
set "AUD="
if %AUDIO%==no (set AUD=-an)

if %MAKEDARK%==no goto VIDEO

ffmpeg -i %DARKVID% -vf "crop=2824:ih, pad=iw:2824:-1:-1, scale=1200:1200, tmix=128, format=rgb48" -frames 1 -y dark.png
::VIDEO
ffmpeg -ss %CUT% -i %IN% -y soundtrack.wav
ffmpeg -i %IN% -i dark.png -i soundtrack.wav -filter_complex "crop=2824:ih, pad=iw:2824:-1:-1, scale=1200:1200, format=rgb48[a];[a][1]blend=subtract, colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%, eq=saturation=%SAT%:gamma=%GAM%, split[b][c];[b]setpts=PTS-%CUT%/TB, atadenoise=0a=%D%:1a=%D%:2a=%D%:0b=%D%:1b=%D%:2b=%D%[d];[d][e]blend=lighten:shortest=1" -map 2:a -q:v 1 -c:v mpeg4 %AUD% -y %OUT%
pause
This batch file does the same postprocessing with a loop over all CUT*.MOV files in the current folder:

```bash
set "BP_R=0.02"                 :: Black point red, positive value makes background darker
set "BP_G=0.00"                 :: Black point green, positive value makes background darker
set "BP_B=0.02"                 :: Black point blue, positive value makes background darker
set "WP=0.2"                    :: White point
                                :: Make sure that all pixel values in the dark frame
                                :: are below this white point
set "SAT=1.0"                   :: Saturation, normal = 1.0, set to 0 for monochrome
set "GAM=1.0"                   :: Gamma, normal = 1.0
set "D=0.3"                     :: Parameter for Atadenoise filter,
                                :: 0.3 = strongest noise reduction
set "TMIX=25"                   :: Tmix count, more than 25 isn't required
set "CUT=15"                    :: Duration that is cut off from the beginning
                                :: In the input video the duration from the beginning
                                :: to the first appearance of the meteor must be greater
                                :: than the duration of the output video
set "AUDIO=no"                  :: Copy audio yes / no
                                :: Make sure that all pixel values in the dark frame
                                :: are below this white point
set AUD=""                      :: Copy audio yes / no
if %AUD%==no (set AUD=-an)

for %%f in (CUT*.MOV) do call :for_body %%f
goto :the_end

:for_body
ffmpeg -i %1 -i dark.png -filter_complex "crop=2824:ih,pad=iw:2824:-1:-1,scale=1200:1200,format=rgb48[a];[a]
[1]blend=subtract,colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%,eq=saturation=%SAT%
gamma=%GAM%,split[b][c][b]setpts=PTS-%CUT%/TB,atadenoise=0a=%D%:1a=%D%:2a=%D%:0b=%D%:1b=%D%:2b=%D%[d][c]tmix=%TMIX%
[e][d][e]blend=lighten:shortest=1" -q:v 1 -c:v mpeg4 %AUD% -y %~n1.mp4
exit /b

:the_end
pause
```
Workflow for night sky videos with GH5S

Workflow for videos from the starry sky with Panasonic GH5S and Nippon Kogaku 8mm Fisheye:

Step 1, apply gradation curve and then extract an image and insert the CLUT:

```bash
set "IN=P1000128.mov" :: Input video
set "T=1" :: Time where image is extracted
ffmpeg -ss %T% -i %IN% -f lavfi -i haldclutsrc=8 -filter_complex ":format=pix_fmts=rgb48be[a];[b]xstack=inputs=2:layout=0_0|w0_0" -frames 1 -y image_with_clut.png

pause
```

Step 2, after editing with GIMP, the CLUT is extracted and applied to the video:

```bash
set "IN=P1000128.mov" :: Input video
set "OUT=128.mp4" :: Output video
set "BR=0.04" :: Small brightness adjustment before applying the CLUT
ffmpeg -i processed_image_with_clut.png -vf crop=512:512:0:0 -y clut.png
ffmpeg -i %IN% -i clut.png -filter_complex ":format=pix_fmts=rgb48be,crop=2824:ih,pad=iw:2824:-1:-1,eq=brightness=%BR%,scale=1200:1200[a],[a][1]haldclut" -an -y %OUT%

del clut.png

pause
```
2.119 Combine many options and filters

Of course, you can also combine any number of options and filters in a single batch file, as in this example. With this batch file you can cut a temporal part out of a video, change width and height, adjust the frame rate, change the speed (slow motion or time lapse), if necessary crop to square format, if necessary remove the original sound, and change contrast, brightness, saturation and gamma.

```plaintext
set "INPUT=PanoView.mp4" :: Input video  
set "OUTPUT=out.mp4"      :: Output video  
set "SIZE=800x800"        :: Width and height of output video; the aspect ratio should be the same as in the  
                            :: input video, or square if QUAD=yes was selected  
set "RATE=30"             :: Output framerate  
set "START=1.0"           :: Start time in seconds (in input video)  
set "LENGTH=3"            :: Length in seconds (in input video)  
set "SPEED=3.0"           :: Speed factor: < 1 timelapse, 1 real time, > 1 slow motion  
set "QUAD=no"             :: no: keep the aspect ratio as-is  
                          :: yes: crop to square aspect ratio  
set "AUDIO=no"            :: no: suppress sound  
                          :: yes: keep the original sound (with unchanged speed)  
set "CONTRAST=1.0"        :: Contrast in range [-1000 ... 1000], normal is 1.0  
set "BRIGHT=0.0"          :: Brightness in range [-1.0 ... 1.0], normal is 0.0  
set "SATUR=1.0"           :: Saturation in range [0.0 ... 3.0], normal is 1.0  
set "GAMMA=1.0"           :: Gamma in range [0.1 ... 10.0], normal is 1.0  
set "QU=2"                :: MP4 Quality, 0 is best Quality, 2 is normal, 9 is strongest compression

set CROP=iw:ih
if %QUAD%==yes (set CROP=ih:ih)

set SOUND=
if %AUDIO%==no (set SOUND=-an)

ffmpeg -ss %START% -t %LENGTH% -i %INPUT% %SOUND% ^  
-vf crop=%CROP%,setpts=%SPEED%*PTS,eq=contrast=%CONTRAST%:brightness=%BRIGHT%:saturation=%SATUR%:gamma=%GAMMA% ^  
-s %SIZE% -r %RATE% -q:v %QU% -codec:v mpeg4 %OUTPUT%
```

pause
2.120  Timelapse example with masking, deflicker, rotation, fading

This batch file creates a time lapse from many images, with masking and deflicker filter, with slow rotation of the image, fade in and fade out at the beginning and end:

```batch
set "IN=IMG_%%4d.jpg"    :: Input pictures
set "SN=3551"            :: Number of first picture
set "SIZE=1200"          :: Video size (square)
set "QU=2"               :: MP4 quality level, 0 is best quality, 2 is normal, 9 is strong compression
set "FPS=30"             :: Output Framerate
set "FADE=3"             :: Fade-In and Fade-Out duration in seconds
set "FADEOUT=11.5"       :: Start time of Fade-Out (Length - Fade)
set "CONTRAST=1.0"       :: Contrast in range [-1000 ... 1000], normal is 1.0
set "BRIGHT=0"           :: Brightness in range [-1.0 ... 1.0], normal is 0.0
set "SATUR=1.2"          :: Saturation in range [0.0 ... 3.0], normal is 1.0
set "GAMMA=1.1"          :: Gamma in range [0.1 ... 10.0], normal is 1.0
set "ROT=0.0+n*0.002"    :: Rotation angle in radians
set "DEF=20"             :: Deflicker frames
set "AUDIO=birds.mp3"    :: Audio file
set "VOL=1.5"            :: Audio volume
set "OUT=out.mp4"        :: Output filename

rem  A is the duration in seconds how long a single image is shown (without crossfade duration), here: 0.2
rem  B is the crossfade duration in seconds, here: 0.2
set "D=2"    :: enter (A+B)/B   D=1: 100% fade   D=2: 50% fade   D=4: 25% fade
set "F=5"    :: enter 1/B
set "E=13"   :: enter FADE * F - D (longer duration for first and last picture)
set "L=30"   :: Number of pictures

ffmpeg -start_number %SN% -i %IN% -i Circle_3648.png -i %AUDIO% -shortest ^
-filter_complex crop=ih:ih,scale=3648:3648,eq=contrast=%CONTRAST%:brightness=%BRIGHT%:saturation=%SATUR%:gamma=%GAMMA%,^
framerate=%FPS%:interp_start=0:interp_end=255:scene=100,rotate=%ROT%,deflicker=size=%DEF%,^
fade=in:d=%FADE%:fa=0:out=st=%FADEOUT%:d=%FADE% ^
-af volume=%VOL%,afade=in:st=0:d=%FADE%,afade=out:st=%FADEOUT%:d=%FADE% -codec:v mpeg4 -q:v %QU% -y %OUT%
pause
```

254
2.121 Slow motion with Panasonic GH5S at 240fps

Set "Rec Format" to "MOV" and "Rec Quality" to one of these:

<table>
<thead>
<tr>
<th>System Frequency</th>
<th>Rec Quality</th>
<th>Available Framerates</th>
</tr>
</thead>
<tbody>
<tr>
<td>59.94Hz (NTSC)</td>
<td>[FHD/8bit/100M/60p]</td>
<td>2 30 56 58 60 62 64 90 120 150 180 210 240</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/30p]</td>
<td>2 15 26 28 30 32 34 45 60 75 90 105 120 135 150 165 180 195 210 225 240</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60 72 84 96 108 120 132 144 156 168 180 192 204 216 228 240</td>
</tr>
<tr>
<td>50.00Hz (PAL)</td>
<td>[FHD/8bit/100M/50p]</td>
<td>2 25 46 48 50 52 54 75 100 125 150 200 240</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/25p]</td>
<td>2 12 21 23 25 27 30 37 50 62 75 87 100 112 125 137 150 175 200 225 240</td>
</tr>
<tr>
<td>24.00Hz (CINEMA)</td>
<td>[FHD/8bit/100M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60 72 84 96 108 120 132 144 156 168 180 192 204 216 228 240</td>
</tr>
</tbody>
</table>

Set "Variable Frame Rate" to "On" and set the desired framerate.

The video can be played as-is with VLC player and is already a timelapse. The speed factor can be calculated as follows:

\[
\text{Speed\_Factor} = \frac{\text{FrameRate}}{\text{Base\_FrameRate}}
\]

where Base\_FrameRate is either 24, 25, 30, 50 or 60 as specified in "Rec Quality".

If you want a higher speed factor, you have to use the setpts filter:

```bash
set "IN=P1000128.mov" :: Input video
set "S=5.0" :: Slow motion factor
ffmpeg -i %IN% -vf setpts=5*PTS -y out.mp4
pause
```

Example:

The video was taken in [FHD/8bit/100M/50p] mode at 240fps. It has already a 4.8x speed factor if played as-is.

Output from FFprobe: FrameRate = 50, VFRatio = 50/240

With the additional factor 5 the total speed factor becomes 24x, with other words one second in reality is shown as 24 seconds in playback. The output framerate can be set with the -r option if required; this doesn't affect the speed factor.

255
Table for setpts value (for videos taken at 240fps):

<table>
<thead>
<tr>
<th>Base Framerate</th>
<th>2x</th>
<th>2.4x</th>
<th>2.5x</th>
<th>3x</th>
<th>4x</th>
<th>4.8x</th>
<th>5x</th>
<th>6x</th>
<th>8x</th>
<th>9.6x</th>
<th>10x</th>
<th>12x</th>
<th>16x</th>
<th>19.2x</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>0.2</td>
<td>0.24</td>
<td>0.25</td>
<td>0.3</td>
<td>0.4</td>
<td>0.48</td>
<td>0.5</td>
<td>0.6</td>
<td>0.8</td>
<td>0.96</td>
<td>1</td>
<td>1.2</td>
<td>1.6</td>
<td>1.92</td>
</tr>
<tr>
<td>25</td>
<td>0.20833</td>
<td>0.25</td>
<td>0.26042</td>
<td>0.3125</td>
<td>0.41667</td>
<td>0.5</td>
<td>0.52083</td>
<td>0.625</td>
<td>0.8333</td>
<td>1</td>
<td>1.0417</td>
<td>1.25</td>
<td>1.6667</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>0.25</td>
<td>0.3</td>
<td>0.3125</td>
<td>0.375</td>
<td>0.5</td>
<td>0.6</td>
<td>0.625</td>
<td>0.75</td>
<td>1</td>
<td>1.2</td>
<td>1.25</td>
<td>1.5</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>50</td>
<td>0.41667</td>
<td>0.5</td>
<td>0.52208</td>
<td>0.625</td>
<td>0.83333</td>
<td>1</td>
<td>1.0417</td>
<td>1.3021</td>
<td>1.6667</td>
<td>2</td>
<td>2.0833</td>
<td>2.5</td>
<td>3.3333</td>
<td>4</td>
</tr>
<tr>
<td>60</td>
<td>0.5</td>
<td>0.6</td>
<td>0.625</td>
<td>0.75</td>
<td>1</td>
<td>1.2</td>
<td>1.25</td>
<td>1.5</td>
<td>2</td>
<td>2.4</td>
<td>2.5</td>
<td>3</td>
<td>4</td>
<td>4.8</td>
</tr>
<tr>
<td>Output fps</td>
<td>120</td>
<td>100</td>
<td>96</td>
<td>80</td>
<td>60</td>
<td>50</td>
<td>48</td>
<td>40</td>
<td>30</td>
<td>25</td>
<td>24</td>
<td>20</td>
<td>15</td>
<td>12.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Base Framerate</th>
<th>20x</th>
<th>24x</th>
<th>25x</th>
<th>30x</th>
<th>40x</th>
<th>48x</th>
<th>50x</th>
<th>60x</th>
<th>80x</th>
<th>96x</th>
<th>100x</th>
<th>120x</th>
<th>160x</th>
<th>192x</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>2</td>
<td>2.4</td>
<td>2.5</td>
<td>3</td>
<td>0.4</td>
<td>4.8</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>9.6</td>
<td>10</td>
<td>12</td>
<td>16</td>
<td>19.2</td>
</tr>
<tr>
<td>30</td>
<td>2.5</td>
<td>3</td>
<td>3.125</td>
<td>3.75</td>
<td>5</td>
<td>6</td>
<td>6.25</td>
<td>7.5</td>
<td>10</td>
<td>12</td>
<td>12.5</td>
<td>15</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>60</td>
<td>5</td>
<td>6</td>
<td>6.25</td>
<td>7.5</td>
<td>10</td>
<td>12</td>
<td>12.5</td>
<td>15</td>
<td>20</td>
<td>24</td>
<td>25</td>
<td>30</td>
<td>40</td>
<td>48</td>
</tr>
<tr>
<td>Output fps</td>
<td>12</td>
<td>10</td>
<td>9.6</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>4.8</td>
<td>4</td>
<td>3</td>
<td>2.5</td>
<td>2.4</td>
<td>2</td>
<td>1.5</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Setpts_Value = Desired_Slow_Motion_Factor * Base_Framerate / 240

If the output framerate is too slow, you could use the "minterpolate" filter to calculate intermediate frames with motion interpolation.
Create a light curve of a star occultation

set "IN=P1000479.mov"  :: Input video
set "OUT=occultation.mp4" :: Output video

set "BP_R=0.015"  :: Black point red, positive value makes background darker
set "BP_G=0.005"  :: Black point green, positive value makes background darker
set "BP_B=0.015"  :: Black point blue, positive value makes background darker
set "WP=0.26"    :: White point

set "S=300"      :: Start time
set "T=40"       :: Duration

set "X=926"      :: X Position of star
set "Y=475"      :: Y Position of star
set "B=10"       :: Half of the box size for averaging the brightness, in this example the box is 20x20 pixels
set "MIN=60"     :: Minimum brightness for Y axis
set "MAX=90"     :: Maximum brightness for Y axis

set "FONT=arial.ttf" :: Font for the clock
set "COLOR=white"  :: Font color
set "BCOLOR=black" :: Background color
set "SIZE=30"      :: Font size
set "POS_X=0"      :: X position of clock
set "POS_Y=(h-th)" :: Y position of clock
set "OFFSET=2340" :: Offset time in seconds, added to the timestamp of the first frame

set CLOCK=drawtext='fontfile=%FONT%:text=%{pts\:hms:%OFFSET%}:fontcolor=%COLOR%:boxcolor=%BCOLOR%:box=1:fontsize=%SIZE\:x=%POS_X%:y=%POS_Y%'

rem  Create a video with contrast enhancement, with clock, but without light curve and markers:
rem ffmpeg -ss %S% -i %IN% -vf "colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%,%CLOCK%" -pix_fmt yuv420p -t %T% -y %OUT%
rem  Extract the first frame, for finding the x,y coordinates of the star:
ffmpeg -ss %S% -i %IN% -vf "colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%" -frames 1 -y frame1.png

rem Find the coordinates of the star (for example with IrfanView) and set the variables X,Y accordingly.
rem Then create a video with contrast enhancement, clock, light curve and markers at the star.
rem The light curve can be YAVG for average over the box or YMAX for the maximum in the box:

ffmpeg -ss %S% -i %IN% -lavfi "crop=2*%B%:2*%B%:%X%-%B%:%Y%-%B%,
signalstats,
drawgraph=m3=lavfi.signalstats.YAVG:mode=line:slide=scroll:min=%MIN%:max=%MAX%:size=1920x270:bg=0x000000@0.0[1];
[0]colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%,
drawbox=x=%X%+2*%B%:y=%Y%:c=yellow:t=1:w=2*%B%:h=1,
drawbox=x=%X%:y=%Y%+2*%B%:c=yellow:t=1:w=1:h=2*%B%[2];
[2][1]overlay=y=810,%CLOCK%" -pix_fmt yuv420p -t %T% -y %OUT%

Please note that for better readability the command line is shown here with line feeds. These must be removed in the real batch file.

Important note for the drawgraph filter:
The colors of the curves must be specified in the non-standard format 0xAABBGGR, however the background color must be specified in FFmpeg's normal color format 0xRRGGBBAA, for which many predefined colors are available.
This is the light curve of the star TYC1932-00469-1 which was occulted by the asteroid (87) Sylvia on October 30, 2019. The video was taken with a Canon EF 400mm f/2.8 lens, SpeedBooster 0.64x and Panasonic GH5S camera at 25600 ISO, FHD 25fps and [Ex. Tele Conv.] = 2.1x.
This is a batch file for drawing light curves of two stars simultaneously:

```batch
set "IN=P1000479.mov"      :: Input video
set "OUT=occultation.mp4"  :: Output video
::
set "BP_R=0.015"           :: Black point red, positive value makes background darker
set "BP_G=0.005"           :: Black point green, positive value makes background darker
set "BP_B=0.015"           :: Black point blue, positive value makes background darker
set "WP=0.26"              :: White point
::
set "S=300"                :: Start time
set "T=40"                 :: Duration
::
set "X1=926"               :: X Position of star
set "Y1=475"               :: Y Position of star
set "C1=0xffffffff"        :: Color for star curve, in format 0xAABBGGRR
set "X2=1054"              :: X Position of reference star
set "Y2=267"               :: Y Position of reference star
set "C2=0xffff00ff"        :: Color for reference star curve, in format 0xAABBGGRR
set "BG=0x00000000"        :: Background color for curves, in format 0xRRGGBBAA
set "B=10"                 :: Half of the box size for averaging the brightness
set "MIN=60"               :: Minimum brightness for Y axis
set "MAX=90"               :: Maximum brightness for Y axis
::
set "FONT=arial.ttf"       :: Font
set "COLOR=white"          :: Font color
set "BCOLOR=black"         :: Background color
set "SIZE=30"              :: Font size
set "POS_X=0"              :: X position of clock
set "POS_Y=(h-th)"         :: Y position of clock
set "OFFSET=2340"          :: Offset time in seconds, added to the timestamp of the first frame
::
set CLOCK=drawtext='fontfile=%FONT%:text=%%{pts\:hms\:%OFFSET%}:fontcolor=%COLOR%:boxcolor=%BCOLOR%:box=1:fontsize=%SIZE %:x=%POS_X%:y=%POS_Y%'

rem Extract the first frame:
rem ffmpeg -ss %S% -i %IN% -vf "colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%"
```

260
rem Find the coordinates of the star (with IrfanView) and set them to the variables X1,Y1 and X2,Y2.  
rem Then create a video with light curves and markers at the stars. The light curve can be YAVG for average over the box or YMAX for maximum in the box:

```bash
ffmpeg -ss %S% -i %IN% -lavfi "[0]crop=2*%B%:2*%B%:%X1%-%B%:%Y1%-%B%,signalstats,
drawgraph=m1=lavfi.signalstats.YAVG:mode=line:slide=scroll:min=%MIN%:max=%MAX%:size=1920x270:fg1=%C1%:bg=%BG%[1];
[0]crop=2*%B%:2*%B%:%X2%-%B%:%Y2%-%B%,signalstats,
drawgraph=m1=lavfi.signalstats.YAVG:mode=line:slide=scroll:min=%MIN%:max=%MAX%:size=1920x270:fg1=%C2%:bg=%BG%[2];
[2][1]overlay[3];
[0]colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%,
drawbox=x=%X1%+2*%B%:y=%Y1%:c=White:t=1:w=2*%B%:h=1,
drawbox=x=%X1%:y=%Y1%+2*%B%:c=White:t=1:w=1:h=2*%B%,
drawbox=x=%X2%+2*%B%:y=%Y2%:c=Violet:t=1:w=2*%B%:h=1,
drawbox=x=%X2%:y=%Y2%+2*%B%:c=Violet:t=1:w=1:h=2*%B%[4];
[4][3]overlay=y=810,%CLOCK%" -pix_fmt yuv420p -t %T% -y %OUT%
```  

This is a batch file for drawing the light curve and the audio level simultaneously. But it has the problem that the two graphs aren't running exactly synchronously. I don't know why.

```bash
set "IN=Pi00479.mov" :: Input video
set "OUT=occultation.mp4" :: Output video

set "BP_R=0.015" :: Black point red, positive value makes background darker
set "BP_G=0.005" :: Black point green, positive value makes background darker
set "BP_B=0.015" :: Black point blue, positive value makes background darker
set "WP=0.26" :: White point

set "S=300" :: Start time
set "T=40" :: Duration

set "X=926" :: X Position of star
set "Y=475" :: Y Position of star
set "C1=0xffffffff" :: Color for star curve, in format 0xAABBGGRR
```
set "C2=0xffff00ff" :: Color for audio curve, in format 0xAABBGGRR
set "BG=0x00000000" :: Background color for curves, in format 0xRRGGBBAA
set "B=10" :: Half of the box size for averaging the brightness
set "MAX=90" :: Maximum brightness for Y axis
set "MIN=70" :: Minimum brightness for Y axis
set "AMAX=0" :: Maximum audio level
set "AMIN=-50" :: Minimum audio level
set "FONT=arial.ttf" :: Font
set "COLOR=white" :: Font color
set "BCOLOR=black" :: Background color
set "SIZE=30" :: Font size
set "POS_X=0" :: X position of clock
set "POS_Y=(h-th)" :: Y position of clock
set "OFFSET=2340" :: Offset time in seconds, added to the timestamp of the first frame

set CLOCK=drawtext='fontfile=%FONT%:text=%%{pts:hms:%OFFSET%}:fontcolor=%COLOR%:boxcolor=%BCOLOR%:box=1:fontsize=%SIZE
:x=%POS_X%:y=%POS_Y%'

rem  Extract the first frame:
rem ffmpeg -ss %S% -i %IN% -vf "colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%"
                   -frames 1 -y frame1.png

rem  Find the coordinates of the star (with IrfanView) and set them to the variables X and Y.
rem  Then create a video with light curve and audio level curve. The light curve can be YAVG for average over the box or
YMAX for maximum in the box.

ffmpeg -ss %S% -i %IN% -lavfi [0:v]crop=2*B%:2*B%:X-%B%:Y-%B
       %,signalstats,drawgraph=m1=lavfi.signalstats.YAVG:mode=line:slide=scroll:min=%MIN%:max=%MAX%:size=1920x200:fg1=%C1%:bg=%BG%[1];
       [0:a]assetsamples=n=1920,astats=metadata=1:reset=1,drawgraph=m1=lavfi.astats.1.RMS_level:mode=bar:slide=scroll:min=%AMIN%:max=%AMAX%:size=1920x100:fg1=%C2%:bg=%BG%[2];[1][2]vstack[3];[0]colorlevels=rimin=%BP_R%:gimin=%BP_G%:bimin=%BP_B%:rimax=%WP%:gimax=%WP%:bimax=%WP%,
       drawbox=x=%X%+2*B%:y=%Y%:c=White:t=1:w=2*B%:h=1,drawbox=x=%X%:y=%Y%+2*B%
       %:c=White:t=1:w=2*B%[4];[4][3]overlay=y=750,%CLOCK% -pix_fmt yuv420p -t %T% -y %OUT%

pause
The value 1920 for `asetsamples` is the number of audio samples per video frame, in this case 48000 / 25 = 1920.
2.123 Oscilloscope

The oscilloscope filter can show the brightness of a video line:

<table>
<thead>
<tr>
<th>Set Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set &quot;IN=Pi1000479.mov&quot;</td>
<td>Input video</td>
</tr>
<tr>
<td>set &quot;OUT=out.mp4&quot;</td>
<td>Output video</td>
</tr>
<tr>
<td>set &quot;LINE=495&quot;</td>
<td>The shown line</td>
</tr>
<tr>
<td>set &quot;H=1080&quot;</td>
<td>Height of the video</td>
</tr>
</tbody>
</table>

ffmpeg -i %IN% -lavfi "oscilloscope=x=0.5:y=%LINE%/%H%:c=1" -t 10 -y %OUT%

pause

c=1 means show only the first component, in this case the luma component.
c=7 means show the first three components.

If you want to show the RGB components, add "format=rgb24," before the oscilloscope filter.

Note: It seems oscilloscope doesn't work with 10-bit videos.

2.124 Vectorscope

<table>
<thead>
<tr>
<th>Set Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set &quot;IN=test.mp4&quot;</td>
<td>Input video</td>
</tr>
</tbody>
</table>

ffplay %IN% -vf "split[a][b],[b]vectorscope=g=green[c],[a][c]overlay=x=W-w:y=H-h"
2.125 Capture a video from a webcam

List all supported, connected capture devices:

```bash
ffmpeg -list_devices 1 -f dshow -i dummy
```

List the possible video sizes, frame rates ans pixel formats for one capture device:

```bash
ffmpeg -list_options 1 -f dshow -i video="HD WebCam"
```

Capture video from the webcam:

```bash
ffmpeg -y -f dshow -video_size 1280x720 -framerate 10 -pixel_format yuyv422 -i video="HD WebCam" -t 5 out.mp4
```

See also: [https://trac.ffmpeg.org/wiki/DirectShow](https://trac.ffmpeg.org/wiki/DirectShow)

Built-in help for "dshow" input device:

```bash
ffmpeg -h demuxer=dshow
```

```bash
```
2.126 Capture video from the desktop or from a window

Capture the entire desktop:

```
set "FR=10" :: Framerate
ffmpeg -f gdigrab -framerate %FR% -i desktop -y out.mp4
pause
```

Capture a region of the desktop:

```
set "SHOW=1" :: 0 = do not show the border
               1 = show the border of the region on the desktop
set "FR=10" :: Framerate
set "SIZE=500x300" :: Size of the region
set "X=20" :: Left edge of region
set "Y=50" :: Top edge of region

ffmpeg -f gdigrab -show_region %SHOW% -framerate %FR% -video_size %SIZE% -offset_x %X% -offset_y %Y% -i desktop -y out.mp4
pause
```

Capture a window:

```
set "TITLE=*new 1 - Notepad++" :: Name of the window
set "FR=10" :: Framerate

ffmpeg -f gdigrab -framerate %FR% -i title="%TITLE%" -y out.mp4
pause
```

The title is the text in the title line of the window. It's not the name of the process in the task manager. A problem is that the title may dynamically change. For example, the title of an editor changes as soon as you begin to enter something (a * is inserted at the beginning).
Capture video and audio from a HDMI to USB converter

It's also possible to use a cheap HDMI to USB converter and capture the HDMI output from a camera, including audio (tested with GH5S).

The available options (pixel formats and video sizes) of the HDMI to USB converter can be shown with this command:

```bash
ffmpeg -list_options 1 -f dshow -i video="USB Video"
```

This is the output:

```
[dshow @ 0000000000184800] DirectShow video device options (from video devices)
[dshow @ 0000000000184800] Pin "Capture" (alternative pin name "0")
[dshow @ 0000000000184800] vcodec=mjpeg min s=1920x1080 fps=5 max s=1920x1080 fps=30
[dshow @ 0000000000184800] vcodec=mjpeg min s=1600x1200 fps=5 max s=1600x1200 fps=30
[dshow @ 0000000000184800] vcodec=mjpeg min s=1360x768 fps=5 max s=1360x768 fps=30
[dshow @ 0000000000184800] vcodec=mjpeg min s=1280x1024 fps=5 max s=1280x1024 fps=30
[dshow @ 0000000000184800] vcodec=mjpeg min s=1280x960 fps=5 max s=1280x960 fps=50
[dshow @ 0000000000184800] vcodec=mjpeg min s=1280x720 fps=10 max s=1280x720 fps=60.0002
[dshow @ 0000000000184800] vcodec=mjpeg min s=1024x768 fps=10 max s=1024x768 fps=60.0002
[dshow @ 0000000000184800] vcodec=mjpeg min s=800x600 fps=10 max s=800x600 fps=60.0002
[dshow @ 0000000000184800] vcodec=mjpeg min s=720x576 fps=10 max s=720x576 fps=60.0002
[dshow @ 0000000000184800] vcodec=mjpeg min s=720x480 fps=10 max s=720x480 fps=60.0002
[dshow @ 0000000000184800] vcodec=mjpeg min s=640x480 fps=10 max s=640x480 fps=60.0002
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=1280x720 fps=10 max s=1280x720 fps=10
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=1920x1080 fps=5 max s=1920x1080 fps=5
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=1600x1200 fps=5 max s=1600x1200 fps=5
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=1360x768 fps=8 max s=1360x768 fps=8
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=1280x1024 fps=8 max s=1280x1024 fps=8
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=1280x960 fps=8 max s=1280x960 fps=8
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=1024x768 fps=10 max s=1024x768 fps=10
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=800x600 fps=5 max s=800x600 fps=20
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=720x576 fps=5 max s=720x576 fps=25
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=720x480 fps=5 max s=720x480 fps=30
[dshow @ 0000000000184800] pixel_format=yuyv422 min s=640x480 fps=5 max s=640x480 fps=30
```

Note: Actually all these options are listed twice. I don't know why.

Note: Size 1600x1200 is a good choice for anamorphic 4:3 video from the GH5S.
The available audio options of the HDMI to USB converter can be shown with this command:

```bash
ffmpeg -list_options 1 -f dshow -i audio="Digitale Audioschnittstelle (US"
```

This is the output:

```
[dshow @ 000000000000003c48c0] DirectShow audio only device options (from audio devices)
[dshow @ 000000000000003c48c0] Pin "Capture" (alternative pin name "Capture")
[dshow @ 000000000000003c48c0] min ch=1 bits=8 rate= 11025 max ch=2 bits=16 rate=44100
```

This is the batch file for capturing video and audio from the HDMI to USB converter, using yuyv pixel format (no compression) but with low framerate (tested with GH5S):

```bash
ffmpeg -f dshow
```
```bash
-video_size 1920x1080
-framerate 5
-pixel_format yuyv422
-i video="USB Video":audio="Digitale Audioschnittstelle (US"
-q:v 2 -c:v mpeg4 -t 10 -y out.mp4
```

Note: On my (old) computer I use "-c:v mpeg4", because libx264 or libx265 are too slow for real time processing. In such cases "-rtbufsize" with a large buffer size can be used, for example "500M".

If the size is set to 1600x1200, the sample aspect ratio (that's the aspect ratio of the pixels) must be set to 4:3 with the setsar filter:

```bash
ffmpeg -f dshow
```
```bash
-video_size 1600x1200
-framerate 5
-pixel_format yuyv422
-i video="USB Video":audio="Digitale Audioschnittstelle (US"
-vf setsar=4/3 -q:v 2 -c:v mpeg4 -t 10 -y out.mp4
```

The same can also be done without filtering, by using the "-sar" option:

```bash
ffmpeg -f dshow
```
```bash
-video_size 1600x1200
-framerate 5
-pixel_format yuyv422
-i video="USB Video":audio="Digitale Audioschnittstelle (US"
-sar 4/3 -q:v 2 -c:v mpeg4 -t 5 -y out.mp4
```

This is the batch file for capturing video and audio from the HDMI to USB converter, using mjepg compression which allows higher framerates (tested with GH5S):
Note: In this example "-vcodec" selects a decoder, because it’s written before the input file. In most cases "-c" or "-codec" or "-vcodec" is an output option, but it can also be an input option either to tell FFmpeg which codec an input has, or (as in this case) to tell the input device which codec to emit.

This is the same for size 1600x1200 with anamorphic video from the GH5S:

```bash
ffmpeg -f dshow -rtbufsize 500M -video_size 1600x1200 -framerate 30 -vcodec mjpeg -i video="USB Video":audio="Digitale Audioschnittstelle (US)" -vf setsar=4/3 -q:v 2 -c:v mpeg4 -t 10 -y out.mp4
```

This is the same without filtering, with "-sar" option instead:

```bash
ffmpeg -f dshow -rtbufsize 500M -video_size 1600x1200 -framerate 30 -vcodec mjpeg -i video="USB Video":audio="Digitale Audioschnittstelle (US)" -sar 4/3 -q:v 2 -c:v mpeg4 -t 10 -y out.mp4
```

If mjpeg compression is used, it's possible to avoid re-encoding by using the "-c:v copy" option:

```bash
ffmpeg -f dshow -video_size 1920x1080 -framerate 30 -vcodec mjpeg -i video="USB Video":audio="Digitale Audioschnittstelle (US)" -c:v copy -t 10 -y out.mp4
```

Note: Unfortunately it's impossible to use "setsar" or "-sar" and "-c:v copy" at the same time.

The HDMI to USB converter has a built-in test image which shows 8 vertical color bars.

**Warning:** The test image is not generated if the converter is already plugged in when the computer is started. You must first start the computer and then plug in the converter.
Adding *.srt subtitles to a video

Create a *.srt subtitle file with the content in this format, the text must begin in the first line:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:00:00,000 --&gt; 00:00:03,000</td>
<td>This is the title text</td>
</tr>
<tr>
<td>2</td>
<td>00:00:04,000 --&gt; 00:00:05,000</td>
<td>Hello!</td>
</tr>
<tr>
<td>3</td>
<td>00:00:06,500 --&gt; 00:00:07,000</td>
<td>This video will end soon...</td>
</tr>
</tbody>
</table>

Method 1, burning the subtitles directly into the video frames:

```bash
ffmpeg -i subtitle.srt -y subtitle.ass
ffmpeg -i test.mp4 -vf ass=subtitle.ass -y out.mp4
```

Method 2, the output seems to be the same as with method 1:

```bash
ffmpeg -i test.mp4 -vf subtitles=subtitle.srt -y out.mp4
```

Same thing with specifying the fontsize and yellow color:

```bash
ffmpeg -i test.mp4 -vf subtitles=subtitle.srt:force_style='Fontsize=26,PrimaryColour=&H00ffff' -y out.mp4
```
Method 3, creating a subtitle stream that can be switched on/off in the player. Works fine with VLC player, but not with FFplay:

```
ffmpeg -i test.mp4 -i subtitle.srt -c:v copy -c:a copy -c:s mov_text -metadata:s:s:0 language=ger -y out.mp4
```

See also https://jacknorthrup.com/Multiple-Program-Languages-Documentation/FFMPEG.html

See also these Wiki pages:
https://trac.ffmpeg.org/wiki/HowToBurnSubtitlesIntoVideo
https://trac.ffmpeg.org/wiki/HowToConvertSubtitleToASS
https://trac.ffmpeg.org/wiki/HowToConvertYouTubeSBVtoSRT

If you want to create many subtitles, consider using a subtitler, for example "aegisub": http://www.aegisub.org/
2.129 Adding *.ass subtitles or credit files to a video

Create a *.ass subtitle file with this content:

```
[Script Info]
ScriptType: v4.00+
Collisions: Normal
PlayResX: 1920
PlayResY: 1080
Timer: 100.0000

[V4+ Styles]
Style: testStyle,Verdana,96,&H00A0A0A0,&H00000000,&H00000000,-1,0,0,100,100,0,0,0,1,1,0,2,0,0,0,0

[Events]
Format: Layer, Start, End, Style, Actor, MarginL, MarginR, MarginV, Effect, Text
Dialogue: 0,0:00:02.00,0:00:12.00,testStyle,,0000,0000,0000,,{"move(960,800,960,100)}{"fad(1000,1000)}This is the first line of this
Dialogue: 0,0:00:04.00,0:00:14.00,testStyle,,0000,0000,0000,,{"move(960,800,960,100)}{"fad(1000,1000)}test
Dialogue: 0,0:00:06.00,0:00:16.00,testStyle,,0000,0000,0000,,{"move(960,800,960,100)}{"fad(1000,1000)}This is a shorter line
Dialogue: 0,0:00:08.00,0:00:18.00,testStyle,,0000,0000,0000,,{"move(960,800,960,100)}{"fad(1000,1000)}Last line
```

Some hints about the *.ass file format:
- PlayResX and PlayResY are the dimensions of the screen where the texts are drawn on.
- "PrimaryColour" is the color of the text. The colors are in format &HAABBGGR, where AA = 00 means opaque and AA = FF means transparent
- Alignment: 1 = left, 2 = centered, 3 = right. Add 4 for a "Toptitle" and add 8 for a "Midtitle".
- The \move commands contain the x,y coordinates where the text appears and disappears. In this case the text is scrolling up.
- The \fad commands contain the fade-in and fade-out durations in milliseconds.

The example was found here: [http://forum.doom9.org/showthread.php?t=82025](http://forum.doom9.org/showthread.php?t=82025)

The *.ass file format is explained in detail here: [https://fileformats.fandom.com/wiki/SubStation_Alpha](https://fileformats.fandom.com/wiki/SubStation_Alpha)
Applying the texts to a video is very simple:

```
set "IN=blue.mov" :: Input video
set "T=20" :: Duration
set "OUT=out.mp4" :: Output video

ffmpeg -i %IN% -vf subtitles=credits.ass -t %T% -y %OUT%
```

2.130 frei0r

This seems to be an external library for video effects, but I was unable to find a list of possible effects. The documentation is horrible.

"Frei0r is mostly adopted on GNU/Linux and OSX platforms, counts more than 100 plugins and is used by several video software applications ..."

Where are these >100 plugins?

See also [https://frei0r.dyne.org/](https://frei0r.dyne.org/)

See also [https://en.wikipedia.org/wiki/Frei0r](https://en.wikipedia.org/wiki/Frei0r)
2.131   Attaching a preview image to a video

```
set "IN=test.mp4" :: Input video
set "T=3" :: Time of preview image
set "OUT=out.mp4" :: Output video

ffmpeg -ss %T% -i %IN% -vf scale=320:-1 -frames 1 -y image.png
ffmpeg -i %IN% -i image.png -map 0 -map 1 -c:0 copy -c:v:1 png -disposition:v:1 attached_pic -y %OUT%
```

Note: The codec for the preview image can be "png" or "mjpeg".

I did use a video from a Canon 5D-MK4 and got this error message:
"Could not find tag for codec none in stream #2, codec not currently supported in container. Could not write header for output file #0 (incorrect codec parameters?): Invalid argument"

This is probably because the video has an additional timecode stream, in this case replace "-map 0" by "-map 0:0 -map 0:1":

```
set "IN=test.mp4" :: Input video
set "T=3" :: Time of preview image
set "OUT=out.mp4" :: Output video

ffmpeg -ss %T% -i %IN% -vf scale=320:-1 -frames 1 -y image.png
ffmpeg -i %IN% -i image.png -map 0:0 -map 0:1 -map 1 -c:0 copy -c:v:1 png -disposition:v:1 attached_pic -y %OUT%
```

pause
It's also possible to write all in one command line:

```bash
set "IN=test.mp4" :: Input video
set "T=3" :: Time of preview image
set "OUT=out.mp4" :: Output video

ffmpeg -i %IN% -ss %T% -i %IN% -map 1:v:0 -map 0 -filter:v:0 scale=320:-1,trim=end_frame=1 -c copy -c:v:0 png -disposition:0 attached_pic -y %OUT%
pause
```

Or use this command line if your video has a timecode stream:

```bash
set "IN=test.mp4" :: Input video
set "T=3" :: Time of preview image
set "OUT=out.mp4" :: Output video

ffmpeg -i %IN% -ss %T% -i %IN% -map 1:v:0 -map 0:0 -map 0:1 -filter:v:0 scale=320:-1,trim=end_frame=1 -c copy -c:v:0 png -disposition:0 attached_pic -y %OUT%
pause
```
## 2.132 Expression evaluation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>abs(x)</code></td>
<td>Return absolute value of x.</td>
</tr>
<tr>
<td><code>acos(x)</code></td>
<td>Return arccosine of x.</td>
</tr>
<tr>
<td><code>asin(x)</code></td>
<td>Return arcsine of x.</td>
</tr>
<tr>
<td><code>atan(x)</code></td>
<td>Return arctangent of x.</td>
</tr>
<tr>
<td><code>atan2(x, y)</code></td>
<td>Return the arc tangent of <code>y/x</code> in the range -PI to PI.</td>
</tr>
<tr>
<td><code>between(x, min, max)</code></td>
<td>Return 1 if <code>x</code> is greater than or equal to <code>min</code> and lesser than or equal to <code>max</code>, 0 otherwise.</td>
</tr>
<tr>
<td><code>bitand(x, y)</code></td>
<td>Compute bitwise and operation on <code>x</code> and <code>y</code>.</td>
</tr>
<tr>
<td><code>bitor(x, y)</code></td>
<td>Compute bitwise or operation on <code>x</code> and <code>y</code>.</td>
</tr>
<tr>
<td><code>ceil(expr)</code></td>
<td>Round the value of expression <code>expr</code> upwards to the nearest integer. For example, &quot;ceil(1.5)&quot; is &quot;2.0&quot;.</td>
</tr>
<tr>
<td><code>clip(x, min, max)</code></td>
<td>Return the value of <code>x</code> clipped between <code>min</code> and <code>max</code>.</td>
</tr>
<tr>
<td><code>cos(x)</code></td>
<td>Compute cosine of <code>x</code>.</td>
</tr>
<tr>
<td><code>eq(x, y)</code></td>
<td>Return 1 if <code>x</code> and <code>y</code> are equivalent, 0 otherwise.</td>
</tr>
<tr>
<td><code>exp(x)</code></td>
<td>Compute exponential of <code>x</code>.</td>
</tr>
<tr>
<td><code>floor(expr)</code></td>
<td>Round the value of expression <code>expr</code> downwards to the nearest integer. For example, &quot;floor(-1.5)&quot; is &quot;-2.0&quot;.</td>
</tr>
<tr>
<td><code>gt(x, y)</code></td>
<td>Return 1 if <code>x</code> is greater than <code>y</code>, 0 otherwise.</td>
</tr>
<tr>
<td><code>gte(x, y)</code></td>
<td>Return 1 if <code>x</code> is greater than or equal to <code>y</code>, 0 otherwise.</td>
</tr>
<tr>
<td><code>hypot(x, y)</code></td>
<td>Return sqrt(x^2 + y^2)</td>
</tr>
<tr>
<td><code>if(x, y)</code></td>
<td>Evaluate <code>x</code>, and if the result is non-zero return the result of the evaluation of <code>y</code>, return 0 otherwise.</td>
</tr>
<tr>
<td><code>if(x, y, z)</code></td>
<td>Evaluate <code>x</code>, and if the result is non-zero return the evaluation result of <code>y</code>, otherwise the evaluation result of <code>z</code>.</td>
</tr>
<tr>
<td><code>ld(var)</code></td>
<td>Load the value of the internal variable with number <code>var</code>, which was previously stored with <code>st(var, expr)</code>. The function returns the loaded value. Please note that variables are currently not shared between expressions. But the variable keeps its value from one frame to the next. All variables are intialized with 0 at the beginning. Warning: The random(x) function uses the same variables!</td>
</tr>
</tbody>
</table>

---

276
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lerp(x, y, z)</td>
<td>Return x if z = 0, y if z = 1 and a linear interpolation for any value of z. There is no clipping for z &lt; 0 or z &gt; 1. The return value is: x + z * (y - x)</td>
</tr>
<tr>
<td>log(x)</td>
<td>Compute natural logarithm of x.</td>
</tr>
<tr>
<td>lt(x, y)</td>
<td>Return 1 if x is lesser than y, 0 otherwise.</td>
</tr>
<tr>
<td>lte(x, y)</td>
<td>Return 1 if x is lesser than or equal to y, 0 otherwise.</td>
</tr>
<tr>
<td>max(x, y)</td>
<td>Return the maximum between x and y.</td>
</tr>
<tr>
<td>min(x, y)</td>
<td>Return the minimum between x and y.</td>
</tr>
<tr>
<td>mod(x, y)</td>
<td>Return the remainder of division of x by y.</td>
</tr>
<tr>
<td>pow(x, y)</td>
<td>Return the power of x elevated y, it is equivalent to &quot;(x)^(y)&quot;.</td>
</tr>
<tr>
<td>print(t)</td>
<td>Print the value of expression t and returns the value of the expression printed.</td>
</tr>
<tr>
<td>random(x)</td>
<td>Return a pseudo random value between 0.0 and 1.0. x is the index of the internal variable which will be used to save the seed/state. Warning: The ld(var) and st(var, expr) functions use the same variables! Note: random(0) uses the variable 0 as a seed value. If you want to set the seed value, you must use the st(0, expr) function.</td>
</tr>
<tr>
<td>root(expr, max)</td>
<td>Find an input value for which the function represented by expr with argument ld(0) is 0. The input value must be in the interval [0..max]. The expression in expr must denote a continuous function or the result is undefined. ld(0) is used to represent the function input value, which means that the given expression will be evaluated multiple times with various input values that the expression can access through ld(0). When the expression evaluates to 0 then the corresponding input value will be returned. Warning: If there is no input value in the [0..max] interval for which the result of the expression becomes 0, then the root() function returns a wrong result!</td>
</tr>
<tr>
<td>round(expr)</td>
<td>Round the value of expression expr to the nearest integer. For example, &quot;round(1.5)&quot; is &quot;2.0&quot;.</td>
</tr>
<tr>
<td>sgn(x)</td>
<td>Return the sign of x (-1, 0 or +1)</td>
</tr>
<tr>
<td>sin(x)</td>
<td>Return sine of x.</td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>Return the square root of x.</td>
</tr>
<tr>
<td>st(var, expr)</td>
<td>Store the value of the expression expr in an internal variable. var specifies the number of the variable where to store the value, and it is a value ranging from 0 to 9. The function returns the value stored in the internal variable. Please note that variables are currently not shared between expressions. But the variable keeps its value from one frame to the next. All variables are initialized with 0 at the beginning. Warning: The random(x) function uses the same variables!</td>
</tr>
<tr>
<td>tan(x)</td>
<td>Compute tangent of x.</td>
</tr>
</tbody>
</table>
trunc(expr)  Round the value of expression expr towards zero to the nearest integer. For example, "trunc(-1.5)" is "-1.0".

while(cond, expr)  Evaluate expression expr while the expression cond is non-zero, and returns the value of the last expr evaluation, or NAN if cond was always false.

PI  approximately 3.1415

Two expressions expr1 and expr2 can be combined to form another expression "expr1;expr2". expr1 and expr2 are evaluated in turn, and the new expression evaluates to the value of expr2.

Workaround for segment-wise linear interpolation, one segment per second:
\[
\text{between}(t,0,1) \ast \text{lerp}(v0,v1,t) + \text{between}(t,1,2) \ast \text{lerp}(v1,v2,t-1) + \text{between}(t,2,3) \ast \text{lerp}(v2,v3,t-2) + \ldots
\]

Workaround for segment-wise linear interpolation, two segments per second:
\[
\text{between}(t,0,0.5) \ast \text{lerp}(v00,v05,2*t) + \text{between}(t,0.5,1.0) \ast \text{lerp}(v05,v10,2*(t-0.5)) + \text{between}(t,1.0,1.5) \ast \\
\text{lerp}(v10,v15,2*(t-1.0)) + \ldots
\]

The above two workarounds have a problem: If t is exactly at the border of two segments, then both "between" expressions are true. As a workaround, you can add 0.0001 to t.

If used inside the geq filter, the variable 't' must be written as a capital 'T'.

Please note that for most parameters expressions can't be used. They are only allowed for those parameters which are described as "expr" in the documentation. If in doubt, use ffmpeg -h filter=name_of_filter to get the types of the parameters.

2.133  Uploading videos to Facebook

It's possible to upload 10-bit videos to Facebook, but there may be a problem with the preview image, which is shown corrupted.

Use an 8-bit pixel format to avoid this problem, for example -pix_fmt yuv420p

If all videos don't play in Facebook, re-starting the browser may help.
2.134 Youtube recommended settings

```
```

Found on Rodrigo Polo’s github site: https://github.com/rodrigopolo/cheatsheets/blob/master/ffmpeg.md
2.135 Streaming from FFmpeg to YouTube Live

See also: How to encode Videos for YouTube, Facebook, Vimeo, twitch, and other Video Sharing Sites  [https://trac.ffmpeg.org/wiki/Encode/YouTube](https://trac.ffmpeg.org/wiki/Encode/YouTube)
See also: Encoding for streaming sites  [https://trac.ffmpeg.org/wiki/EncodingForStreamingSites](https://trac.ffmpeg.org/wiki/EncodingForStreamingSites)

The following hints were posted by Moritz Barsnick on the FFmpeg user list on April 18, 2020 (edited by me):

- In my experience, an audio stream is a "must" for YouTube live streaming. You can help yourself with silent audio with minimal bandwidth. I believe I have succeeded with 8 and 16 kb/s AAC. (I used to believe it needs to be two-channel audio, but my notes say I succeeded with mono.)
- You can add silence by adding a second input: `-f lavfi -i anullsrc` and encoding it with low bandwidth AAC: `-c:a aac -b:a 8k`
- The framerate must be 24, 25, 30 or 60 fps (or those divided by 1.001).
- Don't forget to check your YouTube live stream console. It gives some (hazy) indications about your stream's health.
- YouTube has some codec recommendations, but I believe you can deviate from them with little harm: [https://support.google.com/youtube/answer/2853702?hl=en](https://support.google.com/youtube/answer/2853702?hl=en)
Hardware acceleration

This command lists all hardware acceleration methods supported in this build of FFmpeg, regardless if the hardware is really available in this computer:

```
ffmpeg -hwaccels
pause
```

See also: [https://trac.ffmpeg.org/wiki/HWAccelIntro](https://trac.ffmpeg.org/wiki/HWAccelIntro)

Hardware accelerated h.264 encoding on machines with Nvidia GPUs:

```
ffmpeg -i in.mp4 -c:v h264_nvenc out.mp4
```

Hardware accelerated h.265/HEVC encoding on machines with Nvidia GPUs:

```
ffmpeg -i in.mp4 -c:v hevc_nvenc out.mp4
```

Note: Not all pixel formats are supported.

Note: It's possible that you get an error message "Driver does not support the required nvenc API version. Required: 10.0 Found: 9.1"

In this case go to the Windows Device Manager and update the Nvidia driver.
2.137 FFmpeg console output

<table>
<thead>
<tr>
<th>Explanation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAR</td>
</tr>
<tr>
<td>DAR</td>
</tr>
<tr>
<td>fps</td>
</tr>
<tr>
<td>tbr</td>
</tr>
<tr>
<td>tbn</td>
</tr>
<tr>
<td>tbc</td>
</tr>
</tbody>
</table>

2.138 FFmpeg source code

Many FFmpeg features have incomplete documentation. In these cases it may help to have a look at the source code as follows:

- Go to the main FFmpeg website [http://www.ffmpeg.org/](http://www.ffmpeg.org/)
- Click on "Source Code" and download the file "ffmpeg-snapshot.tar.gz"
- After unpackung, you see all the source code.
- The filters are in the folder "libavfilter"
- Most audio filters begin with "af_" and most video filters begin with "vf_"
FFmpeg: Suggestions for improvement

Listed in alphabetic order:

"acrossover" audio filter:
-- Documentation: Missing examples.

"afftfilt" filter:
-- Documentation: The possible options for the win_func parameter are missing.
-- Also, for the overlap parameter, what are the default values if set to 1?

"amix" filter:
-- Documentation: What are the default weights? Are they 1 or 1/n?
-- Possible bug, see ticket 8489.

"amplify" filter:
-- Many parameters are in the [0..65535] range. It's unclear how these parameters must be set for 8-bit videos. Is it [0..255] in this case, or is it always [0..65535]? Which range for a 10-bit video, [0..1023] or [0..65535]? Please add this info to the documentation.
-- Documentation for "threshold" parameter: "Any difference greater or equal to this value..." I'm not sure if this is correct, or if it should read "Any absolute difference greater or equal to this value...". Same question also for "tolerance" parameter.
-- Documentation: Does it have a built-in limiter?

"blend" filter:
-- Please add documentation what all the modes do. A few modes are self-explaining, but most are not.
"chromanr" filter:
-- Documentation of the threshold option is unclear. Possible typo, replace "or" by "of"? The algorithm also uses the absolute difference of Y components, this is missing in the official documentation.

This is how I think it works:
Foreach neighbour pixel
{
    A = absolute difference of Y components of current pixel and neighbour pixel
    B = absolute difference of U components of current pixel and neighbour pixel
    C = absolute difference of V components of current pixel and neighbour pixel
    if (A+B+C < threshold) then
        use this neighbour pixel for averaging
}

If the above is correct, I suggest to change the description as follows:
"The algorithm calculates the absolute difference of the Y components of the current pixel and a neighbour pixel. The same is also calculated for the U and V components. A neighbour pixel is used for averaging, if the sum of all three absolute differences is lower than the threshold. Only the U and V components are averaged. The Y component remains unchanged."

-- Also the meaning of "stepw" and "steph" options is unclear. Do these options refer to the pixels? If stepw=2 and steph=2 then only a quarter of the pixels is analyzed, and the same U and V components are used for 4 pixels? Or do they refer to the selection of the neighbour pixels? If stepw=2 and steph=2 then only a quarter of the neighbour pixels is analyzed? After looking in the source, it seems the latter is the case.

"colorbalance" filter:
Documentation: I don't understand it. In the first sentence it's said the filter modifies the intensity. What's meant here by "intensity"? Brightness or saturation? However in the third sentence it's said the colors are shifted. For me it's unclear what this filter does. More examples and/or a better description is required.

"cue" filter:
-- please add some examples to the documentation
"curves" filter:
-- In the documentation for "preset", please add the coordinates of the points for all presets. For example, what does "strong_contrast" mean? How strong is it? It would really help to know the coordinates of the points. This would also be usable as a starting point for defining your own curves.
-- Documentation of "master" option: I didn't understand it, please add an example and explain it.
-- Feature request: Allow import of curves files from GIMP.
-- Feature request: Make an option for using straight lines instead of smooth curves. This would be a nice workaround for strong linear contrast enhancement, by using only four points: 0/0 b/0 w/1 1/1 where b is the black point and w is the white point.

"deband" filter:
-- Documentation: Missing examples. I don't understand the description of this filter. The unit of the threshold values is unclear.

"deflicker" filter:
-- Documentation: It's unclear how this filter corrects the brightness of the images. Does it add a constant, or does it multiply by a constant? With other words: Does it change the brightness or the contrast?

"derain" filter:
-- Documentation: Please add examples.

"despill" filter:
-- Documentation is incomplete. Possible options are not explained. Missing examples. "Mix" and "Expand" parameters need more explanation how they work.

"DNG" Decoder:
-- Missing documentation. Searching for "DNG" in ffmpeg-all.html finds nothing.
-- Problem: FFmpeg's DNG decoder doesn't work with RAW images from Canon 6D or Canon 5D-MK4 which were converted to DNG with Adobe DNG converter 12.4, and it also doesn't work with DNG images from Pentax K5, and it also doesn't work correctly with the DNG images that were posted here: http://ffmpeg.org/pipermail/ffmpeg-user/2020-August/049681.html So far, I didn't find any DNG image that could be decoded correctly. The
"zscale" filter may be required, but I haven't figured out how the parameters must be set. If anybody has a working example for DNG to JPG or PNG conversion with FFmpeg, please let me know.

"drawgraph" filter:
-- Make clear in the documentation that the color format for the graphs is 0xAABBGGRR, however the color format for the background is 0xRRGGBBAA. Or even better: Correct this bug. But that would break existing command lines.

"eq" filter:
-- Documentation: Please add "Do note that what contrast does is scale the distance of a pixel's value from the median value i.e. 128 for a 8-bit input. So, if a pixel channel has a value of 100, then a contrast of 3 results in a value of 128 + 3*(100-128) = 44."

-- Documentation: Please also add that the order is always contrast -> brightness -> gamma, regardless of the order in the command line. Also fill in where saturation fits in the order (I don't know).

-- Documentation: Please add this example for brightness before contrast: eq=brightness=0.3,eq=contrast=5, and explain that eq=brightness=0.3;contrast=5 will be executed in the order contrast before brightness.

-- Documentation: It might be helpful to point out that this filter includes a limiter, if the result is out of range.

-- Feature request:
If we assume the video data is in the [0..1] range, the transfer function (before applying gamma) is: out = brightness + 0.5 + contrast * (in - 0.5)
This works fine for images where most pixels are in the center of the histogram. But it fails for images which consist mainly of a dark background, with only a few bright details. To amplify contrast in such images, both brightness and contrast must be adjusted, which is complicated.
I suggest to add a new parameter "pivot" in the range [0..1] which is 0.5 by default (so that it's compatible with old command lines).
out = brightness + pivot + contrast * (in - pivot)
With the p value properly set, the contrast can be adjusted without changing the brightness. This feature is already implemented in 3D_LUT_Creator.

-- Feature request: Correct handling of videos with alpha channel. The alpha channel should remain as it is. See my example in the "Blue screen" chapter.

Equirectangular 360° spherical videos:
-- Feature request: Inserting the required metadata so that VLC (and other players) recognize these videos as spherical. The same thing that the "Spatial Media Metadata Injector" does.
Expression evaluation:

-- `lerp(x, y, z)`  Documentation, better explanation: `lerp(x,y,z)` returns `x` if `z`=0, `y` if `z`=1 and interpolates if `0<z<1`. The return value is `x + z * (y - x)`. There is no clipping for `z < 0` or `z > 1`.

-- `print(expr)`  Suggestion for improvement: Optionally allow to add a string before the expression, so that during debugging it's easier to identify multiple print outputs in the log file. `print(string, expr)`

-- `root(expr, max)`  Problem: This function returns a wrong result if there is no root in the `[0...max]` interval. In such cases it should either throw an error message or return `NAN`.

-- `taylor(expr, x)`  `taylor(expr, x, id)`  Documentation: Better explanation and example required. I know what a taylor series is, but I don't understand how to use this FFmpeg function.

-- Feature request: A new function `getval(x, y, filename)` which reads a text file (or CSV file) and returns `x`-th value from the `y`-th line. Usable for many purposes where you have a parameter that can't be expressed by a simple formula, so that reading from an external file is easier. For example if you want to overlay a small video over the main video, at a variable position which is an arbitrary function of time.

-- Feature request: Share variables between expressions. This is especially required if inside the `geq` filter a variable is calculated by a long and complicated expression which depends only on `T` (with other words: a function of time). This expression will be evaluated for each pixel again and again, making the filter extremely slow. It would be much better to calculate the variable only one time per frame, and then make this variable readable inside the `geq` filter.

-- Feature request: Allow to set variables by reading from an input text or CSV file.

-- Feature request: A function for segment-wise linear or spline interpolation of a curve through a set of given points. Similar to this workaround for segment-wise linear interpolation: `between(T+0.001,0,1)*lerp(200,400,T)+between(T+0.001,1,2)*lerp(400,500,T-1)+between(T+0.001,2,3)*lerp(500,550,T-2)`

"fade" filter:

-- Feature request: Please allow a fade-out at the end of the video, where the start time is defined relative to the end of the video. With other words it should work without knowing the length of the video.

"filter_complex_script"

-- Documentation: Please add that line feeds and empty lines are allowed in the script file, which makes the content much better readable.

-- Feature request: Allow to set variables, like in a batch file. (This is the main reason why I don't use filter_complex_script)

-- Feature request: Allow comments in the script file.
"find_rect" video filter

-- The following things should be added to the documentation:

-- The meaning of the 'xmin, ymin, xmax, ymax' options is unclear. It should be mentioned in the documentation that these coordinates refer to
the top left corner of the object image. It's not the center of the object image! In the special case if the result is near the bottom right corner of the
specified range, the object image would lie almost completely outside of the specified range. To get the center coordinates, half the size of the object
image has to be added.

-- If the input video contains multiple instances of the object, find_rect will find only one of them.

-- threshold = 0.01 means only exact matches, threshold = 0.99 means almost everything matches (I'm not sure if this is correct)

-- The parameter "mipmaps" must be in the range [1...5]. The meaning of "mipmaps" is unclear.

-- If the threshold value is met, find_rect writes the result to these internal variables: lavfi.rect.w (width of object), lavfi.rect.h (height of object),
lavfi.rect.x (x position of object), lavfi.rect.y (y position of object). However it seems that cover_rect does always cover an area, totally independent of
the threshold. That makes no sense.

-- It is possible to write these variables to a file with ffprobe and -show_entries

-- Examples section: Find the position of an object in each frame and write it to a log file:
ffprobe -f lavfi movie=test.mp4,find_rect=object=object.pgm:threshold=0.3 -show_entries
frame=pkt_pts_time:frame_tags=lavfi.rect.x,lavfi.rect.y -of csv 1> log.csv

-- Feature request: Please make also the internal variable "best_score" available, so that it can be written to the log file as well.

-- Feature request: Add an option for using the result from the last frame as the starting point (plus or minus a specified range) for the current frame.
For example, if the last result was (x=700, y=500) let the search range for the current frame be from (700-R, 500-R) to (700+R, 500+R), where R is is used-
defined parameter. This could significantly speed up the algorithm, under the assumption that the object doesn't move far from one frame to the next.

"gdigrab"

-- Suggestion for improvement: When capturing a window with the "title" option, it's sometimes difficult to specify the title, because that may be a very
long string and copy-and-paste doesn't work. Another problem may be that the window title changes dynamically. I suggest that the title mustn't be an
exact match, however it should be sufficient if the window title contains the specified string.

"geq" filter:

-- Feature request: Allow commands, especially it should be possible to set a variable sd() inside the geq argument.
-- Feature request: Make it possible to access not only the current frame, but also the previous frames. For designing filters which require several frames.

-- Please add to the documentation that this filter has no built-in limiter for overflow handling if the result is out of range, and that the functions must be different for 8-bit or 10-bit videos. This filter doesn't interpret video data in the [0..1] range, but instead [0..255] for 8-bit video and [0..1023] for 10-bit video. This isn't clear in the documentation.

"gradients" video source:
-- Documentation: Typo in the line "x0, y0, y0, y1", correct is "x0, y0, x1, y1"

"lagfun" filter:
-- Please add to the documentation that it doesn't work with RGB24 pixel format, and throw an error message if the input is RGB24.

"maskfun" filter:
-- Documentation: I don't understand what this filter is doing.

"mpeg4" video encoder:
-- Missing documentation. There is some information in the wiki, but it's missing in the official documentation.
https://trac.ffmpeg.org/wiki/Encode/MPEG-4

"overlay" filter:
-- Feature request: "enable" option

"perspective" filter:
-- When the "sense=destination" option is used, the output size is smaller than the input size and the outer area is filled with the colors of the pixels at the edge of the video. It would be better to define a uniform color, with black as default. As a workaround I used the "pad" filter to create a black border around the video, before using the perspective filter. Please see my example in the "Video-in-Video" chapter.
"scale" filter:
-- Feature request: An option for reducing the size of an image, which calculates the brightness of the destination pixel not by averaging the source pixels, but instead by choosing the brightest (or darkest) of the source pixels. Usable for shrinking images of the starry night sky, without losing faint stars.

"selectivecolor" filter:
-- I don't understand the difference between "absolute" and "relative". Please add a better explanation and an example to the documentation.

"sendcmd" filter:
-- Documentation: Please add an example for sendcmd, if the target filter has more than one input. In this case the sendcmd filter can't be inserted directly before the target filter. When choosing a suitable position for sendcmd in the filter chain, make sure that at this position the duration is sufficient. If you have signal sources of different lengths, always choose the longest one for sendcmd.
-- Documentation: All arguments of the target filter must be initialized with valid values, even if these values are never used because sendcmd does always overwrite them.
-- Feature request: Allow that the sendcmd filter accepts any number of inputs, and just pass the inputs to the next filter. This would simplify things if you need a sendcmd before a target filter which has more than one input.

"speecnorm" filter
Documentation: I didn't understand how this filter works. "bellow" = "below"? What is a "local half cycle"? An example with a graphical waveform would be helpful.

"streamselect" filter:
-- Documentation: The examples are misleading. In the more general case it's impossible to write sendcmd directly before streamselect, because streamselect requires at least two inputs but sendcmd accepts only one input.
-- Feature request: Please allow expressions for "map". That's easier than sendcmd.

"superequalizer" audio filter:
-- Documentation: The unit of the parameters is unclear. Example missing.
"testsrc2" video source:
Documentation: "The testsrc2 source is similar to testsrc, but supports more pixel formats instead of just rgb24. This allows using it as an input for other tests without requiring a format conversion." Please explain how a different pixel format can be selected. The only way I know is to add ",format=yuv422p10le" for example. But isn't that a format conversion?

"tmix" filter:
-- Undocumented feature missing in documentation: If not specified, all weights are 1 by default.
-- Add the most simple example to the documentation: "tmix" works and is the same as "tmix=frames=3"
-- Undocumented feature missing in documentation: "tmix=frames=1" works and is a bypass mode.
-- The last sentence in the documentation is misleading and should be changed to "By default scale is set to 1 / (sum of weights)".

"trim" filter:
-- Feature request: Please allow expressions for the times.
-- Feature request: Please allow to specify the end time relative to the end of the video, so that it's possible to trim 5 seconds from the end of the video, without having to know the video length.

"v360" filter:
-- Error with "perspective" output: The output image is mirrored and the center of the input image is not in the center of the output image. The command v360=e:perspective should give the output that this workaround gives: v360=e:perspective:pitch=90:v_flip=1
-- Feature request: Support 4th order polynomial fisheye projection, as described on Paul Bourke's website: http://paulbourke.net/dome/fisheye/rectify/

"vibrato" audio filter
-- Documentation: It's unclear how a phase modulation can be expressed as a percentage. What does d=1.0 mean?

"xfade" filter:
It would be nice if it could be used similar as "acrossfade", where the crossfade is always at the end of the first video, so that you don't have to know the length of the first video. Could be used as a workaround for fade-out at the end of a video, by crossfading with a black video. I know that this requires a lot of memory, but this is acceptable as in most cases crossfadings aren't longer than 1-2 seconds.

"zmq" filter:
Please add this to the documentation:
It's possible to send zmq messages to all filters and options that support commands (these are the same same as for sendcmd). If you send a zmq message to an option that doesn't accept commands, then there is no error message, but the filter just ignores the message.

However, not everything that's possible is also useful. For example, a message can be sent to the "width" option of the "scale" filter and change the size of a video stream. But changing the size mid-stream isn't supported by many other filters (for example "eq", "colorkey" and "despill"). In some cases it works (for example "scale" immediately followed by "overlay"), but in most other cases it fails.

Suggestion for improvement: The zmq filter has the same problem as the sendcmd filter. Sometimes it's difficult to find a suitable location in the filter chain, where zmq can be inserted. It must be a location where exactly _one_ input exists. Not zero and not more than one. As the zmq filter doesn't really need any input, it should just pass all input streams unchanged to the output.

"zoompan" filter:
Suggestion for improvement: Please allow expressions (especially "iw" and "ih") for the "s" option.
Suggestion for improvement: Please allow zoom values smaller than 1, and use a color to fill the unused area.

"zscale" video filter:
Documentation: Much better description is required for almost all options. Examples are required. This filter seems to be required for decoding DNG images, but I wasn't able to find out how it works. This filter is unusable because of missing documentation.

Feature request for reading PGM files (required for example for "remap" filter): These files should contain only values in the [0..65535] range. In the case of ASCII PGM (P2) files, it's theoretically possible that the file contains negative values. FFmpeg should give a warning if a negative value is found.
Feature request: Cross correlation for audio. For comparing two audio channels from the same sound source, which were recorded with two microphones at different places, or with two different recorders that weren't started simultaneously. Automatically find the best-fit delay time between two audio channels. Can be used for synchronizing in-camera audio with sound from external recorder.

Feature request: Video stabilization for v360 videos (like deshake / vidstabdetect / vidstabtransform), see my C# workaround in this book.

Documentation: There are a few pairs of video filters which share a common description: lut2/tlut2, setdar/setsar and weave/doubleweave. The problem is that the second filter is difficult to find if you search by alphabet. I suggest to list these filters in correct alphabetic order, and the description of the second one contains only a link to the other filter. Same problem also for "afifo", "agraphmonitor" and "astreamselect" which are listed only in chapter "Video filters". They should be listed in "Audio filters". A link to the corresponding video filter would be sufficient.

All video sources:
-- Harmonize the default size of all video sources to the same value. At the moment some sources are 320x240 and some are 640x480. For example, if you need a "color" and a "testssrc2" source in a command line, you have to specify the size for at least one of them, because the default sizes are different.

"ffplay-all.htm"

Why are the values of the options "-left" and "-top" called "title"? That should be integers.
3 Audio processing with FFmpeg

3.1 Combine multiple audio files with crossfadings

```plaintext
rem  Combine multiple audio files with crossfadings

set "FILE1=Sound_Day.wav"    :: First audio filename
set "V1=1.0"                 :: Volume
set "S1=0"                   :: Start time
set "L1=14"                  :: Length
set "FILE2=Sound_Night.wav"  :: Second audio filename
set "V2=0.2"                 :: Volume
set "S2=20"                  :: Start time
set "L2=55"                  :: Length
set "FILE3=Sound_Day.wav"    :: Third audio filename
set "V3=1.0"                 :: Volume
set "S3=20"                  :: Start time
set "L3=30"                  :: Length
set "DUR=5"                  :: Crossfade duration
set "OUT=sound.mp3"          :: Output audio filename

ffmpeg -ss %S1% -i %FILE1% -t %L1% -af volume=%V1% -y s1.wav
ffmpeg -ss %S2% -i %FILE2% -t %L2% -af volume=%V2% -y s2.wav
ffmpeg -ss %S3% -i %FILE3% -t %L3% -af volume=%V3% -y s3.wav
ffmpeg -i s1.wav -i s2.wav -filter_complex acrossfade=d=%DUR% -y s12.wav
ffmpeg -i s12.wav -i s3.wav -filter_complex acrossfade=d=%DUR% -y %OUT%
```

In this example three audio files are concatenated with crossfadings. For each file the volume, start time and length can be specified. At first three temporary files are created, then the first two are combined, and in the last step the third file is added. There is no quality loss because "*.wav is an uncompressed audio format.
3.2 Change audio sample rate and number of audio channels

```plaintext
rem Change the audio sample rate and the number of audio channels

set "IN=PanoView.mp4" :: Input video
set "START=5" :: Start time
set "LEN=5" :: Length
set "OUT=out.mp4" :: Output video

ffmpeg -ss %START% -t %LEN% -i %IN% -ac 2 -af aresample=44100 -y %OUT%

pause
```

- `-ac 2` sets the number of audio channels to 2. If you want to copy a mono channel to both stereo channels, use `aeval=val(0)|val(0)`
- `-af aresample=44100` changes the audio sample rate to 44100 Hz
### 3.3 Change audio length, sample rate and/or pitch

<table>
<thead>
<tr>
<th>Command line</th>
<th>Length</th>
<th>SR</th>
<th>Pitch</th>
<th>What has changed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ffmpeg -f lavfi -i sine=f=1000:r=48000:d=5 s.wav</td>
<td>5 s</td>
<td>48000</td>
<td>1000 Hz</td>
<td>(This is the input file)</td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af asetrate=24000,atempo=2.0,aresample=48000 t1.wav</td>
<td>5 s</td>
<td>48000</td>
<td>500 Hz</td>
<td>Pitch</td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af aresample=24000 t2.wav</td>
<td>5 s</td>
<td>24000</td>
<td>1000 Hz</td>
<td>SR</td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af asetrate=24000,atempo=2.0 t3.wav</td>
<td>5 s</td>
<td>24000</td>
<td>500 Hz</td>
<td>SR and Pitch</td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af atempo=2.0 t4.wav</td>
<td>2.5 s</td>
<td>48000</td>
<td>1000 Hz</td>
<td>Length</td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af asetrate=24000,aresample=48000 t5.wav</td>
<td>10 s</td>
<td>48000</td>
<td>500 Hz</td>
<td>Length and Pitch</td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af atempo=2.0,aresample=24000 t6.wav</td>
<td>2.5 s</td>
<td>24000</td>
<td>1000 Hz</td>
<td>Length and SR</td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af asetrate=24000 t7.wav</td>
<td>10 s</td>
<td>24000</td>
<td>500 Hz</td>
<td>Length, SR and Pitch</td>
</tr>
<tr>
<td>Change the length by factor L and pitch by factor P:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set &quot;L=1.2&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set &quot;P=1.5&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ffmpeg -i s.wav -af atempo=1/%L%/%P%,asetrate=48000*%P%,aresample=48000 t8.wav</td>
<td>6 s</td>
<td>48000</td>
<td>1500 Hz</td>
<td>Length and Pitch</td>
</tr>
</tbody>
</table>

Please note that the argument of the atempo filter must be in the [0.5 ... 2.0] range.

See also the rubberband filter.
3.4 Replace a segment of the audio stream by silence

```
set "B=10"  :: Time where silence begins
set "E=10"  :: Time where silence ends
ffmpeg -i in.mp4 -c:v copy -af "volume=enable='between(t,%B%,%E%)':volume=0" out.mp4
```

3.5 Add an audio stream to a video, if no audio stream exists

In this example a silent stereo audio stream is added to a video, if (and only if) the video has no audio stream. Otherwise the audio stream remains unchanged:

```
rem Make a test video without audio stream:
ffmpeg -f lavfi -i testsrc2=size=vga -t 6 -y none.mp4
rem Make a test video with mono audio stream:
ffmpeg -f lavfi -i testsrc2=size=vga -f lavfi -i sine=1000 -t 6 -y mono.mp4
rem Make a test video with stereo audio stream:
ffmpeg -f lavfi -i testsrc2=size=vga -f lavfi -i sine=1000 -t 6 -ac 2 -y stereo.mp4
ffmpeg -i none.mp4 -flavfi -i anullsrc=cl=stereo -shortest -y test1.mp4
ffmpeg -i mono.mp4 -flavfi -i anullsrc=cl=stereo -shortest -y test2.mp4
ffmpeg -i stereo.mp4 -flavfi -i anullsrc=cl=stereo -shortest -y test3.mp4
```

In this example test1.mp4 will have a silent stereo audio stream, test2.mp4 will have the original mono audio stream and test3.mp4 will have the original stereo audio stream.

This example is similar, but the output audio stream is forced to be mono in all three cases:
This example is similar, but the output audio stream is forced to be stereo in all three cases:

```bash
ffmpeg -i none.mp4 -f lavfi -i anullsrc -shortest -ac 2 -y test1.mp4
ffmpeg -i mono.mp4 -f lavfi -i anullsrc -shortest -ac 2 -y test2.mp4
ffmpeg -i stereo.mp4 -f lavfi -i anullsrc -shortest -ac 2 -y test3.mp4
```

How does it work?

If "map" is not specified, FFmpeg selects a single audio stream from among the inputs with the highest channel count. If there are two or more streams with same number of channels, it selects the stream with the lowest index. `anullsrc` here has one channel, so it will be passed over except when the source video has an audio stream.

See also: [https://stackoverflow.com/questions/37862432/ffmpeg-output-silent-audio-track-if-source-has-no-audio-or-audio-is-shorter-th](https://stackoverflow.com/questions/37862432/ffmpeg-output-silent-audio-track-if-source-has-no-audio-or-audio-is-shorter-th)

### 3.6 Stereo --> mix into one mono channel

Both channels of the stereo stream will be downmixed into the stream:

```bash
ffmpeg -i stereo.wav -ac 1 mono.wav
```
3.7  Check if both stereo channels are equal

In this example the difference between the two audio channels is calculated and written to a mono file. If the result is silence, then both input channels were equal. The input can be a video or an audio file.

```bash
ffmpeg -i input.mp4 -af "aeval=val(0)-val(1)" mono.wav
```

3.8  Extract one mono channel from stereo

```bash
ffmpeg -i stereo.wav -filter_complex \\
"[0:a]channelsplit=channel_layout=stereo:channels=FR[right]\" -map \"[right]\" \\
front_right.wav
```

3.9  Stereo --> two mono channels

```bash
ffmpeg -i stereo.wav -filter_complex \\
"[0:a]channelsplit=channel_layout=stereo[left][right]\" -map \"[left]\" left.wav -map \"[right]\" right.wav
```

This command line does the same thing:

```bash
ffmpeg -i stereo.wav -map_channel 0.0.0 left.wav -map_channel 0.0.1 right.wav
```

3.10  Mono --> stereo

Of course both stereo channels will be identical.

```bash
ffmpeg -i input.wav -ac 2 output.wav
```
3.11 Two mono channels --> stereo

```bash
ffmpeg -i left.mp3 -i right.mp3 -filter_complex "[0:a][1:a]join=inputs=2:channel_layout=stereo[a]" -map "[a]" output.mp3
```

3.12 Mix two stereo channels to one stereo channel

```bash
ffmpeg -i input1.wav -i input2.wav -filter_complex "[0:a][1:a]amerge=inputs=2,pan=stereo|c0<c0+c2|c1<c1+c3[a]" -map "[a]" output.mp3
```

Or use this command line, the output may be different:

```bash
ffmpeg -i input1.wav -i input2.wav -filter_complex "[0:a][1:a]amerge=inputs=2[a]" -map "[a]" -ac 2 output.mp3
```

3.13 How to choose the correct audio volume level

Normally music is normalized to the maximum value (+-32676 for 16-bit). That means the loudest part uses the maximum possible values, just without clipping. You can use the music for your video as-is, or you can make it quieter. If you make it louder, then it may be clipped.

Things are totally different when you make your own sound records, for example nature sounds.

As the first step, I recommend to calibrate the volume knob of your amplifier. To do this, show several videos from different sources (not your own selfmade videos), and adjust the volume knob so that all videos sound just right, with other words: Adjust the volume knob so, as you would like to hear these videos in the planetarium. To make sure that the frequency response is acceptable, use good 3-way boxes. Leave the volume knob in this position and don't change it.

Now you can adjust the volume of your own video, so that it also sounds great in the planetarium. This ensures that you can play all videos (your own and other videos) one after the other. You don't want to touch the volume knob during a presentation!
3.14 Remove low frequencies (wind noise) from an audio track

rem Audio high pass filtering and volume adjustment

set "IN=sound.wav" :: Input soundtrack
set "AS=20" :: Start time
set "LEN=60" :: Length
set "HP=500" :: Cut-off frequency of the high pass filter
set "VOL=10" :: Volume factor
set "OUT=out.mp3" :: Output soundtrack

ffmpeg -ss %AS% -i %IN% -af highpass=f=%HP%,highpass=f=%HP%,highpass=f=%HP%,volume=%VOL% -t %LEN% -y %OUT%
pause

The high pass filter attenuates low frequencies by 12 dB per octave. At the specified cut-off frequency, the filter has 3dB attenuation. In this example, the same filter is used three times in a row, resulting in 36dB per octave.
3.15 Make a video from an audio file

Suppose an audio file is to be shown on Facebook. However, this is not possible because only pictures or videos can be shown there. Solution: The audio file is extended with a monochrome picture to a video.

```
ffmpeg -f lavfi -i color=c=black -i audio.mp3 -shortest out.mp4
```

Do the same thing with a picture:

```
ffmpeg -loop 1 -i picture.jpg -i audio.mp3 -shortest out.mp4
```

Do the same thing and use only a time segment from the audio file:

```
set "IN=IMG_1313.jpg"          :: Input image
set "SOUND=190923_0019_12.wav" :: Sound file
set "SS=110"                   :: Start time in the sound file
set "SIZE=1600x1200"           :: Size of output video
set "T=10"                     :: Duration
set "OUT=out.mp4"              :: Output video

ffmpeg -loop 1 -i %IN% -ss %SS% -i %SOUND% -s %SIZE% -t %T% -y %OUT%
```

pause
3.16  Convert ultrasound to the audible range, e.g. to hear bats

There are two fundamentally different methods of converting a high frequency to a lower frequency. The first method is to divide the frequency by a constant (for example 2), in this case the frequency range from 0 to 25kHz is converted to the range from 0 to 12.5kHz.

```plaintext
rem Make 2 seconds 1kHz sinewave, followed by 2 seconds silence:
ffmpeg -f lavfi -i "sine=frequency=1000:sample_rate=48000:duration=2" -af apad -t 4 sine.wav

rem Halving the sampling rate doubles the duration and halves the pitch
ffmpeg -i sine.wav -af asetrate=24000 out1.mp3

rem The atempo=2 filter causes the duration to be halved and the pitch to remain unchanged
rem (The factor must be in the range [0.5 .. 2.0], if necessary you can use the filter several times in a row)
ffmpeg -i sine.wav -af atempo=2.0 out2.mp3

rem A combination of these two effects causes the duration to remain unchanged and the pitch to be halved:
ffmpeg -i sine.wav -af asetrate=24000,atempo=2.0 out3.mp3
```

```
rem 2 seconds 1kHz sinewave, followed by 2 seconds silence:
```

The second method is to subtract a constant frequency. In this case, for example, the frequency range [15kHz ... 25kHz] is converted to the range [0kHz ... 10kHz]. The advantage is that the frequency range from 0 to 15kHz is completely suppressed.

```plaintext
rem Ultrasound converter by mixing (subtraction of the mixing frequency)

set "IN=Fledermaus_44100.wav"     :: Input soundtrack (containing ultrasound)
set "SR=44100"                    :: Sample rate of the input soundtrack
set "MF=15000"                    :: Mixing frequency (this is the frequency to be subtracted)
set "BB=10000"                    :: Bandwidth
                              :: The frequency range [MF ... MF+BB] is converted to the range [0Hz ... BB]
set "VOL=3"                       :: Volume factor
set "OUT=out.wav"                 :: Output soundtrack

ffmpeg -ss 100 -i %IN% -f lavfi -i aevalsrc="sin(%MF%*2*PI*t):c=stereo:s=%SR%" ^
-filter_complex "[0]volume=%VOL%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%[sound];
[sound][1]amultiply,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%" -y %OUT%
```

303
The amultiply filter in the above example does multiply the input signal by a sine wave. Both inputs and the output are in the [-1...+1] range.

In this example, the ultrasound from a video is mixed to the audible range and the video is copied as-is:

```plaintext
rem  Ultrasound converter by mixing (subtraction of the mixing frequency)
set "IN=7Z7A1699.MOV"             :: Input video
set "SR=48000"                    :: Sample rate of the input soundtrack
set "MF=12000"                    :: Mixing frequency (this is the frequency to be subtracted)
set "BB=10000"                    :: Bandwidth
:: The frequency range [MF ... MF+BB] is converted to the range [0Hz ... BB]
set "VOL=40"                      :: Volume factor
set "OUT=699.mp4"                 :: Output video
ffmpeg -i %IN% -f lavfi -i aevalsrc="sin(%MF%*2*PI*t):c=stereo:s=%SR%" ^
-filter_complex ":[0]volume=%VOL%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%[sound];
[sound][1]amultiply,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%" -y %OUT%
```

If the output audio sample rate is specified with -ar, it must be a sane sample rate such as 44.1k or 48k.

The same thing can now (October 2020) be made much easier with the new "afreqshift" filter:

```plaintext
set "IN=s.wav"                   :: Input file
set "FS=-13000"                  :: Frequency shift
rem  Make a ultrasonic 15kHz test tone
ffmpeg -f lavfi -i sine=f=15000:r=48000:d=5 -y %IN%
rem  Shift the frequency into the audible range
ffmpeg -i %IN% -af afreqshift=%FS% -y test.wav
```
In this example I did add the "showspectrum" filter for visualizing the spectrum:

```bash
set "FS=14000"              :: Frequency shift
rem Make a video with a 15kHz test tone:
ffmpeg -f lavfi -i sine=f=15000:r=48000:d=5 -f lavfi -i color=black -lavfi showspectrum=legend=1 -y test.mp4
rem Shift the frequency down:
ffmpeg -i test.mp4 -lavfi highpass=FS%,highpass=FS%,highpass=FS%,afreqshift=-FS%,asplit[a][b];
[b]showspectrum=legend=1 -map [a] -y out.mp4
pause
```

Note: "showspectrum" is a audio to video filter. That's why the audio signal must be duplicated with the "asplit" filter.

Note: The multiple "highpass" filters are used to strongly attenuate the frequencies below the shift frequency, so that these frequencies can't be heard when they are mirrored into the negative frequency range.

Note: "afreqshift" does not operate in the frequency domain. It uses two groups of allpass sections of biquad IIR filters to do most of the work.
Record sound with the computer's built-in microphone

With this batch file you can see which microphones are available:

```bash
ffmpeg -list_devices 1 -f dshow -i dummy
pause
```

With this batch file you can display the properties of a specific microphone:

```bash
ffmpeg -list_options 1 -f dshow -i "audio=Mikrofon (Realtek High Definiti"
pause
```

With this batch file you can record sound with the internal microphone:

```bash
ffmpeg -f dshow -channels 2 -i audio="Mikrofon (Realtek High Definiti" -t 5 -f mp3 -y out.mp3
pause
```

With this batch file you can record sound with a cheap chinese USB soundcard (Model "3D SOUND"):

```bash
rem ffmpeg -list_devices 1 -f dshow -i dummy
rem ffmpeg -list_options 1 -f dshow -i "audio=Mikrofon (USB Audio Device)"
ffmpeg -f dshow -sample_rate 44100 -sample_size 16 -channels 2 -i audio="Mikrofon (USB Audio Device)" -t 5 -f mp3 -y out.mp3
pause
```
3.18 Record a "Voice-Over" audio track

A "Voice-Over" track is an audio track that's recorded while simultaneously a video is played. Useful also for making sound effects that fit to the video.

Note: Later I found out that it's much easier to record a Voice-Over track with Davinci Resolve. But for completeness I'm also showing here how it can be done with FFmpeg.

```
set "IN=test.mp4"           :: Input video
set "AUDIO=audio.wav"       :: Output audio

ffmpeg -re -i %IN% -f dshow -audio_buffer_size 100 -channels 2 -i audio="Mikrofon (Realtek High Definiti" -y -map 1:a %AUDIO% -map 0:v -f sdl2 -
```

Please note that this command line has several problems:

1. It doesn't stop when the end of the video has been reached. The audio file gets longer than the video. But you can manually close the console window.
2. Video and audio are not perfectly synchronised. This depends also on the "audio_buffer_size" value.
3. If audio_buffer_size is large (or if the large default value is used), FFmpeg doesn't play the video continuously. It's stop-and-go.
4. I found no documentation for -f sdl2

Alternatively it's also possible to pipe the video output to FFplay. In this case the scale filter is required to make the FFplay window smaller, so that you can still see the console window and close it manually when the video has ended.

```
set "IN=test.mp4"           :: Input video
set "AUDIO=audio.wav"       :: Output audio

ffmpeg -an -i %IN% -f dshow -audio_buffer_size 100 -channels 2 -i audio="Mikrofon (Realtek High Definiti" -y -map 1:a %AUDIO% -map 0:v -vf scale=iw/2:-1 -f nut - | ffplay -
```

This example that was proposed by Gyan Doshi in the FFmpeg user list on January 17th, 2020. It uses the mpegts format instead. It works, but unfortunately it takes about 5 seconds until the FFplay window appears. The -autoexit option should close FFplay when EOF is detected, but in this example it doesn't work.
The original video (with original audio) and the new recorded audio can now be mixed with this command line. The best value for "offset" has to be found by try and error.

```bash
set "IN=test.mp4" :: Input video (with audio)
set "AUDIO=audio.wav" :: Input audio
set "OFFSET=0.35" :: Offset time in seconds, a positive value means audio is shifted towards the beginning
set "W1=0.2" :: Weight of original sound from the video
set "W2=2.5" :: Weight of sound from audio file

ffmpeg -i %IN% -i %AUDIO% -filter_complex ",atrim=%OFFSET%,amix=weights='1:%W1% %W2%'" -y -shortest -c:v mpeg4 -c:a aac out.mp4
```

Note: In the above example I did use the mpeg4 encoder, because with the default lib264 encoder there is a problem. Unexpectedly the audio in the output file is shorter than the video. The problem can be reproduced as follows:

```bash
ffmpeg -f lavfi -i testsrc=size=vga -f lavfi -i sine=1000 -t 6 -y video.mp4
ffmpeg -i video.mp4 -i video.mp4 -lavfi "[0:a][1:a]amix=weights='1.0 0.1'" -y out.mp4

rem 3 known workarounds:
rem ffmpeg -i video.mp4 -i video.mp4 -lavfi "[0:a][1:a]amix=weights='1.0 0.1'" -c:v mpeg4 -y out.mp4
rem ffmpeg -i video.mp4 -i video.mp4 -lavfi "[0:a]apad[p];[p][1:a]amix=weights='1.0 0.1'" -shortest -y out.mp4
rem ffmpeg -i video.mp4 -i video.mp4 -lavfi "[0:a]apad=pad_len=1[p];[p][1:a]amix=weights='1.0 0.1'" -y out.mp4
```

pause
3.19 Passing the FFmpeg output to FFplay

This batch file passes the video and audio output of FFmpeg to FFplay

```
ffmpeg -i in.mp4 (insert some filters here) -f nut - | ffplay -
```

```
pause
```

3.20 Record sound and pass the output to FFplay

This batch file records sound from the computer's microphone (or audio input) and passes the output to FFplay

```
ffmpeg -f dshow -sample_rate 44100 -sample_size 16 -channels 2 -i "audio=Microphone (SoundMAX Integrated)" (insert some filters here) -f wav - | ffplay -
```

```
pause
```
3.21 Live ultrasound conversion

It's possible to make ultrasound conversion in almost real time. You can input the ultrasound via the computer's microphone (or 3.5mm input jack), and play the converted sound via the computer's speakers (or 3.5mm headphone output to an audio amplifier). The conversion has about 1-2 seconds delay. Make sure that in the Windows control panel, in the microphone properties under "Microphone Extensions", no filter should be used.

```
rem ffmpeg -list_devices 1 -f dshow -i dummy
rem ffmpeg -list_options 1 -f dshow -i "audio=Mikrofon (Realtek High Definiti"

rem Live ultrasound converter by mixing (subtraction of the mixing frequency)

set "SR=44100"        :: Sample rate of the input soundtrack
set "MF=10000"        :: Mixing frequency (this is the frequency to be subtracted)
set "BB=10000"        :: Bandwidth
                      :: The frequency range [MF ... MF+BB] is converted to the range [0Hz ... BB]
set "VOL=30"          :: Volume factor

ffmpeg -f dshow -channels 2 -i audio="Mikrofon (Realtek High Definiti" -f lavfi -i aevalsrc="sin (%MF%*2*PI*t):c=stereo:s=%SR%" -filter_complex ":[0]volume=%VOL%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%,highpass=f=%MF%[sound];[sound][1]amultiply,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%" -f nut - | ffmpeg -
```

Pinout of 3.5mm stereo connectors: Tip contact is left channel, middle contact is right channel, outer contact is ground.

This is another method for live ultrasound conversion. It's faster and uses the FFT filter:

```
rem ffmpeg -list_devices 1 -f dshow -i dummy
rem ffmpeg -list_options 1 -f dshow -i "audio=Mikrofon (Realtek High Definiti"

rem Live ultrasound converter by mixing (subtraction of the mixing frequency)

set "SR=44100"        :: Input sample rate
set "F=4000"          :: Subtracted frequency in Hz
set "VOL=30"          :: Volume factor
```
In this example the delay time between input and output can be minimized by setting the -audio_buffer_size to a small value, for example with 10ms buffer size the delay is about 0.5 seconds.

See also: https://trac.ffmpeg.org/wiki/DirectShow#BufferingLatency
3.22 Extract the audio from a video

```
ffmpeg -i video.mp4 -vn audio.mp3
```

3.23 Split a video into audio-only and video-only

Audio and video are saved if individual files.

```
ffmpeg -i input.mp4 -vcodec mpeg2video output_video.m2v -acodec copy output_audio.mp3
```

3.24 Synchronize audio with video

If you have a video with out-of-sync audio, you can synchronize it as follows. In this example a 0.5 seconds delay is added to the audio stream:

```
ffmpeg -i input.mp4 -itsoffset 0.5 -i input.mp4 -map 0:0 -map 1:1 -acodec copy -cvodec copy output.mp4
```

See also the "compensationdelay" filter for delaying audio.

3.25 Sources for royalty-free music

Royalty-free music can be found here for example: [http://opsound.org](http://opsound.org)

But royalty-free doesn't mean you can do what you want with the music. You should read the licence carefully. For example it may be required to give the artist a proper credit, and show a link to the licence in the video.
### Sound effects, from frequency domain to time domain

<table>
<thead>
<tr>
<th>Frequency domain</th>
<th>Time domain</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>freq(t)</td>
<td>[y(t) = \sin(2\pi \cdot \text{freq}(t) \cdot t)]</td>
<td>This formula is wrong. It doesn't work this way.</td>
</tr>
<tr>
<td>freq(t)</td>
<td>[y(t) = \sin(2\pi \cdot \int_0^t \text{freq}(t) , dt)]</td>
<td>This is the correct way to calculate the signal in the time domain.</td>
</tr>
<tr>
<td>freq(t) = f0</td>
<td>[y(t) = \sin(2\pi \cdot t \cdot f_0)]</td>
<td>Sine tone with constant frequency</td>
</tr>
<tr>
<td>freq(t) = f0 + (f1 - f0) \cdot t / p</td>
<td>[y(t) = \sin(2\pi \cdot t \cdot (f_0 + \frac{f_1 - f_0}{p} \cdot t))]</td>
<td>Linear chirp from f0 to f1 in p seconds</td>
</tr>
<tr>
<td>freq(t) = f0 + f1 \cdot \sin(2\pi \cdot t)</td>
<td>[y(t) = \sin(2\pi \cdot t \cdot f_0 - f_1 \cdot \cos(2\pi \cdot t))]</td>
<td>Sinusoidally rising and falling tone from f0-f1 to f0+f1 with a period of one second</td>
</tr>
<tr>
<td>freq(t) = f0 + f1 \cdot \sin(2\pi \cdot t / p)</td>
<td>[y(t) = \sin(2\pi \cdot t \cdot f_0 - f_1 \cdot p \cdot \cos(2\pi \cdot t / p))]</td>
<td>Sinusoidally rising and falling tone with a period of p seconds</td>
</tr>
</tbody>
</table>

Here are a few examples:

```plaintext
rem Create a sine tone
set "F0=1000"               :: Frequency in Hz
set "T=5"                   :: Duration in seconds
set "OUT=out.wav"           :: Output filename

ffmpeg -f lavfi -i sine=%F0%:d=%T% -y %OUT%
pause
```
rem Sawtooth wave

set "F=550" :: Frequency in Hz
set "T=10" :: Duration in seconds
set "VOL=0.02" :: Volume
set "OUT=out.wav" :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*(2*mod(%F%*t,1)-1):c=mono:s=48000' -t %T% -y %OUT%

pause

rem Triangular wave

set "F=440" :: Frequency in Hz
set "T=10" :: Duration in seconds
set "VOL=0.2" :: Volume
set "OUT=out.wav" :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*(abs(4*mod(%F%*t,1)-2)-1):c=mono:s=48000' -t %T% -y %OUT%

pause

rem Linear chirp from 1kHz to 10kHz in 9 seconds

set "F0=1000" :: Start frequency in Hz
set "F1=10000" :: End frequency in Hz
set "P=9" :: Duration in seconds
set "VOL=0.1" :: Volume
set "OUT=out.wav" :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*sin(2*PI*t*(%F0%+t*(%F1%-%F0%)/(2*%P%))):c=stereo:s=44100' -t %P% -y %OUT%

pause
rem Linear chirp from 20Hz to 2kHz in 10 seconds
rem This is an approximated rectangular wave consisting of the fundamental wave and three overtones
rem This is also an example for the st() and ld() functions
rem First the phase is calculated and saved in variable "0", then the saved phase is used multiple times
rem to calculate the amplitudes of the fundamental wave and its overtones

set "F0=20"                 :: Start frequency in Hz
set "F1=2000"               :: End frequency in Hz
set "P=10"                  :: Duration in seconds
set "VOL=0.2"               :: Volume
set "OUT=out.wav"           :: Output filename

ffmpeg -f lavfi -i aevalsrc='st(0,'2*PI*t*(%F0%+t*(%F1%-%F0%)/(2*%P%));%VOL%*(sin(ld(0))
+sin(3*ld(0))/3+sin(5*ld(0))/5+sin(7*ld(0))/7):c=stereo:s=48000' -t %P% -y %OUT%
pause

rem Linear chirp from 20Hz to 2kHz in 10 seconds
rem This is a rectangular wave

set "FF=c:\ffmpeg\ffmpeg"   :: Path to FFmpeg
set "F0=20"                 :: Start frequency in Hz
set "F1=2000"               :: End frequency in Hz
set "P=10"                  :: Duration in seconds
set "VOL=0.2"               :: Volume
set "OUT=out.wav"           :: Output filename

%FF% -f lavfi -i aevalsrc='st(0,'2*PI*t*(%F0%+t*(%F1%-%F0%)/(2*%P%));%VOL%*lt(mod(ld(0),2*PI),PI):c=stereo:s=48000' -t %P% -y %OUT%
pause
rem Linear chirp from 20Hz to 10kHz in 10 seconds
rem This is an approximated triangular wave consisting of the fundamental wave and three overtones

set "F0=20" :: Start frequency in Hz
set "F1=10000" :: End frequency in Hz
set "P=10" :: Duration in seconds
set "VOL=0.2" :: Volume
set "OUT=out.wav" :: Output filename

ffmpeg -f lavfi -i aevalsrc='st(0,'2*PI*t*(%F0%+t*(%F1%-%F0%)/(2*%P%));%VOL%*(sin(ld(0))-sin(3*ld(0))/9+sin(5*ld(0))/25-sin(7*ld(0))/49):c=stereo:s=48000' -t %P% -y %OUT%
pause

rem Linear chirp from 20Hz to 2kHz in 10 seconds
rem The waveform is a needle impulse which a width of 1/25 of the period

set "F0=20" :: Start frequency in Hz
set "F1=2000" :: End frequency in Hz
set "W=1/25" :: Width of needle impulses
set "P=10" :: Duration in seconds
set "VOL=1" :: Volume
set "OUT=out.wav" :: Output filename

ffmpeg -f lavfi -i aevalsrc='st(0,'2*PI*t*(%F0%+t*(%F1%-%F0%)/(2*%P%));%VOL%*lt(mod(ld(0),2*PI),2*PI*%W):c=stereo:s=96000' -t %P% -y %OUT%
pause
rem Sinusoidally rising and falling tone from 400Hz to 1600Hz with a period of 2 seconds

set "F0=1000" :: Mid frequency in Hz
set "F1=600" :: Half of frequency sweep in Hz
set "P=2" :: Period in seconds
set "T=10" :: Duration in seconds
set "VOL=0.1" :: Volume
set "OUT=out.wav" :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*sin(2*PI*t*%F0%-%F1%*%P%*cos(2*PI*t/%P%)):c=stereo:s=44100' -t %T% -y %OUT%

pause
### 3.27 Create band-limited noise

```plaintext
rem Create band-limited noise from \( f_0 \) to \( f_1 \)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>F0</code></td>
<td><code>2000</code></td>
<td>Lower frequency in Hz</td>
</tr>
<tr>
<td><code>F1</code></td>
<td><code>4000</code></td>
<td>Upper frequency in Hz</td>
</tr>
<tr>
<td><code>SR</code></td>
<td><code>44100</code></td>
<td>Sample rate in Hz</td>
</tr>
<tr>
<td><code>WS</code></td>
<td><code>65536</code></td>
<td>Window size for FFT filter, bigger size reduces click noise</td>
</tr>
<tr>
<td><code>T</code></td>
<td><code>5</code></td>
<td>Duration in seconds</td>
</tr>
<tr>
<td><code>VOL</code></td>
<td><code>0.5</code></td>
<td>Volume</td>
</tr>
<tr>
<td><code>OUT</code></td>
<td><code>out.wav</code></td>
<td>Output filename</td>
</tr>
</tbody>
</table>

```ffmpeg -f lavfi -i anoisesrc=r=%SR%:a=%VOL%:d=%T% -af afftfilt='win_size=%WS%:real=re*between(0.5*sr*b/nb,%F0%,%F1%):imag=im*between(0.5*sr*b/nb,%F0%,%F1%)' -y %OUT%
```

This example sends the sound directly to FFplay:

```plaintext
rem Create band-limited noise for two bands from \( f_0 \) to \( f_1 \) and from \( f_2 \) to \( f_3 \)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>F0</code></td>
<td><code>500</code></td>
<td>Lower band start frequency in Hz</td>
</tr>
<tr>
<td><code>F1</code></td>
<td><code>600</code></td>
<td>Lower band stop frequency in Hz</td>
</tr>
<tr>
<td><code>F2</code></td>
<td><code>1000</code></td>
<td>Upper band start frequency in Hz</td>
</tr>
<tr>
<td><code>F3</code></td>
<td><code>1200</code></td>
<td>Upper band stop frequency in Hz</td>
</tr>
<tr>
<td><code>SR</code></td>
<td><code>44100</code></td>
<td>Sample rate in Hz</td>
</tr>
<tr>
<td><code>T</code></td>
<td><code>5</code></td>
<td>Duration in seconds</td>
</tr>
<tr>
<td><code>VOL</code></td>
<td><code>1</code></td>
<td>Volume</td>
</tr>
</tbody>
</table>

```ffmpeg -f lavfi -i anoisesrc=r=%SR%:a=%VOL%:d=%T% -af afftfilt='win_size=65536:real=re*bitor(between(0.5*sr*b/nb,%F0%,%F1%),between(0.5*sr*b/nb,%F2%,%F3%)):imag=im*bitor(between(0.5*sr*b/nb,%F0%,%F1%),between(0.5*sr*b/nb,%F2%,%F3%))' -f nut - | c:\ffmpeg\ffplay -autoexit -
```

```
```
rem Create band-limited noise for three bands from f0 to f1, from f2 to f3 and from f4 to f5

set "F0=500" :: Lower band start frequency in Hz
set "F1=550" :: Lower band stop frequency in Hz
set "F2=1000" :: Mid band start frequency in Hz
set "F3=1100" :: Mid band stop frequency in Hz
set "F4=1500" :: Upper band start frequency in Hz
set "F5=1650" :: Upper band stop frequency in Hz
set "SR=44100" :: Sample rate in Hz
set "T=5" :: Duration in seconds
set "VOL=1" :: Volume
set "OUT=out.wav" :: Output filename

ffmpeg -f lavfi -i anoisersrc=r=%SR%:a=%VOL%:d=%T% -af afftfilt='win_size=65536:real=re*bitor(bitor(between(0.5*sr*b/nb,%F0%,%F1%),between(0.5*sr*b/nb,%F2%,%F3%)),between(0.5*sr*b/nb,%F4%,%F5%)):imag=im*bitor(bitor(between(0.5*sr*b/nb,%F0%,%F1%),between(0.5*sr*b/nb,%F2%,%F3%)),between(0.5*sr*b/nb,%F4%,%F5%))' -y %OUT%
pause
3.28  Make an audio file with a short test tone

rem  Make a 10 seconds audio file with a short 3kHz tone at t=3s
ffmpeg -f lavfi -i sine=3000:duration=0.1 -af adelay=3000,apad -t 10 -y audio.wav
pause

3.29  Measuring the audio volume

The audio volume can be measured with the "volumedetect" filter, which doesn't change the input signal. The volume is written to the log output.
For example, this is useful if you have an unknown audio filter and want to know how much the volume is attenuated at a given frequency. In this case you would analyze the volume before and after applying the filter.

rem  Create a 5 seconds 2kHz audio file:
ffmpeg -f lavfi -i sine=2000 -t 5 -y test.wav

rem  Analyze the volume of this file:
ffmpeg -i test.wav -af volumedetect -f null NUL
pause

In this example the output format is "null" because no output file is required. It's also possible to replace "NUL" by "-" and this has the advantage that it runs under Linux and Windows:

ffmpeg -i test.wav -af volumedetect -f null -
3.30 Create an alternating left/right stereo sound

This batch file creates a sound file with this sequence:
Frequency F1 on left channel and silence on right channel for duration P/2, then silence on left channel and frequency F2 on right channel for duration P/2, then repeat.

```plaintext
set "P=0.5"                   :: Duration of one cycle in seconds
set "F1=1000"                 :: Frequency for left channel
set "F2=2000"                 :: Frequency for right channel
set "T=10"                    :: Duration in seconds

ffmpeg -f lavfi -i sine=%F1% -f lavfi -i sine=%F2% -filter_complex \\
  "[0]volume='lt(mod(t,%P%),%P%/2)':eval=frame[a];[1]volume='gte(mod(t,%P%),%P%/2)':eval=frame[b];[a][b]join=inputs=2:channel_layout=stereo" -t %T% -y out.wav

rem Alternatively you can also use this command line:

ffmpeg -f lavfi -i sine=%F1% -f lavfi -i sine=%F2% -filter_complex \\
  "[0]volume=0:enable='lt(mod(t,%P%),%P%/2)'[a];[1]volume=0:enable='gte(mod(t,%P%),%P%/2)'[b];[a][b]join=inputs=2:channel_layout=stereo" -t %T% -y out.wav
```

pause
3.31 Which equipment is useful for making sound records?

I use the following equipment:

- **TASCAM DR-70D recorder**, has 4 input channels, sample rate 44100, 48000, 96000Hz, 16 or 24 Bit
- The successor DR-701D has some improvements: The input amplifiers are somewhat less noisy, the four level controls can be electronically coupled, and the sampling rate can be up to 192kHz.
- **Rode NT1 microphone** with fur windshields (deadcats), very low-noise and excellent for quiet nature sounds
- Microphone cable in 5m length, so that you can stand a few meters away from the microphone when recording. If you stand too close to the microphone, you will have your own noise in the recording. You would hear every movement, every swallowing, every stomach growl...
- **HAMA Joy Powerbank 10400mAh**, as additional power supply for the recorder, because the built-in batteries only allow a very short recording time when the phantom power for the microphones is activated.
Comparison of Rode NT1 and NTG2 microphones

I did compare these microphones with a quite silent 1kHz sine source at about 2.5m distance. The NTG2 microphone was connected to the left channel and the NT1 microphone to the right channel of the TASCAM DR-701D recorder. Phantom power was set to 48V for both microphones. Sensitivity was set to "HI+" and the level control was turned fully clockwise. No fur windshields (deadcats) were used for this test. In a second measurement the sine source was switched off and the microphones were covered under a pillow in a silent room. The measured noise level is a combination of microphone noise and amplifier noise in the recorder.

<table>
<thead>
<tr>
<th>Microphone</th>
<th>Measured level of 1kHz sine</th>
<th>Measured noise level</th>
<th>Sensitivity in manufacturer's data sheet</th>
<th>Noise level in manufacturer's data sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rode NTG2</td>
<td>RMS level dB: -22.4</td>
<td>RMS level dB: -40.4</td>
<td>-36.0 dB re 1 Volt/Pascal +2dB @ 1kHz</td>
<td>18 dBA</td>
</tr>
<tr>
<td>Rode NT1</td>
<td>RMS level dB: -12.1</td>
<td>RMS level dB: -41.3</td>
<td>-29.0 dB re 1 Volt/Pascal +2dB @ 1kHz</td>
<td>4 dBA</td>
</tr>
</tbody>
</table>

Result: The NT1 gives about 10dB more signal than the NTG2, at about the same noise level. For recording of quiet nature sounds the NT1 is clearly the superior microphone.

This batch file was used for the analysis:

```bash
ffmpeg -ss 5 -i 1kHz.wav -map_channel 0.0.0 -t 10 NTG2_1kHz.wav -map_channel 0.0.1 -t 3 -y NT1_1kHz.wav
ffmpeg -ss 5 -i silence.wav -map_channel 0.0.0 -t 10 NTG2_silence.wav -map_channel 0.0.1 -t 3 -y NT1_silence.wav
ffmpeg -i NTG2_1kHz.wav -af astats=metadata=1 -y out_NTG2_1kHz.wav
ffmpeg -i NT1_1kHz.wav -af astats=metadata=1 -y out_NT1_1kHz.wav
ffmpeg -i NTG2_silence.wav -af astats=metadata=1 -y out_NTG2_silence.wav
ffmpeg -i NT1_silence.wav -af astats=metadata=1 -y out_NT1_silence.wav
pause
```
Mathematical properties of sample rates 44100 and 48000

44100 = 2^2 * 3^2 * 5^2 * 7^2  48000 = 2^7 * 3^1 * 5^3  The greatest common divisor of 44100 and 48000 is 300.

Divisors of 48000:
1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 25, 30, 32, 40, 48, 50, 60, 64, 75, 80, 96, 100, 120, 125, 128, 150, 160, 192, 200, 240, 250, 300, 320, 375, 384, 400, 480, 500, 600, 640, 750, 800, 960, 1000, 1200, 1500, 1600, 1920, 2000, 2400, 3000, 3200, 4000, 4800, 6000, 8000, 9600, 12000, 16000, 24000, 48000

Divisors of 44100:
1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 14, 15, 18, 20, 21, 25, 28, 30, 35, 36, 42, 45, 49, 50, 60, 63, 70, 75, 84, 90, 98, 100, 105, 126, 140, 147, 150, 175, 180, 196, 210, 225, 245, 252, 294, 300, 315, 350, 420, 441, 450, 490, 525, 588, 630, 700, 735, 882, 900, 980, 1050, 1225, 1260, 1470, 1575, 1764, 2100, 2205, 2450, 2940, 3150, 3675, 4410, 4900, 6300, 7350, 8820, 11025, 14700, 22050, 44100

Divisors of 9600:
1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 25, 30, 32, 40, 48, 50, 60, 64, 75, 80, 96, 100, 120, 128, 150, 160, 192, 200, 240, 300, 320, 384, 400, 480, 600, 640, 800, 960, 1200, 1600, 1920, 2400, 3200, 4800, 9600

Divisors of 8820:
1, 2, 3, 4, 5, 6, 7, 9, 10, 12, 14, 15, 18, 20, 21, 28, 30, 35, 36, 42, 45, 49, 60, 63, 70, 84, 90, 98, 105, 126, 140, 147, 150, 175, 180, 196, 210, 225, 245, 252, 294, 315, 420, 441, 490, 588, 630, 735, 882, 980, 1260, 1470, 1764, 2205, 2940, 4410, 8820

Divisors of 4800:
1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 18, 20, 24, 25, 30, 32, 40, 48, 50, 60, 64, 75, 80, 96, 100, 120, 150, 160, 192, 200, 240, 300, 320, 400, 480, 600, 800, 960, 1200, 1600, 2400, 4800

Divisors of 4410:
1, 2, 3, 5, 6, 7, 9, 10, 14, 15, 18, 21, 30, 35, 42, 45, 49, 63, 70, 90, 98, 105, 126, 147, 210, 245, 294, 315, 441, 490, 630, 735, 882, 1470, 2205, 4410

324
<table>
<thead>
<tr>
<th>Framerate</th>
<th>Ticks of 1/90000s clock (Mpeg TS system clock timebase)</th>
<th>Audio samples per frame @44.1 kHz</th>
<th>Audio samples per frame @48 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.976 fps = 24 / 1.001</td>
<td>3753.75</td>
<td>1839.3375</td>
<td>2002</td>
</tr>
<tr>
<td>24 fps</td>
<td>3750</td>
<td>1837.5</td>
<td>2000</td>
</tr>
<tr>
<td>25 fps</td>
<td>3600</td>
<td>1764</td>
<td>1920</td>
</tr>
<tr>
<td>29.97 fps = 30 / 1.001</td>
<td>3003</td>
<td>1471.47</td>
<td>1601.6</td>
</tr>
<tr>
<td>50 fps</td>
<td>1800</td>
<td>882</td>
<td>960</td>
</tr>
<tr>
<td>59.94 fps = 60 / 1.001</td>
<td>1501.5</td>
<td>735.735</td>
<td>800.8</td>
</tr>
</tbody>
</table>
3.34 Speed of sound

Speed of sound in air at 20°C: 343.2 m/s

<table>
<thead>
<tr>
<th>Time [ms]</th>
<th>Distance [m]</th>
<th>Samples @ 44100Hz</th>
<th>Samples @ 48000Hz</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.022675</td>
<td>0.00778</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.020833</td>
<td>0.00715</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.3432</td>
<td>44.1</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.6864</td>
<td>88.2</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>2.2675</td>
<td>0.7782</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.9138</td>
<td>1</td>
<td>128.497</td>
<td>139.860</td>
<td></td>
</tr>
<tr>
<td>3.3333</td>
<td>1.144</td>
<td>147</td>
<td>160</td>
<td>This is the smallest possible time interval where the number of samples is an integer at sample rate 44100 Hz and also at 48000 Hz</td>
</tr>
<tr>
<td>5</td>
<td>1.716</td>
<td>220.5</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>5.8275</td>
<td>2</td>
<td>256.699</td>
<td>279.720</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3.432</td>
<td>441</td>
<td>480</td>
<td>Both sample numbers are integers</td>
</tr>
<tr>
<td>14.569</td>
<td>5</td>
<td>642.483</td>
<td>699.301</td>
<td></td>
</tr>
<tr>
<td>29.138</td>
<td>10</td>
<td>1284.97</td>
<td>1398.60</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>34.32</td>
<td>4410</td>
<td>4800</td>
<td>Both sample numbers are integers</td>
</tr>
<tr>
<td>1000</td>
<td>343.2</td>
<td>44100</td>
<td>48000</td>
<td>Both sample numbers are integers</td>
</tr>
</tbody>
</table>

300Hz is the largest frequency where one cycle consists of an integer number of samples at 44100Hz and also at 48000Hz sample rate.
4 FFprobe

How to examine a video file with FFprobe without having to write the name of the video into a batch file each time?

It's very simple, just create this batch file once and put it on your desktop:

```bash
ffprobe %1
pause
```

Now you can simply drag the video you want to examine with the mouse onto the icon of this batch file, and you will immediately see the result without having pressed a single key. The parameter %1 causes the file name to be passed to FFprobe.

See also: [https://trac.ffmpeg.org/wiki/FFprobeTips](https://trac.ffmpeg.org/wiki/FFprobeTips)

By the way, it's also possible to let FFmpeg examine a file.

To see whether FFmpeg recognizes the file as something:

```bash
ffmpeg -i myfile.xxx
pause
```

To see whether FFmpeg can decode the file:

```bash
ffmpeg -i myfile.xxx -f null -
pause
```
## 5 FFplay

Keyboard commands while playing:

<table>
<thead>
<tr>
<th>Key/f Generator</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>q, ESC</strong></td>
<td><strong>Quit</strong></td>
</tr>
<tr>
<td><strong>f, left mouse double-click</strong></td>
<td><strong>Toggle full screen</strong></td>
</tr>
<tr>
<td><strong>p, SPACE</strong></td>
<td><strong>Pause</strong></td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>Step to the next frame. Pause if the stream is not already paused, step to the next video frame, and pause.</td>
</tr>
<tr>
<td><strong>m</strong></td>
<td>Toggle mute.</td>
</tr>
<tr>
<td><strong>9, 0</strong></td>
<td>Decrease and increase volume respectively.</td>
</tr>
<tr>
<td>**/ , ***</td>
<td>Decrease and increase volume respectively.</td>
</tr>
<tr>
<td><strong>a</strong></td>
<td>Cycle audio channel in the current program.</td>
</tr>
<tr>
<td><strong>v</strong></td>
<td>Cycle video channel.</td>
</tr>
<tr>
<td><strong>t</strong></td>
<td>Cycle subtitle channel in the current program.</td>
</tr>
<tr>
<td><strong>c</strong></td>
<td>Cycle program.</td>
</tr>
<tr>
<td><strong>w</strong></td>
<td>Cycle video filters or show modes.</td>
</tr>
<tr>
<td><strong>left/right</strong></td>
<td>Seek backward/forward 10 seconds.</td>
</tr>
<tr>
<td><strong>down/up</strong></td>
<td>Seek backward/forward 1 minute.</td>
</tr>
<tr>
<td><strong>page down/page up</strong></td>
<td>Seek to the previous/next chapter, or if there are no chapters seek backward/forward 10 minutes.</td>
</tr>
<tr>
<td><strong>right mouse click</strong></td>
<td>Seek to percentage in file corresponding to fraction of width.</td>
</tr>
</tbody>
</table>

This is a batch file that you can put on your desktop, and then play a video simply by drag-and-drop:

```bash
ffplay %1 -autoexit
```
Note: Contrary to FFmpeg, FFplay doesn’t need "-i" before the input.

Note: FFplay supports only one input. It doesn’t support "filter_complex".

This is a batch file for playing audio files by drag-and-drop (without video output):

```
ffplay %1 -nodisp -autoexit
```

List of the most important FFplay options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-s size</code></td>
<td>Set frame size (WxH or abbreviation)</td>
</tr>
<tr>
<td><code>-fs</code></td>
<td>Start in fullscreen mode</td>
</tr>
<tr>
<td><code>-an</code></td>
<td>Disable audio</td>
</tr>
<tr>
<td><code>-vn</code></td>
<td>Disable video</td>
</tr>
<tr>
<td><code>-ss pos</code></td>
<td>Seek to pos</td>
</tr>
<tr>
<td><code>-nodisp</code></td>
<td>Disable graphical display</td>
</tr>
<tr>
<td><code>-noborder</code></td>
<td>Borderless window</td>
</tr>
<tr>
<td><code>-alwaysontop</code></td>
<td>Window always on top</td>
</tr>
<tr>
<td><code>-f fmt</code></td>
<td>Force format</td>
</tr>
<tr>
<td><code>-loop number</code></td>
<td>Loops movie playback <code>&lt;number&gt;</code> times. 0 means forever</td>
</tr>
<tr>
<td><code>-vf filtergraph</code></td>
<td>Create the filtergraph specified by <code>filtergraph</code> and use it to filter the video stream</td>
</tr>
<tr>
<td><code>-af filtergraph</code></td>
<td><code>filtergraph</code> is a description of the filtergraph to apply to the input audio</td>
</tr>
<tr>
<td><code>-autoexit</code></td>
<td>Exit when video is done playing</td>
</tr>
<tr>
<td><code>-exitonkeydown</code></td>
<td>Exit if any key is pressed</td>
</tr>
<tr>
<td><code>-exitonmousedown</code></td>
<td>Exit if any mouse button is pressed</td>
</tr>
</tbody>
</table>
6 Exiftool

With this tool you can show all exif data that are contained in pictures or videos.

https://exiftool.org/

Usage is very simple if you create this batch file once and put it on your desktop:

```bash
exiftool %1
pause
```

Now you can simply drag the picture or video you want to examine with the mouse onto the icon of this batch file, and you will immediately see the result without having pressed a single key. The parameter %1 causes the file name to be passed to Exiftool.

Exiftool can be combined with the batch command "findstr", if you want to filter only a few lines from the large Exiftool output:

```bash
@echo off
exiftool %1 | findstr /C:"File Name" /C:"File Size" /C:"Duration" /C:"Image Width" /C:"Image Height" /C:"Video Frame Rate" /C:"Exposure Time" /C:"F Number" /C:"Exposure Program" /C:"ISO" /C:"Photo Style" /B /C:"Noise Reduction" /C:"Contrast " /C:"Saturation" /C:"Sharpness" /C:"Avg Bitrate" /C:"Track Create Date"
pause
```

"findstr" is in detail explained here: https://ss64.com/nt/findstr.html

Exiftool does also list "Internal Serial Number" and "Lens Serial Number", however in both cases the listed numbers don't agree with the serial numbers printed on my GH5S and Leica DG 12-60mm f/2.8-4.0.

330
7 IrfanView

How to show the mouse coordinates:
Options → Properties/Settings → Viewing → Show_mouse_coordinates_in_the_Statusbar

8 Adobe DNG Converter

This is a tool for converting RAW images from many different cameras to the DNG format, which has a lossless compression.
Note: You can only specify the folder and not the images. It's normal that you don't see the images. Make sure that you click on "Convert" and not on "Extract".
9 Batch files (Windows 7 and 10)

Some useful links for writing batch files:

https://en.wikibooks.org/wiki/Windows_Batch_Scripting  (english)
https://de.wikibooks.org/wiki/Batch-Programmierung/_Druckversion  (german)
https://ss64.com/nt/
https://ss64.com/nt/syntax.html

9.1 Wildcards in filenames

* any sequence of one or more characters  
? a single character other than a period "."

When a command-line argument contains a filename, a special syntax can be used to get various information about this file:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Result</th>
<th>Example for F:\Meteors_2019\CUT00380.MOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>%1</td>
<td></td>
<td>CUT00380.MOV</td>
</tr>
<tr>
<td>%~1</td>
<td>%1 with no enclosing quotation marks</td>
<td>CUT00380.MOV</td>
</tr>
<tr>
<td>%~f1</td>
<td>Full path with a drive letter</td>
<td>F:\Meteors_2019\CUT00380.MOV</td>
</tr>
<tr>
<td>%~d1</td>
<td>Drive letter</td>
<td>F:</td>
</tr>
<tr>
<td>%~p1</td>
<td>Drive-less path with the trailing backslash</td>
<td>\Meteors_2019\</td>
</tr>
<tr>
<td>%~n1</td>
<td>For a file, the file name without path and extension For a folder, the folder name</td>
<td>CUT00380</td>
</tr>
<tr>
<td>%~x1</td>
<td>File name extension including the period</td>
<td>.MOV</td>
</tr>
</tbody>
</table>

The same syntax applies to single-letter variables created by the FOR command.
Change the extension of a filename in a batch file:

```
set OLDFILENAME=%1
set NEWFILENAME=%OLDFILENAME:MOV=MP4%
pause
```

Please note that all instances of "MOV" will be replaced by "MP4". This fails if "MOV" is part of the path or filename, as in "MOVEMENT.MOV"

9.2 Create beeps in a batch file

```
@echo #
@timeout 1
@echo #
@timeout 1
@echo #
```

In this example the # stands for the non-printable character (ASCII code 7), which you can't enter with Notepad. You can type any other character instead and later use a hex editor to replace it by 0x07.

Another way for creating the ASCII 7 is to type this command line at the command prompt:

```
echo @echo ^G>test33.bat
```

where ^G means typing CTRL G

This is an endless loop for beeping every 10 seconds, without any output on the screen (except a line feed):

```
:beep
@echo #
@timeout 10 > nul
@goto :beep
```
9.3  Loop over all files in a directory

```plaintext
for %f in (img*.jpg) do call :for_body %f
goto :the_end
:for_body
    ffmpeg -i %1 -y %~n1.png
exit /b
:the_end
pause
```

Note: "goto :the_end" can be replaced by "goto :eof" which is a predefined label at the end of the file. In this case it's unnecessary to write ":eof" at the end.

9.4  Create short text files or append text to a file

```plaintext
echo Hello ! > test.txt
echo This is the 2nd line>> test.txt
```

Note: The first line (with ">") creates a new file or overwrites an existing file. The second line (with ">>") appends text to an already existing file.
9.5 if Conditions

| if %MODE%==1 echo test1
| pause |

This doesn’t work because the variable "MODE" is undefined. The left side of the comparison is empty. The batch file will immediately exit in the first line without any error message.

To avoid this problem, you can add two dots to each side of the comparison (thanks to Dan Bridges for this idea):

| if %MODE%==1. echo test2
| pause |

In this case the left side isn't empty. You won't see the "test2" echo because the left and right sides aren't equal. The batch file won't exit in the first line and will wait for a keypress in the second line.
VLC Player

https://www.videolan.org/vlc/

My notebook doesn't have enough computing power for playing 4K videos (400Mbit/s from Panasonic GH5S) smoothly with VLC player. This batch file reduces the size to 50% and then plays the video.

Simply drag and drop the video on the batch file's icon. The parameter %1 causes the file name to be passed to FFmpeg.

In the second line the path must be written in quotation marks because there is a space character in "Program Files".

```bash
ffmpeg -i %1 -vf scale=w=iw/2:h=ih/2 -y half_size.mp4
"c:\Program Files\VideoLAN\VLC\vlc.exe" half_size.mp4
```

This is a subset of VLC’s keyboard hotkeys:

<table>
<thead>
<tr>
<th>Key</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Toggle fullscreen</td>
</tr>
<tr>
<td>ESC</td>
<td>Leave fullscreen/close dialogue</td>
</tr>
<tr>
<td>space</td>
<td>Play/pause</td>
</tr>
<tr>
<td>E</td>
<td>Next frame</td>
</tr>
<tr>
<td>+</td>
<td>Faster</td>
</tr>
<tr>
<td>-</td>
<td>Slower</td>
</tr>
<tr>
<td>=</td>
<td>Normal rate</td>
</tr>
<tr>
<td>]</td>
<td>Faster (fine)</td>
</tr>
<tr>
<td>[</td>
<td>Slower (fine)</td>
</tr>
<tr>
<td>S</td>
<td>Stop</td>
</tr>
<tr>
<td>T</td>
<td>Position/time</td>
</tr>
<tr>
<td>Ctrl + Q</td>
<td>Quit</td>
</tr>
<tr>
<td>M</td>
<td>Mute</td>
</tr>
<tr>
<td>V</td>
<td>Cycle subtitle track</td>
</tr>
<tr>
<td>Shift + s</td>
<td>Make a snapshot picture</td>
</tr>
</tbody>
</table>
11 Color grading with 3D LUT Creator

3D LUT Creator is a software for color grading. A free demo version is available, and the full version costs $249 (July 2019). Sometimes there seems to be 25% discount.

Website: https://3dlutcreator.com/

Drawback of this software: The written manual is totally outdated, and for the latest version you have to watch many video tutorials (what I don't like so much).

All video tutorials can be found here: https://3dlutcreator.com/3d-lut-creator---tutorials.html

A guide for all hotkeys in 3D LUT Creator:
https://medium.com/@alexeyadamitsky/3d-lut-creator-ultimate-hotkeys-guide-17a32f957077

This is a very powerful software for creating and editing color-look-up-tables. It's also possible to match the colors to a ColorChecker.

A ColorChecker is a card with 24 different colors, which is hold in the camera's field of view.

Original X-Rite ColorChecker:
https://www.xrite.com/categories/calibration-profiling/colorchecker-targets

There are also cheap ($20-$25) chinese ColorCheckers available. Their colors may be a little bit different from the original.

Hotkeys:
CTRL N  New Preset (Reset All)
CTRL O  Load an image
CTRL E  Save the LUT
CTRL +  Zoom in
CTRL -  Zoom out
I'd like to describe the workflow for making a video with a ColorChecker, and how to create a LUT and apply that LUT to the video.

Step 1: Make a video where at some time the ColorChecker is visible, preferably at the beginning of the video. It doesn't care which file format you use, as FFmpeg can decode almost all of them.

Step 2: Open the video in a viewer (for example VLC player) and check at which time the ColorChecker is visible.

Step 3: Use this batch file to extract a single picture from the video and save it lossless as PNG. Of course, you must enter your filename and the correct time in the batch file. The picture will be automatically 8-bit or 16-bit PNG, depending on the bit depth of the input video, so that there is no loss of quality.

```batch
set "IN=my_video.mov"        :: Input video
set "T=3"                    :: Time in seconds, where picture shall be extracted
ffmpeg -ss %T% -i %IN% -frames 1 -y picture.png
pause
```

In the previous example I did use variables, because they make the batch file more readable. The following batch file is doing exactly the same thing:

```batch
ffmpeg -ss 3 -i my_video.mov -frames 1 -y picture.png
pause
```

The -y option means that the output file will be overwritten if it already exists (without asking). Without the -y option, FFmpeg would ask before overwriting.

The pause command means that you have to press a button before the window closes. Without this command the window would close immediately when FFmpeg is finished, making it impossible to see if there were any error messages.

Step 4: Drag and drop this picture into 3D LUT Creator (or alternatively press CTRL+O). In the field to the left of the "Match" button select "Curves+3DLUT". Click on the "Color Chart Grid Tool" icon (this is a small rectangle with 6 dots in it). Move the corners of the grid tool with the mouse to the corresponding ColorChecker fields in the picture. Then click on "Match". After a few seconds the picture should be shown with all colors.
matched to the ColorChecker. Click on "Save 3DLUT" in the bottom left corner (or alternatively press CTRL+E) to save the CLUT as my_lut.cube.

**Step 5:** Use this batch file to apply the LUT to your video:

```bash
set "IN=my_video.mov"  :: Input video
set "LUT=my_lut.cube"   :: Look-Up-Table
set "OUT=out.mov"      :: Output video

ffmpeg -i %IN% -vf lut3d="%LUT%" -y %OUT%

pause
```

If you want to see only a few seconds at the beginning of the video, you can limit the length with the `-t` parameter.

```bash
set "IN=my_video.mov"  :: Input video
set "LUT=my_lut.cube"   :: Look-Up-Table
set "OUT=out.mov"      :: Output video
set "T=10"             :: Length of output video

ffmpeg -i %IN% -vf lut3d="%LUT%" -t %T% -y %OUT%

pause
```

You can also show the video simultaneously before and after applying the LUT. The scale filter is used to reduce the size to 50%, and the hstack filter is used to combine two videos side by side.

```bash
set "IN=my_video.mov"  :: Input video
set "LUT=my_lut.cube"   :: Look-Up-Table
set "OUT=out.mov"      :: Output video
set "T=10"             :: Length of output video

ffmpeg -i %IN% -filter_complex scale=iw/2:ih/2,split[a][b];[b]lut3d="%LUT%" [c];[a][c]hstack -t %T% -y %OUT%

pause
```

Some notes about paths and filenames under Windows:
This doesn't work because the colon ":" isn't escaped properly and the filter regards the path as two separate filter parameters:

<table>
<thead>
<tr>
<th>Lut3d=C:/test/test.cube</th>
</tr>
</thead>
</table>

Instead use one of these solutions:

<table>
<thead>
<tr>
<th>Lut3d=C::\test\test.cube</th>
<th>// don't use '' but escape twice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lut3d='C::\test\test.cube'</td>
<td>// use '' and escape once</td>
</tr>
</tbody>
</table>

Here is a converter for different types of color-look-up-tables:


This software can also fit colors to a ColorChecker:

[https://www.xrite.com/service-support/downloads/c/colorchecker_camera_calibration_v1_1_1](https://www.xrite.com/service-support/downloads/c/colorchecker_camera_calibration_v1_1_1)

But it has two severe drawbacks:
1. It requires a DNG image as input. That's no problem for photography, but there's no way to extract a DNG from a video.
2. The output of this software is a camera profile in Adobe *.dcp format and I don't know how this can be converted into a CLUT for FFmpeg.

### 11.1 Known problems and solutions

Problem: Error message "MSVCP120.dll" and "MSVCR120.dll" missing when running 3D Lut Creator on Windows 10.
Solution: Install either Visual Studio 2013, or the Visual-C-2013-Redistributable Package

Problem: When you drag and drop an image to 3D Lut Creator, the image doesn't become visible in 3D Lut Creator.
Solution: Preferences --> Integration --> Remove the tick at "Enable Open CL"
Another tool for color grading is "Grossgrade" which is available here: https://grossgrade.com/en/
11.2   Beginner tutorials for 3D LUT Creator

1. What is the LUT?   [https://www.youtube.com/watch?time_continue=2&v=3ZpbUOGDWLE](https://www.youtube.com/watch?time_continue=2&v=3ZpbUOGDWLE)
   This video explains 1D and 3D LUTs. 3D LUT Creator can save LUTs in many different formats, including *.cube and *.png

2. Working with LUTs in Photoshop and 3D LUT Creator   [https://www.youtube.com/watch?time_continue=7&v=K0t-HSO-OUU](https://www.youtube.com/watch?time_continue=7&v=K0t-HSO-OUU)

3. Working with Lightroom and 3D LUT Creator   [https://www.youtube.com/watch?v=o968FH1kV3w](https://www.youtube.com/watch?v=o968FH1kV3w)

4. Working with the Image window   [https://www.youtube.com/watch?v=TmZAITX5tfU](https://www.youtube.com/watch?v=TmZAITX5tfU)
   The working image can be loaded by drag and drop.
   For an optional reference image use "File / Load Reference Image".
   In the video he says you can toggle between working image and reference image by pressing the = key, but that's wrong. It's the + key.
   CTRL + Zoom in   CTRL - Zoom out
   For moving the image in the window, press the mouse wheel and move the mouse.++
   "Image / Assign Color Profile" doesn't change the image, it changes only the way how the image is displayed.
   "Image / Convert to Profile" does change the image.
   Toggle the compare mode by pressing the c key.
   By pressing the x key the image is split in the middle, one half is before and the other is after.
   By pressing the v key you can toggle between horizontal and vertical splitting.

5. Look manager   [https://www.youtube.com/watch?v=dY_6MeE-qAg](https://www.youtube.com/watch?v=dY_6MeE-qAg)
   Window / Look Manager
   Open a folder to see all presets applied to your image.
   The size of the images can be changed with the + and - buttons in the top left corner.

6. Working principle of A/B and C/L color grids   [https://www.youtube.com/watch?v=AI5SYkJdDdqs](https://www.youtube.com/watch?v=AI5SYkJdDdqs)
In the A/B grid we can change only hue (position angle) and saturation (radial distance from center). Lightness stays unchanged.
In the C/L grids we can change only saturation (left - right, neutral in center) and lightness (vertical).

7. HSP and LAB color models  https://www.youtube.com/watch?v=mJfEgvheWeM
   In this video he explains the difference between the different HSP color models, and which model to use for which purpose.

8. LXY, MXY, MABe, MXYe, SXY, YUV, CMYK and RGBW color models  https://www.youtube.com/watch?v=7uC1vtS1BnU

9. The Luminance curve and the Brightness slider  https://www.youtube.com/watch?v=BBjY3ivCjPg

10. Saturation curves  https://www.youtube.com/watch?v=TnUp3Dsb_DU

11. Basics of working with the A/B color grid  https://www.youtube.com/watch?v=35EoR_c4D9w

12. Practice with A/B color grid, part 1  https://www.youtube.com/watch?v=BYe_V0UF5os

13. Practice with A/B color grid, part 2  https://www.youtube.com/watch?v=dR4pjHRpU0Y

14. Tools for working with the A/B color grid  https://www.youtube.com/watch?v=etIX_e8-_lk

15. Batch processing in 3D LUT Creator  https://www.youtube.com/watch?v=1wv1NqXywiY

11.3 Advanced tutorials for 3D LUT Creator

1. Color Match with the Reference image  https://youtu.be/k0YQNm7TINM

2. How to create 3D LUT files from Lightroom presets or Photoshop Plugins  https://youtu.be/MOiEciUIISU

3. RAW photo developing with 3D LUT Creator  https://youtu.be/3Sm120XC37Q

4. Skin tone with CMYK and 2D-curves  https://youtu.be/W6PYOKvo3rl

5. Teal & Orange grading in 3D LUT Creator  https://youtu.be/XQezpVjCUsI

6. How to change third party LUTs in 3D LUT Creator  https://youtu.be/Lx6ppOm9kCY

7. How to darken the sky with no artifacts?  https://youtu.be/uxiTsm80Xho

343
8. Blend Modes in 3D LUT Creator  https://youtu.be/SKvZg_ZdI9M
10. Color correction in game production with 3D LUT Creator & Unity  https://youtu.be/pzJXtyseARo
11. Skin tone color correction by numbers with RGB curves  https://youtu.be/NyJXdpJDPU
12. Adjusting the Skin Tone using color match tool  https://youtu.be/rgVFTuu9Kls
13. Color Masks  https://youtu.be/rQHooXewsN0

11.4 Working with Color Targets in for 3D LUT Creator

1. Color correction with Color Checkers in 3D LUT Creator, part 1  https://youtu.be/mZvrj8__5r0
2. Color correction with Color Checkers in 3D LUT Creator, part 2  https://youtu.be/0UALWETt1q4
3. Working with ChromaDuMonde in 3D LUT Creator and Davinci Resolve  https://youtu.be/5oCS4WqPKB8

11.5 Working with video in for 3D LUT Creator

1. Working with LOG video footage in 3D LUT Creator  https://youtu.be/jX3i34wFsG0
2. Using 3D LUT Creator with Davinci Resolve & Red Camera Footage  https://youtu.be/4e4OrN60_wc
3. Working with ChromaDuMonde in 3D LUT Creator and Davinci Resolve  https://youtu.be/5oCS4WqPKB8
This is the custom ColorChecker file for the cheap Chinese ColorChecker, using the RGB values printed on the back side and converted to LAB. Save this file as "MyColorChecker.txt". The differences to the original X-Rite ColorChecker seem to be quite small.

<table>
<thead>
<tr>
<th>NUMBER_OF_SETS</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>LGOROWLENGTH</td>
<td>6</td>
</tr>
<tr>
<td>INFO</td>
<td>&quot;sRGB&quot;</td>
</tr>
<tr>
<td>INSTRUMENTATION</td>
<td>&quot;3DLUTCreator&quot;</td>
</tr>
<tr>
<td>BEGIN_DATA_FORMAT</td>
<td></td>
</tr>
<tr>
<td>SampleID</td>
<td>SAMPLE_NAME</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>A1</td>
</tr>
<tr>
<td>2</td>
<td>A2</td>
</tr>
<tr>
<td>3</td>
<td>A3</td>
</tr>
<tr>
<td>4</td>
<td>A4</td>
</tr>
<tr>
<td>5</td>
<td>A5</td>
</tr>
<tr>
<td>6</td>
<td>A6</td>
</tr>
<tr>
<td>7</td>
<td>B1</td>
</tr>
<tr>
<td>8</td>
<td>B2</td>
</tr>
<tr>
<td>9</td>
<td>B3</td>
</tr>
<tr>
<td>10</td>
<td>B4</td>
</tr>
<tr>
<td>11</td>
<td>B5</td>
</tr>
<tr>
<td>12</td>
<td>B6</td>
</tr>
<tr>
<td>13</td>
<td>C1</td>
</tr>
<tr>
<td>14</td>
<td>C2</td>
</tr>
<tr>
<td>15</td>
<td>C3</td>
</tr>
<tr>
<td>16</td>
<td>C4</td>
</tr>
<tr>
<td>17</td>
<td>C5</td>
</tr>
<tr>
<td>18</td>
<td>C6</td>
</tr>
<tr>
<td>19</td>
<td>D1</td>
</tr>
<tr>
<td>20</td>
<td>D2</td>
</tr>
<tr>
<td>21</td>
<td>D3</td>
</tr>
<tr>
<td>22</td>
<td>D4</td>
</tr>
<tr>
<td>23</td>
<td>D5</td>
</tr>
<tr>
<td>24</td>
<td>D6</td>
</tr>
</tbody>
</table>

Color converter for different color spaces: [https://www.nixsensor.com/free-color-converter/]
The software looks very interesting, but has so many functions that you don't know where to begin.

https://www.blackmagicdesign.com/de/products/davinciresolve/

I got this book: Paul Saccone, Dion Scoppetuolo: "Der ultimative Leitfaden zu DaVinci Resolve 15" (I got the german translation, but it's also available in english).

Please note that this book is for version 15. Version 16 seems to have a different user interface, so for learning with this book it's better to use the older version 15. The official manual for version 16 is extremely long, more than 3000 pages.
12.1 Convert 10-bit videos from GH5S for DaVinci Resolve

The free DaVinci Resolve 15 (and 16) version can't import 4K 4:2:2 10 bit videos (from Panasonic GH5S). But this FFmpeg conversion does the job:

```bash
ffmpeg -i P1000975.MOV -map_metadata 0 -pix_fmt yuv422p10le -c:v dnxhd -profile:v 4 -c:a pcm_s24le -color_range pc -movflags write_colr out.mov
```

What's the meaning of the options in this command line?

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c:v dnxhd</td>
<td>Video codec: &quot;DNxHD&quot; (Digital Nonlinear Extensible High Definition) You won't find much about this in FFmpeg documentation.</td>
</tr>
<tr>
<td>-profile:v 4</td>
<td>Can be in range [0..5], see below output of ffmpeg -h encoder=dnxhd</td>
</tr>
<tr>
<td>-c:a pcm_s24le</td>
<td>Audio codec: &quot;PCM signed 24-bit little-endian&quot;</td>
</tr>
<tr>
<td>-color_range pc</td>
<td>Set the color range in the stream, 'tv' means limited range and 'pc' means full range.</td>
</tr>
<tr>
<td>-movflags write_colr</td>
<td>This is an option of the &quot;mov&quot; muxer, see output of ffmpeg -h muxer=mov</td>
</tr>
</tbody>
</table>

Output of ffmpeg -h encoder=dnxhd

Encoder dnxhd [VC3/DNxHD]:
- General capabilities: threads
- Threading capabilities: frame and slice
- Supported pixel formats: yuv422p yuv422p10le yuv444p10le gbrp10le
dnxhd AVOptions:
- -nitris_compat <boolean> E..V....... encode with Avid Nitris compatibility (default false)
- -ibias <int> E..V....... intra quant bias (from INT_MIN to INT_MAX) (default 0)
- -profile <int> E..V....... (from 0 to 5) (default dnxhd)
  dnxhd 0 E..V....... 
dnxhr_444 5 E..V....... 
dnxhr_hqx 4 E..V....... 
dnxhr_hq 3 E..V....... 
dnxhr_sq 2 E..V....... 
dnxhr_lb 1 E..V....... 

347
You can also apply the VLog-L to Rec709 LUT in the same command line:

```bash
ffmpeg -i P1000975.MOV -vf lut3d="VLog_to_V709.cube" -map_metadata 0 -pix_fmt yuv422p10le -c:v dnxhd -profile:v 4 -c:a pcm_s24le -color_range pc -movflags write_colr out.mov
```

The LUT file can be downloaded here: [https://av.jpn.support.panasonic.com/support/global/cs/dsc/download/lut/index.html](https://av.jpn.support.panasonic.com/support/global/cs/dsc/download/lut/index.html)

Another VLog-L to Rec709 LUT is available here: [https://nickdriftwood.com/product/driftwood-v-log-v-gamut-sample-lut](https://nickdriftwood.com/product/driftwood-v-log-v-gamut-sample-lut)

This batch file is doing the same thing, but you can drag and video over it’s icon on the desktop and it will automatically generate a renamed (P100*.* → C*.*) and converted video in the same folder where the input video came from:

```batch
set "INPUT=%1"                                          :: Input by drag-and-drop
set "OUTPUT=%INPUT:P100=C%"                             :: Specify how to rename the video
set "LUT=C:\Users\astro\Desktop\VLog_to_V709.cube"  :: Full path of *.cube LUT file,
                                                      :: use either single forward slashes or double backslashes!
rem Rename P100*. to C.*, apply LUT VLog to V709 and convert for DaVinci Resolve
ffmpeg -i %INPUT% -vf lut3d=%LUT% -map_metadata 0 -pix_fmt yuv422p10le -c:v dnxhd -profile:v 4 -c:a pcm_s24le -color_range pc -movflags write_colr -y %OUTPUT%
pause
```

Note: If the LUT file is in the same folder as the input video, you can simply use lut3d=%LUT% but if it's in another folder, then you must specify the full path and use this tricky and hard to understand escaping: lut3d=%LUT%'

Apply the LUT and convert all P100*.MOV videos in the current folder and save the converted videos as C*.MOV, so that they become readable by the free DaVinci Resolve version. This works fine for FHD and 4K videos:

```batch
rem Convert all videos in current folder for DaVinci Resolve
for %%f in (P1*.MOV) do call :for_body %%f
goto :the_end
```

348
set INPUT=%1
set OUTPUT=%INPUT:P100=C%

ffmpeg -i %INPUT% -vf lut3d="VLog_to_V709.cube" -map_metadata 0 -pix_fmt yuv422p10le -c:v dnxhd -profile:v 4 -c:a pcm_s24le -color_range pc -movflags write_colr -y %OUTPUT%
exit /b

:the_end
pause
Mouse buttons and keyboard shortcuts:

<table>
<thead>
<tr>
<th>Mouse Button</th>
<th>Keyboard</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; • &gt;</td>
<td>(seems to be the same as moving the position with the mouse)</td>
<td></td>
</tr>
<tr>
<td>▪</td>
<td>J</td>
<td>Play backward</td>
</tr>
<tr>
<td></td>
<td>J+K</td>
<td>Play backward with half speed</td>
</tr>
<tr>
<td></td>
<td>Hold K and press J once, or press arrow left</td>
<td>One frame backward</td>
</tr>
<tr>
<td>■</td>
<td>K</td>
<td>Stop</td>
</tr>
<tr>
<td></td>
<td>Hold K and press L once, or press arrow right</td>
<td>One frame forward</td>
</tr>
<tr>
<td>K+L</td>
<td>Play forward with half speed</td>
<td></td>
</tr>
<tr>
<td>▶</td>
<td>L</td>
<td>Play forward</td>
</tr>
<tr>
<td></td>
<td>press L twice</td>
<td>Play with double speed</td>
</tr>
<tr>
<td>▪</td>
<td>Jump to the end</td>
<td></td>
</tr>
<tr>
<td>);}</td>
<td>Space</td>
<td>Play / Pause</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>Set the &quot;In&quot; marker</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>Set the &quot;Out&quot; marker</td>
</tr>
<tr>
<td>Q</td>
<td>Toggle between source and timeline viewer</td>
<td></td>
</tr>
<tr>
<td>F9</td>
<td>Insert</td>
<td></td>
</tr>
<tr>
<td>F10</td>
<td>Overwrite</td>
<td></td>
</tr>
<tr>
<td>F11</td>
<td>Replace</td>
<td></td>
</tr>
<tr>
<td>F12</td>
<td>Place on top</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>Trim edit mode</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>Dynamic trim mode (slip)</td>
<td></td>
</tr>
</tbody>
</table>

Note for left column: The symbols were taken from the "Segoe UI Symbol" character set.
12.2 Recording a Voice-Over audio track

This is also explained in german in the chapter "Audio in einer Timeline aufzeichnen" in: Paul Saccone, Dion Scoppettuolo: Der Ultimative Leitfaden zu DAVINCI RESOLVE 15

- Let's assume you are using the built-in microphone in your laptop, or the audio-in connector which is connected to the same A/D converter.
- Open a project, click on File --> "Project Settings", then click on "Capture and Playback", then select in the field "Save Clips to" the folder where you want to save the new audio tracks. Close the project settings with "Save".
- Hide the level meter and show the mixer (if it's not already visible). The mixer icon is at top left.
- Make a right click on any track headline and choose "Add Track" --> "Mono". Or "Stereo", if required. But normally a voice over is recorded in mono.
- Double click the headline of the new audio track and change the name to "VO" for voice over.
- You can also change the track color if you want.
- The following things are only possible if you switch to the "Fairlight" room, that's the ♫ icon.
- The new track is also shown in the mixer, and in the row "Input" it's marked as "No Input". Click on this field, select "Input..." and then a "Patch Input/Output" window will appear. At the left side you can select the microphone and at the right side you select the VO track. Click on "Patch" and close this window.
- All audio tracks have a "R" icon which can be activated for recording. Do this for the "VO" track. Most probably you will now hear an acoustic feedback loop. This is because the sound from the speakers is coupled back to the microphone. To avoid this acoustic feedback, either set the level for the "Main1" output to the lowest possible level (all way down), or simply activate the "Mute" icon for the "Main1" output (this is the "M" icon).
- Set the "VO" track to "Solo" by activating the "S" icon.
- Now you can start a record by clicking on the ● icon in the timeline (which is only available in the "Fairlight" room).
12.3 DaVinci Resolve, Problems and Solutions

DaVinci Resolve is designed for Windows 10. There are many problems when you run it on Windows 7. Don't do that!

One problem is that recording sound from USB soundcards is impossible or difficult. Although the USB soundcard is shown in the "Patch Input/Output" window and a patch can be created, there comes no signal from this soundcard.

I got it working with this workaround: Go to Davinci Resolve --> Preferences --> Video and Audio I/O --> Speaker Setup and make these settings:

- Speaker Configuration: Manual
- Monitor Set: MAIN
- Device: Lautsprecher (Realtek High Definition Audio)
- Monitor Set Format: Stereo

With these settings, I got the USB soundcard working. But it seems my Windows 7 computer is too slow and can't record a voice-over track in real time without disturbing artefacts.

Error message "Render Job 1 failed as the current clip could not be processed."

It's also possible to match an audio track to another audio track that was recorded with the camera, see page 339.
Manually syncing is described on page 340.

Set the timeline resolution in  File → Project Settings

Insert a clip in the timeline between two other clips without overwriting them: Drag and then press ctrl + shift while dropping in the timeline.
OBS - Open Broadcaster Software

The software is available here and the installation is quite simple: [https://obsproject.com/](https://obsproject.com/)

Wiki: [https://obsproject.com/wiki/](https://obsproject.com/wiki/)

After installation, OBS first asks some questions that I don’t remember exactly. But they are important because they are used to make many basic settings. So you should answer the questions as good as possible.

In OBS you can see the black window at the top, this is the area that will be streamed. Set the size of this area to the maximum value that your streaming target accepts, for example 1280x720 for Facebook Live.

Important is the "Sources" window. Here you can specify one or more sources to be displayed in the black window. Click on the "+" icon at the bottom of the "Sources" window. A long list of possibilities appears. The most important are these four:

- Image, for example a background image with a logo.
- Display Capture, that means the whole display area is copied
- Window Capture, that means the window from one application (for example SharpCap) is copied
- Video Capture Device, for example a webcam

The most important thing is "Window Capture", so I choose that one. You can now specify which window should be taken (SharpCap). Of course the SharpCap window must already exist, so SharpCap must have been started before. You can now see a copy of the SharpCap window at the top of the black window. You can drag the size and position with the mouse as you like. Only the black area will be visible later. If you move the SharpCap window out to the side, diagonally striped areas appear which would not be visible in the stream.

In the "Sources" window there is an "Eye" icon to the right of "Window Capture", so you can switch the visibility on and off.

If you don't want to copy the whole SharpCap window but only a part of it, it works like this: In the "Sources" window, right-click on "Window Capture" and then select "Filters". Click on the "+" icon at the bottom left and then select the "Crop/Pad" filter. Now you can set the four values left, top, right and bottom as you like. Then click on "Close".

My "Crop" values for SharpCap: 1, 50, 640, 248

My video resolution in SharpCap: 2560 x 1440, binning=2 (which gives 1280x720)

You can also create multiple sources in the "Sources" window. The order determines what is hidden by whom. You can change the order by right clicking --> Order. So you could e.g. add a "Video Capture Device" source and display it in the lower right corner of the black window, so that you can see yourself while you are talking. If the webcam video is to be displayed in the foreground (i.e. in front of the SharpCap window), then it must be in first place in the "Sources" window. If it is not, you have to change the order.
In OBS there is also the "Audio Mixer" window. Here you can see the levels of all audio sources, and below that there are sliders to adjust the sensitivity. The level should rarely go into the red area, and never to the right stop.

OBS also offers the option of not streaming, but simply recording a video, by clicking "Start recording" in the lower right corner. This is useful for testing. You can watch the just recorded video with File --&gt; Show Recordings (R). The videos are saved in the folder Username / My videos.

13.1 OBS - Problems and solutions

- How can you test in advance if everything works properly without everyone seeing it live on Facebook? There are two possibilities: a) In OBS you don't click on "Start Streaming" but on "Start Recording". Then only a video will be recorded, which you can watch afterwards. b) You can broadcast the stream live to Facebook, but set the target group in Facebook to "Only me" so that no one else can see it. But this is only possible on your own Facebook page.

- Problem: The viewer can't see the mouse pointer. Solution: Two conditions must be met for the mouse pointer to be visible to the viewer. 1. in the properties of the Window Capture (right click on it, Properties) the checkbox "Record mouse pointer" must be checked. 2. The mouse pointer is only visible to the viewer if it is over the source window and if this window is active (i.e. it must be clicked). In Windows only one window is active at a time. If you move the mouse pointer over the OBS window and try to show something there, the viewers can't see the mouse pointer!

- What's with the "Scenes"? One scene is basically always present. You can add more scenes (click on the "+" icon below, then give it a meaningful name). Now you can see what it is all about. You can click on the scenes and then the black window changes. The second scene has no sources yet, so the whole window is black. Each scene can have its own sources (which can be the same or different), and you can also arrange the windows in each scene differently. During the presentation, you can switch back and forth between the scenes by clicking on the corresponding scene in the "Scenes" window. Strangely enough, this doesn't work if you click on the ^ v icons below (I don't know what this is supposed to do).

- How do you switch back and forth between different scenes quickly and easily? You can do that by clicking on the appropriate scenes in OBS. You can also define hotkeys in OBS. You have to be careful that you only use hotkeys that have no function in the other programs used (here: SharpCap). I have now created three scenes for example. In the hotkey list there is an entry "Scene name / Switch to scene" for each scene. There I have entered 1, 2, or 3. Now I can select the corresponding scenes by pressing the buttons 1, 2 or 3. Interestingly, OBS even detects the keypress when it is not active and in the background.

- The image from the webcam is reversed. How do you correct this? Solution: Right click on "Video Capture Device", then "Transform" and "Flip Horizontal".

- The image (or Window capture or Video capture Device) is displayed too small or in the wrong place in the black window. Solution: Simply move or drag the size with the mouse.

- The image (or Window capture or Video capture Device) is displayed too large in the black window so that only a partial area is visible. You
cannot make it smaller with the mouse, because the right border isn't visible at all. How can you adjust it to the visible area? Solution: Right click on the corresponding entry in the "Sources" window, then Transform --> Fit to screen

- OBS also works (in contrast to the simple operating mode with Facebook) with cameras that are not listed under "Image processing devices" in the control panel but instead under "USB controllers", for example the uEye UI-1240ML camera from IDS and the STC-TB33USB camera from Sentech. You only have to install the drivers for the cameras (from the manufacturer's website, after registration) and can then use them directly as "Video Capture Device", i.e. you don't need to run any third-party software (like SharpCap). With these cameras you can set the exposure manually. If someone is interested: I have some of both cameras in stock. The IDS camera has 1280x1024 pixels at 25.8 fps and the Sentech STC-TB33USB has 640x480 pixels at 60 fps, both cameras are monochrome. IDS has a metal housing with a C-mount connector, while Sentech STC-TB33USB is just an electronic module without a housing (i.e. you have to build a housing around it yourself).

- How do you record a presentation to review and then upload it to Facebook? Solution: In OBS you don't click on "Start Streaming" but on "Start Recording". Then a video is recorded which you can watch afterwards. You can also stream and record simultaneously with OBS.

- How can you insert a mouse pointer into a video in postprocessing to explain something? Example: https://www.youtube.com/watch?v=5QRx4GRNxED Solution: Two sources are added to OBS: 1. a Window Capture pointing to the VLC player. 2. An Audio Output Capture where you select the built-in speakers as the device. It is useful to work with two screens. If one screen is used, the VLC Player runs in full screen mode without any menus. In this window you can explain things with the mouse pointer. On the second screen OBS is running. First you start the video in the VLC Player. Then rewind to the beginning and stop it immediately with the space key. Then you start the recording in OBS. Then you start the video in the VLC Player.

- How to apply a LUT to the video capture device? Right click on the video capture device in the "sources" window, choose "Filters", then click on the "+" icon below the "Effect filters" window and select "Apply LUT". Then enter the path to the LUT file which can be a *.cube file. Don't forget to activate the filter by clicking on the icon left of "Apply LUT".

13.2 CPU Load

I've been experiencing audio blackouts. This was not a bandwidth problem when transferring to Facebook, because the same problem occurred when I only recorded the video in OBS.

It turned out that the CPU load was too high. The problem could be mitigated by limiting the framerate to 4 or 8 fps in SharpCap and setting 10 fps in OBS (Settings -> Video). With these settings I see in the Task Manager about 25% CPU load for SharpCap, 20-25% for OBS and about 20% for a total of 6 Firefox processes. That's only about 70%, but even with that there are still short dropouts in the sound. So note: It is recommended to have the Task Manager open on a second screen and keep an eye on the CPU load. The peak load is higher than what you see in the y,t diagram. Rough thumb value: The average value should not exceed 50%.

You can create a shortcut to the Task Manager on the (Windows 7) desktop as follows:

Right click on a free space on the desktop, then New --> Shortcut and then enter "%SYSTEMROOT%\system32\taskmgr.exe".

You can create a shortcut to the Device Manager on the (Windows 7) desktop as follows:
13.3 Facebook Live

This chapter is about live presenting of astronomical contents in Facebook Live. Because my Facebook account is set to german language, I'm not sure how some things are called in the english version. I add the german terms in [brackets].

On your own Facebook page there is a menu item "Live-Video" to the right of "Create Post" [Erstellen]. If you click on it, you can choose between "Camera" [Kamera] and "Connect" [Verbinden]. These are two fundamentally different modes of operation.

First of all: I also get a message "Try Live Producer" [Live Producer ausprobieren]. This didn't work properly for me under Windows 7 and can be clicked away. In the following chapters I describe the two operating modes "Camera" and "Connect".

13.4 Facebook Live - Mode "Camera"

"Camera" [Kamera] is the simpler of the two operating modes. Either the camera built into the notebook is used, or an external webcam connected via USB. Picture and sound is transferred from the browser (or probably by a hidden Javascript) to Facebook. No additional software is required.

In the right window some settings have to be made:

- "Select where you want to post your live broadcast:" [Wähle aus, wo du deine Live-Übertragung posten möchtest:] is set to "Share in your timeline" [In deiner Chronik teilen], or "Share on a page you manage" [Auf einer Seite, die du verwaltet, teilen].
- At "Say something about this live video" [Sag etwas über dieses Live-Video] you can write something meaningful into it. The audience can see this text.
- Below this you set who can see the live video. For the first test it makes sense to set "Only me" [Nur ich]. Later, when you have gained enough experience and are sure that everything works, you can set "Friends" [Freunde] or "Public" [Öffentlich].
- Below that you can select which camera and microphone is used. If only one camera or only one microphone is available, there are of course not many options to choose from.
- At "Title" [Titel] you set a meaningful headline. The audience can see this headline.
- At the very bottom, the "Go Live" [Live gehen] button starts the live broadcast (after a countdown 3-2-1) and at the same place a red "Stop Live Video" [Live-Video beenden] button appears, which allows you to stop the broadcast at any time.

This mode is only useful for shots that have a normal contrast range, e.g. when you sit in front of the camera and talk about something. Simple webcams have an automatic gain control which unfortunately cannot be deactivated. For objects with a high contrast range, e.g. moon or Venus through the telescope, the automatic gain control fails. The bright parts of the image are totally overexposed.

Not all webcams are suitable. Facebook seems to have problems with such cameras that are not listed in the Windows Device Manager under "Image Processing Devices" [Bildverarbeitungsgeräte] but under "USB Controller". This applies, for example, to the uEye UI-124xML camera from IDS and the STC-TB33USB from Sentech. This is unfortunate, because with these cameras you can manually adjust the exposure.
13.5 Facebook Live - Mode "Connect"

"Connect" [Verbinden] is the more complicated operating mode, but it offers much more possibilities. The transmission of audio and video to Facebook isn't done by the browser, but by a special streaming software that you have to install first. There are many streaming applications available. I use OBS (Open Broadcaster Software).

After clicking on the menu item "Live Video" on your Facebook page to the right of "Create Post" [Beitrag erstellen] and then on "Connect" [Verbinden] at the top, you will see a page with the title "Connect your Live Stream to the Live API" [Verbinde deinen Live Stream mit der Live API]. You have to make some settings on the right:

- "Select where you want to post your live broadcast:" [Wähle aus, wo du deine Live-Übertragung posten möchtest:] is set to "Share in your timeline" [In deiner Chronik teilen], or "Share on a page you manage" [Auf einer Seite, die du verwaltet, teilen] and below that, the correct page is selected.
- At "Say something about this live video" [Sag etwas über dieses Live-Vdeo] you can write something meaningful into it. The audience can see this text.
- Below this you set who can see the live video. For the first test it makes sense to set "Only me" [Nur ich]. Later, when you have gained enough experience and you are sure that everything works, then you set "Friends" [Freunde] or "Public" [öffentlich]. On some Facebook pages it's not possible to select "Only me" [Nur ich], i.e. everything that is streamed there is visible to everyone. For testing you therefore better use your own Facebook page first.
- At "Title" you set a meaningful headline. The audience can see the headline.

The "Stream Key" [Stream-Schlüssel] is displayed on the left side of the page. If you click on "copy" [kopieren] to the right of it, this key is copied to the clipboard.

This key must now be entered into OBS under Settings (this is in the lower right corner) --> Stream --> Stream Key [--> Stream --> Streamschlüssel]. If there is already a key in there, it can be overwritten. Then click on "Start Streaming" [Stream starten] and switch back to Facebook in the browser window. This will take a few seconds, and then the streamed video sent from OBS will appear in the browser. What you see here is a few seconds delayed. You can move the scrollbar down to see the whole video. But so far this is only a preview, which is not yet broadcasted live.

Before you start the live broadcast, you select the entry scene in OBS, for example your background logo.

At the bottom right of the Facebook page is the "Go Live" [Live gehen] button. This starts the live broadcast and a red "Stop Live Video" [Live-Video beenden] button appears at the same place, so you can stop the broadcast at any time.

You now switch to the OBS window and start the live broadcast. You can switch back and forth between different scenes. I have used the following scenes so far:

- Background image with observatory logo
- SharpCap window, cropped to hide the menus
• SharpCap window, additionally a small video overlay in the lower right corner where I'm seen while I'm talking (comes from the webcam built into my notebook)
• A window with a star chart (Guide 9.1)
• other images that I've taken in advance

During the live presentation you can operate the camera, move the telescope, change exposure time or gain, and you can also use the cursor to explain things.

Important: If you want to explain something with the mouse pointer, the mouse pointer must not be over the OBS window, but over the source window (e.g. SharpCap or Guide 9.1). If you try to explain something with the mouse pointer in the OBS window, the audience cannot see the mouse pointer! Therefore it's advisable to move the OBS window to a second screen.

### 13.6 YouTube Live

- Log into your YouTube account
- Click the "Create Video" button
- Click on "Go Live" [Livestream starten]
- Select "New Stream" and fill in the requested fields
- Click on "Create Stream"
- Click on "Copy" [Kopieren] right of "Stream Key (paste in encoder)" [Streamschlüssel (in Encoder einfügen)]
- Paste this key in OBS and start streaming
- Click the "Go Live" [LIVESTREAM STARTEN] button.

Youtube Live is complicated. I don't like the user interface. It's very complicated to find the menu where you can delete your own test videos. Facebook Live has better user interface.

See also: [http://johnriselvato.com/ffmpeg-how-to-obtain-a-youtube-streaming-key/](http://johnriselvato.com/ffmpeg-how-to-obtain-a-youtube-streaming-key/)
See also: [http://johnriselvato.com/ffmpeg-how-to-stream-a-file-to-youtube/](http://johnriselvato.com/ffmpeg-how-to-stream-a-file-to-youtube/)

### 13.7 Desktop for live broadcasting

- It is advisable to connect a second screen to the notebook. Then you can leave the windows that should be visible to viewers on the notebook screen, while on the second screen you have the OBS window and additionally the task manager to control the CPU load. The second screen is
set up via Control Panel --> Hardware and Sound --> Connect to a projector --> Advanced or alternatively via a function key on the notebook (this depends on the notebook)
• Secure all (USB) cables with clamps to prevent loose contacts during broadcasting
• The neighbours are a nuisance, as they start lawnmowers or garden shredders in the middle of the live broadcast. I have no solution :-(
• Viewers can write comments and ask questions during the live presentation. But those who do the presentation don't see them, because they are fully occupied with narration and have to switch back and forth between the SharpCap and OBS windows. It is absolutely impossible to have the browser window with Facebook in view at the same time. The only solution is to divide the work: a second person must have a look at the Facebook page at the same time in order to answer questions promptly.
• Problem: The WLAN does not reach far enough to where the transmission is to be made, and there is no LAN cable of the required length available. Solution: Use RG-59 coaxial cable (75 Ω) or the white 75 Ω antenna cable. At each end the shielding is removed over a length of 4 cm. So the inner conductor protrudes 4 cm at the end of the cable. The insulation of the inner conductor does not need to be removed. This is a 1/4 wave antenna. One end is placed close to the WLAN device (a few centimeters next to the transmitting antenna) and the other end is placed near the notebook.

13.8 Live broadcasting, or recording in advance?

What's better, a live broadcast or a recording that is sent with a time delay?

Recording in advance has several advantages:
• You can repeat the recording if something went wrong
• You can upload the recorded video both to Facebook and to the YouTube channel
• Sometimes Facebook Live doesn't seem to work at all, an error message appears when you click on "Go Live" (although the stream is already running and the preview is visible in Facebook), with a note that you should try again. But after the 10th try I gave up and instead made a recording.
• OBS can also be used to stream and record a video simultaneously. Just click on "Start Stream" and "Start Recording" one after the other. But you have to keep an eye on the CPU load.

Whether live broadcast or recording, OBS is the appropriate tool to create a broadcast that uses different sources.
Tips and tricks for video

- Learn by analyzing other videos and films
- Use a variable neutral density filter, so that you can use wide open aperture for narrow depth of field
- Always record 3 seconds before the action begins and also 3 seconds after action has ended.
- Use a good tripod for video. A recommended manufacturer is Sachtler.
- Interviewing two people: Record one from the left and the other from the right.
- Know your camera before you begin to record videos.

Cinema advertizing: DCP format (Digital Cinema Package), see also https://dcpomatic.com/

Checklist:
- Is the focus correct?
- Is the exposure correct? Check with a 18% graycard.
- Is the sound recorder set to the correct level?
- Camera running
- Sound recorder running
- Synchronization signal
- Wait 3 seconds
- Action
- Wait 3 seconds
- Stop camera and sound recorder
15 LINUX and GIT

Some experiments with Linux...

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$ apt-get install ffmpeg</td>
<td>Install FFmpeg</td>
</tr>
<tr>
<td>$ git clone <a href="http://git.ffmpeg.org/ffmpeg.git">http://git.ffmpeg.org/ffmpeg.git</a></td>
<td>Download the FFmpeg source code. Unfortunately the source code is very badly commented (at least those files that I had a look at).</td>
</tr>
<tr>
<td>$ ./configure $ make $ sudo make install</td>
<td>This is the procedure for building FFmpeg. Go to the FFmpeg folder and type these three command lines. For detailed instructions see <a href="https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu">https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu</a></td>
</tr>
<tr>
<td>$ sudo apt install vlc</td>
<td>Install VLC player (or any other program)</td>
</tr>
<tr>
<td>$ chmod +x my_script</td>
<td>Makes a script file executable. In Linux script files have no extension. The first line in the script file must contain: #!/bin/bash</td>
</tr>
<tr>
<td>$ cd ..</td>
<td>Go back to the parent folder. Please note there must be a space character between cd and the two dots.</td>
</tr>
</tbody>
</table>

How to change something in the FFmpeg documentation and how to use git, this example is from Carl Eugen Hoyos 27.9.19 in the FFmpeg user mailing list. The example works but I don’t yet understand how git works. It’s very complicated and seems quite time-consuming to learn.

```
$ git clone http://git.ffmpeg.org/ffmpeg.git
$ cd ffmpeg
edit a file in the doc directory.
$ git commit doc
(I suspect this will ask you to set your name and email when running it for the first time)
$ git format-patch HEAD^ This produces a file that you can send to the mailing list after visual inspection for commit message and your name.
$ git reset HEAD^ 
```

Here is a page with instructions for git http://www.ffmpeg.org/git-howto.html but unfortunately it's not written in a form a beginner can easily understand.
A tutorial for git (in german language) is here [https://open.hpi.de/courses/git2020](https://open.hpi.de/courses/git2020) and the required time is estimated as 2-5 hours per week over 4 weeks.

Summary:

<table>
<thead>
<tr>
<th>Task</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Making changes in the C source code</td>
<td>very easy</td>
</tr>
<tr>
<td>Compiling FFmpeg under Linux without any options</td>
<td>easy, but the resulting executable misses important features</td>
</tr>
<tr>
<td>Compiling FFmpeg under Linux with additional options</td>
<td>I haven't yet figured out how to do this</td>
</tr>
<tr>
<td>Compiling FFmpeg under Windows with Visual Studio</td>
<td>After reading the instructions it seems very complicated, I didn't even try</td>
</tr>
<tr>
<td>Using git</td>
<td>difficult and very time consuming to learn</td>
</tr>
</tbody>
</table>
## Cameras and lenses for fulldome video production

<table>
<thead>
<tr>
<th></th>
<th>Canon 6D</th>
<th>Panasonic LUMIX GH5S</th>
<th>PanoView XDV360</th>
<th>Kodak SP360_4K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="Canon 6D" /></td>
<td><img src="image" alt="Panasonic LUMIX GH5S" /></td>
<td><img src="image" alt="PanoView XDV360" /></td>
<td><img src="image" alt="Kodak SP360_4K" /></td>
</tr>
<tr>
<td>Fulldome resolution</td>
<td>180°: 3648 x 3648 (Pictures)</td>
<td>180°: 2880 x 2880 (Pictures)</td>
<td>220°: 2448 x 2448</td>
<td>235°: 2880 x 2880</td>
</tr>
<tr>
<td></td>
<td>180°: 1080 x 1080 (Video)</td>
<td>180°: 2496 x 2496 (Video)</td>
<td>180°: 2104 x 2104</td>
<td>180°: 2456 x 2456</td>
</tr>
<tr>
<td>Sound recording</td>
<td>stereo 48000 Hz, but both channels are identical, if no external microphone is connected</td>
<td>stereo 48000 Hz, but both channels are identical, if no external microphone is connected</td>
<td>mono 8000 Hz, there is no connector for an external microphone</td>
<td>stereo 48000 Hz, but both channels are almost equal because the microphones are close together; no connector for external microphones</td>
</tr>
<tr>
<td>Suitable for fulldome video?</td>
<td>yes, if a fisheye lens is used which has a 180° image diameter less than 20.2mm</td>
<td>yes, if a fisheye lens is used which has a 180° image diameter less than 13.0mm</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Suitable for fulldome video at night?</td>
<td>yes</td>
<td>yes, very good</td>
<td>no, too much noise</td>
<td>no, too much noise</td>
</tr>
<tr>
<td>Suitable for fulldome timelapse?</td>
<td>yes, arbitrary interval times with external timer</td>
<td>yes, arbitrary interval times with external timer</td>
<td>yes, with internal timer</td>
<td>yes, with internal timer</td>
</tr>
</tbody>
</table>
16.1 Read-out chip size of cameras at different video modes

Problem: A full format chip has the size 36mm x 24mm and thus the format 3:2. For video recording, however, the format 16:9 is used, so that only a part with the dimensions 36mm x 20.25mm is read out. But as a full format fisheye normally illuminates a 24mm diameter circle, there are two strips missing at the top and bottom of the video.

If the entire image circle of the fisheye lens is to be recorded in the video, the image circle diameter of the lens must not be larger than the read-out height of the chip at the set video resolution.

<table>
<thead>
<tr>
<th>Camera</th>
<th>Chip Size</th>
<th>Pixels</th>
<th>Video Resolution</th>
<th>Read-out Part of the Chip, Width x Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon 6D</td>
<td>35.8mm x 23.9mm</td>
<td>5472 x 3648</td>
<td>640 x 480 (4:3)</td>
<td>31.87mm x 23.9mm</td>
</tr>
<tr>
<td>Canon 6D</td>
<td>35.8mm x 23.9mm</td>
<td>5472 x 3648</td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>35.9mm x 20.19mm</td>
</tr>
<tr>
<td>Canon 5D MK4</td>
<td>36mm x 24mm</td>
<td>6720 x 4480</td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>36mm x 20.25mm</td>
</tr>
<tr>
<td>Canon 5D MK4</td>
<td>36mm x 24mm</td>
<td>6720 x 4480</td>
<td>4096 x 2160 C4K (17:9)</td>
<td>21.94mm x 11.57mm (Not the whole chip width is used)</td>
</tr>
<tr>
<td>Canon 7D</td>
<td>22.3mm x 14.9mm</td>
<td>5184 x 3456</td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>22.30mm x 12.54mm</td>
</tr>
<tr>
<td>Canon EOS R</td>
<td>36mm x 24mm</td>
<td>6720 x 4480</td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>36mm x 20.25mm</td>
</tr>
<tr>
<td>Canon EOS R</td>
<td>36mm x 24mm</td>
<td>6720 x 4480</td>
<td>3846 x 2160 4K (16:9)</td>
<td>20.57mm x 11.57mm (Not the whole chip width is used)</td>
</tr>
<tr>
<td>Sony A7S II</td>
<td>35.6mm x 23.8mm</td>
<td>4240 x 2832</td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>35.6mm x 20.0mm</td>
</tr>
<tr>
<td>Sony A7S II</td>
<td>35.6mm x 23.8mm</td>
<td>4240 x 2832</td>
<td>3840 x 2160 4K (16:9)</td>
<td>35.6mm x 20.0mm (The whole chip width is used)</td>
</tr>
<tr>
<td>Panasonic LUMIX DC-GH5S</td>
<td>19.2mm x 13.0mm</td>
<td>4096 x 2760</td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>18.8mm x 10.6mm (yet to be confirmed)</td>
</tr>
<tr>
<td>Panasonic LUMIX DC-GH5S</td>
<td>19.2mm x 13.0mm</td>
<td>4096 x 2760</td>
<td>3846 x 2160 4K (16:9)</td>
<td>18.8mm x 10.6mm (yet to be confirmed)</td>
</tr>
<tr>
<td>Panasonic LUMIX DC-GH5S</td>
<td>19.2mm x 13.0mm</td>
<td>4096 x 2760</td>
<td>4096 x 2160 C4K (17:9)</td>
<td>19.2mm x 10.12mm (The whole chip width is used)</td>
</tr>
<tr>
<td>Panasonic LUMIX DC-GH5S</td>
<td>19.2mm x 13.0mm</td>
<td>4096 x 2760</td>
<td>3328 x 2496 Anamorphic (4:3)</td>
<td>17.3mm x 13.0mm (The whole chip height is used)</td>
</tr>
<tr>
<td>Camera Model</td>
<td>Chip Size</td>
<td>Sensor Size</td>
<td>Output Size</td>
<td>Chip Size Information</td>
</tr>
<tr>
<td>------------------</td>
<td>--------------------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Nikon D800</td>
<td>35.9mm x 24.0mm</td>
<td>7360 x 4912</td>
<td>1920 x 1080 Full HD</td>
<td>32.0mm x 18.0mm (Not the whole chip width is used)</td>
</tr>
<tr>
<td>ZWO ASI178MM</td>
<td>7.4mm x 5.0mm</td>
<td>3096x2080</td>
<td>3096x2080</td>
<td>7.4mm x 5.0mm (The full chip size is used)</td>
</tr>
<tr>
<td>Pulnix TM-9701</td>
<td>8.9mm x 6.6mm</td>
<td>768 x 484</td>
<td>768 x 484</td>
<td>8.9mm x 6.6mm (The full chip size is used)</td>
</tr>
</tbody>
</table>

Effective chip size of GH5S with 0.64x SpeedBooster, in FHD or 4K mode: 29.37mm x 16.56mm
Effective chip size of GH5S with 0.64x SpeedBooster, in Anamorphic 4:3 mode: 27.03mm x 20.31mm
## 16.2 Overview of available fisheye lenses

<table>
<thead>
<tr>
<th>Lens</th>
<th>Mount</th>
<th>Aperture</th>
<th>Image Angle and Image Circle Diameter</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon EF 8-15mm at 8mm</td>
<td>Canon EF</td>
<td>f/4.0</td>
<td>180° 22.9mm (measured myself)</td>
<td>Very good image quality</td>
</tr>
<tr>
<td>Sigma EX DG 8mm</td>
<td>Canon EF ...</td>
<td>f/3.5</td>
<td>180° 22.7mm (measured myself)</td>
<td>Mediocre image quality</td>
</tr>
<tr>
<td>Nippon Kogaku 8mm</td>
<td>M42 / Canon EF</td>
<td>f/2.8</td>
<td>180° 23.0mm (measured myself)</td>
<td>M42 mount with adapter to Canon EF</td>
</tr>
<tr>
<td>Sigma EX DG 4.5mm</td>
<td>Canon EF ...</td>
<td>f/2.8</td>
<td>180° 12.3mm (measured myself)</td>
<td>Mediocre image quality</td>
</tr>
<tr>
<td>Meike 6-11mm at 6mm</td>
<td>Canon EF ...</td>
<td>f/3.5</td>
<td>180° 15.1mm (measured myself)</td>
<td>Good image quality</td>
</tr>
<tr>
<td>Meike 6-11mm at 7.5mm</td>
<td>Canon EF ...</td>
<td>f/3.5</td>
<td>180° 18.4mm (measured myself)</td>
<td>Good image quality</td>
</tr>
<tr>
<td>Meike 6-11mm at 9.5mm</td>
<td>Canon EF ...</td>
<td>f/3.5</td>
<td>180° 23.7mm (measured myself)</td>
<td>Good image quality</td>
</tr>
<tr>
<td>Meike 6-11mm at 11mm</td>
<td>Canon EF ...</td>
<td>f/3.5</td>
<td>180° 28.7mm (measured myself)</td>
<td>Good image quality</td>
</tr>
<tr>
<td>Meike 8mm</td>
<td>Canon EF ...</td>
<td>f/3.5</td>
<td>180° approx. 26.9mm 200° approx. 29.9mm</td>
<td></td>
</tr>
<tr>
<td>Opteka 6.5mm</td>
<td>Canon EF</td>
<td>f/3.5</td>
<td>180° approx. 30mm</td>
<td>Bad image quality, true focal length is about 9mm</td>
</tr>
<tr>
<td>Entaniya HAL250 6.0mm</td>
<td>Canon EF ...</td>
<td>f/5.6</td>
<td>180° 18.2mm 250° 23.7mm</td>
<td>Only suitable for mirrorless cameras, very expensive</td>
</tr>
<tr>
<td>Entaniya HAL250 4.3mm</td>
<td>Canon EF ...</td>
<td>f/4.0</td>
<td>180° 13.1mm 250° 17.0mm</td>
<td>Only suitable for mirrorless cameras, very expensive</td>
</tr>
<tr>
<td>Entaniya HAL250 3.6mm</td>
<td>Canon EF ...</td>
<td>f/2.8</td>
<td>180° 11.0mm 250° 14.25mm</td>
<td>Only suitable for mirrorless cameras, very expensive</td>
</tr>
<tr>
<td>Entaniya HAL250 3.0mm</td>
<td>Canon EF ...</td>
<td>f/2.8</td>
<td>180° 9.2mm 250° 11.9mm</td>
<td>Only suitable for mirrorless cameras, very expensive</td>
</tr>
<tr>
<td>Entaniya HAL200 6.0mm</td>
<td>Canon EF ...</td>
<td>f/4.0</td>
<td>180° 18.2mm 200° 19.9mm</td>
<td>Only suitable for mirrorless cameras, very expensive</td>
</tr>
<tr>
<td>Entaniya HAL200 5.0mm</td>
<td>Canon EF ...</td>
<td>f/5.6</td>
<td>180° 15.2mm 200° 16.6mm</td>
<td>Only suitable for mirrorless cameras, very expensive</td>
</tr>
<tr>
<td>Samyang 8mm Fisheye II</td>
<td>EF-M, Sony E</td>
<td>f/2.8</td>
<td>180° approx. 29.7mm 188° approx. 31mm</td>
<td>Only suitable for mirrorless cameras, short flange distance</td>
</tr>
<tr>
<td>Meike 6.5mm</td>
<td>MFT</td>
<td>f/2.0</td>
<td>180° 15.4mm 190° 15.85mm (measured myself)</td>
<td>Only suitable for mirrorless cameras, short flange distance</td>
</tr>
<tr>
<td>Meike 3.5mm</td>
<td>MFT</td>
<td>f/2.8</td>
<td>180° 11.0mm 220° 12.5mm</td>
<td>Only suitable for mirrorless cameras, short flange distance</td>
</tr>
<tr>
<td>Olympus M.Zuiko 8mm</td>
<td>MFT</td>
<td>f/1.8</td>
<td>180° approx. 22mm</td>
<td>Lens hood must be removed mechanically</td>
</tr>
<tr>
<td>7artisans (Viltrox) 7.5mm</td>
<td>MFT ...</td>
<td>f/2.8</td>
<td>about 27mm (APS-C without vignetting)</td>
<td>Lens hood must be removed mechanically</td>
</tr>
<tr>
<td>Lens Name</td>
<td>Mount Type</td>
<td>f/Value</td>
<td>180° Viewing Distance</td>
<td>200° Viewing Distance</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>---------</td>
<td>-----------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>ZLKC (OCDAY) 7.5mm</td>
<td>MFT</td>
<td>f/2.8</td>
<td>11.6mm</td>
<td>12.9mm</td>
</tr>
<tr>
<td>Laowa 4mm</td>
<td>MFT</td>
<td>f/2.8</td>
<td>11.7mm</td>
<td>13.0mm</td>
</tr>
<tr>
<td>iZugar MKX200-ASPH</td>
<td>MFT</td>
<td>f/2.8</td>
<td>11.7 mm</td>
<td>13.0mm</td>
</tr>
<tr>
<td>iZugar MKX22 3.25mm</td>
<td>MFT</td>
<td>f/2.5</td>
<td>8.2mm</td>
<td>10mm</td>
</tr>
<tr>
<td>Yumiki 2.5mm</td>
<td>CS-Mount</td>
<td>f/1.6</td>
<td>6.1mm</td>
<td>6.4mm</td>
</tr>
<tr>
<td>SMTSEC 2.27mm</td>
<td>CS-Mount</td>
<td>f/1.4</td>
<td>7.2mm</td>
<td></td>
</tr>
<tr>
<td>Fujinon 1.8mm</td>
<td>C-Mount</td>
<td>f/1.4</td>
<td>5.5mm</td>
<td>5.7mm</td>
</tr>
<tr>
<td>Fujinon 2.7mm</td>
<td>C-Mount</td>
<td>f/1.8</td>
<td>8.4mm</td>
<td>8.6mm</td>
</tr>
</tbody>
</table>

For measurements of fisheye lens nonlinearity, see also Paul Bourke's website: [http://paulbourke.net/dome/fisheye正确/](http://paulbourke.net/dome/fisheye正确/)

Note: The x axis of the diagrams is the viewing angle in radians and the y axis is the normalized radius in the image plane (1.0 at the circular edge).
## 16.3 Favorable camera / fisheye combinations

<table>
<thead>
<tr>
<th>Camera</th>
<th>Video resolution</th>
<th>Lens</th>
<th>Aperture</th>
<th>Fully illuminated image circle</th>
<th>Diameter of fully illuminated image circle in pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Canon 6D</strong></td>
<td>640 x 480 (4:3)</td>
<td>Canon EF 8-15mm at 8mm</td>
<td>f/4.0</td>
<td>180°</td>
<td>460 Pixel</td>
</tr>
<tr>
<td></td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>Sigma EX DG 4.5mm</td>
<td>f/2.8</td>
<td>180°</td>
<td>656 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meike 6-11mm at 8.2mm</td>
<td>f/3.5</td>
<td>180°</td>
<td>1080 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Canon EF 8-15mm at 8mm</td>
<td>f/4.0</td>
<td>159°</td>
<td>952 Pixel</td>
</tr>
<tr>
<td><strong>Canon 5D MK4</strong></td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>Sigma EX DG 4.5mm</td>
<td>f/2.8</td>
<td>180°</td>
<td>654 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meike 6-11mm at 8.2mm</td>
<td>f/3.5</td>
<td>180°</td>
<td>1080 Pixel</td>
</tr>
<tr>
<td></td>
<td>4096 x 2160 C4K (17:9)</td>
<td>Sigma EX DG 4.5mm</td>
<td>f/2.8</td>
<td>170°</td>
<td>2160 Pixel</td>
</tr>
<tr>
<td><strong>Canon EOS R</strong></td>
<td>3840 x 2160 4K (16:9)</td>
<td>Entaniya HAL250 3.6mm</td>
<td>f/2.8</td>
<td>180°</td>
<td>2054 Pixel</td>
</tr>
<tr>
<td><strong>Sony A7S II</strong></td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>Sigma EX DG 4.5mm</td>
<td>f/2.8</td>
<td>180°</td>
<td>663 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meike 6.5mm</td>
<td>f/2.0</td>
<td>180°</td>
<td>832 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meike 6-11mm at 8.2mm</td>
<td>f/3.5</td>
<td>180°</td>
<td>1080 Pixel</td>
</tr>
<tr>
<td></td>
<td>3840 x 2160 4K (16:9)</td>
<td>Sigma EX DG 4.5mm</td>
<td>f/2.8</td>
<td>180°</td>
<td>1325 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meike 6.5mm</td>
<td>f/2.0</td>
<td>180°</td>
<td>1663 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meike 6-11mm at 8.2mm</td>
<td>f/3.5</td>
<td>180°</td>
<td>2160 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Olympus M.Zuiko 8mm</td>
<td>f/1.8</td>
<td>approx. 164°</td>
<td>2160 Pixel</td>
</tr>
<tr>
<td><strong>Sony A7S II</strong></td>
<td>3840 x 2160 4K (16:9)</td>
<td>Sigma EX DG 8mm</td>
<td>f/3.5</td>
<td>180°</td>
<td>2060 Pixel</td>
</tr>
<tr>
<td>with external recorder</td>
<td></td>
<td>Olympus M.Zuiko 8mm</td>
<td>f/1.8</td>
<td>180°</td>
<td>ca. 1996 Pixel</td>
</tr>
<tr>
<td><strong>Panasonic LUMIX GH5S</strong></td>
<td>1920 x 1080 Full HD (16:9)</td>
<td>Sigma EX DG 4.5mm, Speedbooster 0.71x</td>
<td>f/2.0</td>
<td>180°</td>
<td>888 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sigma EX DG 4.5mm, Speedbooster 0.64x</td>
<td>f/1.8</td>
<td>180°</td>
<td>800 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Meike 6-11mm at 6.8mm, Speedbooster 0.64x</td>
<td>f/2.2</td>
<td>180°</td>
<td>1080 Pixel</td>
</tr>
<tr>
<td></td>
<td>3328 x 2496 Anamorphic (4:3)</td>
<td>Sigma EX DG 4.5mm</td>
<td>f/2.8</td>
<td>180°</td>
<td>2356 Pixel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nippon Kogaku 8mm, Speedbooster 0.64x</td>
<td>f/1.8</td>
<td>159°</td>
<td>2496 Pixel</td>
</tr>
<tr>
<td>Camera Model</td>
<td>3840 x 2160  4K (16:9)</td>
<td>1920 x 1080  Full HD (16:9)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
<td>----------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Meike 6.5mm</strong></td>
<td>f/2.0</td>
<td>152°</td>
<td>2496 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meike 6-11mm at 7.5mm, Speedbooster 0.71x</td>
<td>f/2.5</td>
<td>180°</td>
<td>2496 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Meike 6-11mm at 8.2mm, Speedbooster 0.64x</strong></td>
<td>f/2.2</td>
<td>180°</td>
<td>2496 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Olympus M.Zuiko 8mm</td>
<td>f/1.8</td>
<td>106°</td>
<td>2496 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meike 3.5mm</td>
<td>f/2.8</td>
<td>220°</td>
<td>2400 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sigma EX DG 4.5mm, Speedbooster 0.71x</strong></td>
<td>f/2.0</td>
<td>180°</td>
<td>1775 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sigma EX DG 4.5mm, Speedbooster 0.64x</td>
<td>f/1.8</td>
<td>180°</td>
<td>1600 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meike 6-11mm at 6.8mm, Speedbooster 0.64x</td>
<td>f/2.2</td>
<td>180°</td>
<td>2160 Pixel</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Nikon D800</strong></td>
<td>1920 x 1080  Full HD (16:9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Meike 6-11mm at 7.3mm</td>
<td>f/3.5</td>
<td>180°</td>
<td>1080 Pixel</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
16.4 Flange distances

<table>
<thead>
<tr>
<th>Mount</th>
<th>Flange Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFT (Micro 4/3)</td>
<td>19.25mm</td>
</tr>
<tr>
<td>Canon EF und EF-S</td>
<td>44.0mm</td>
</tr>
<tr>
<td>Canon EF-M</td>
<td>18.0mm</td>
</tr>
<tr>
<td>Canon R</td>
<td>20.0mm</td>
</tr>
<tr>
<td>Canon FD</td>
<td>42.0mm</td>
</tr>
<tr>
<td>M42 = M42x1.0</td>
<td>45.46mm</td>
</tr>
<tr>
<td>T 2 = M42x0.75</td>
<td>55.0mm</td>
</tr>
<tr>
<td>C-Mount</td>
<td>17.526mm</td>
</tr>
<tr>
<td>CS-Mount</td>
<td>12.526mm</td>
</tr>
<tr>
<td>Sony E-Mount</td>
<td>18.0mm</td>
</tr>
<tr>
<td>Nikon F</td>
<td>46.5mm</td>
</tr>
<tr>
<td>ZWO ASI178MM</td>
<td>12.5mm</td>
</tr>
</tbody>
</table>

16.5 Aperture numbers, rounded and exact

<table>
<thead>
<tr>
<th>Aperture Number</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
<th>1.1</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
<th>1.8</th>
<th>2.0</th>
<th>2.2</th>
<th>2.5</th>
<th>2.8</th>
<th>3.2</th>
<th>3.5</th>
<th>4.0</th>
<th>4.5</th>
<th>5.0</th>
<th>5.6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.794</td>
<td>0.891</td>
<td>1.000</td>
<td>1.122</td>
<td>1.260</td>
<td>1.414</td>
<td>1.587</td>
<td>1.782</td>
<td>2.000</td>
<td>2.245</td>
<td>2.520</td>
<td>2.828</td>
<td>3.175</td>
<td>3.564</td>
<td>4.000</td>
<td>4.490</td>
<td>5.040</td>
<td>5.657</td>
</tr>
</tbody>
</table>

Formula for exact numbers: \( f_{\text{no}} = 2^{n/6} \) with \( n = -2 \) to 15
### 16.6 Test patterns for fulldome projection

Very nice fulldome test patterns on Paul Bourke's website: [http://www.paulbourke.net/dome/testpattern/](http://www.paulbourke.net/dome/testpattern/)

Make a double-fisheye test image and an equirectangular test image:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set &quot;IN=1200.png&quot;</td>
<td>Test pattern from <a href="http://www.paulbourke.net/dome/testpattern/1200.png">http://www.paulbourke.net/dome/testpattern/1200.png</a></td>
</tr>
<tr>
<td>set &quot;OUT=double_fisheye_test.png&quot;</td>
<td>Double fisheye test image</td>
</tr>
<tr>
<td>ffmpeg -i %IN% -i %IN% -lavfi &quot;[0]transpose=1[left];[1]transpose=2,negate[right];[left][right]hstack&quot; -y %OUT%</td>
<td>Double fisheye test image</td>
</tr>
<tr>
<td>set &quot;IN=double_fisheye_test.png&quot;</td>
<td></td>
</tr>
<tr>
<td>set &quot;OUT=equirectangular_test.png&quot;</td>
<td>Equirectangular test image</td>
</tr>
<tr>
<td>ffmpeg -i %IN% -lavfi &quot;v360=input=dfisheye;output=e;ih_fov=180;iv_fov=180:pitch=90&quot; -y %OUT%</td>
<td>Equirectangular test image</td>
</tr>
<tr>
<td>pause</td>
<td></td>
</tr>
</tbody>
</table>
17 Canon 5D-Mark4

Resolution: 6.720 x 4.480, RAW 14-bit

17.1 All Canon 5D-Mark4 video modes for PAL video system

<table>
<thead>
<tr>
<th>MOV / MP4</th>
<th>Movie rec. size</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>4K 25.00P MJPG</td>
<td>4096x2160</td>
<td>25</td>
<td>480 Mbps</td>
<td>4:2:2 / 8 bit</td>
<td>MJPG yuvj422p</td>
</tr>
<tr>
<td></td>
<td>4K 24.00P MJPG</td>
<td>4096x2160</td>
<td>24</td>
<td>480 Mbps</td>
<td>4:2:2 / 8 bit</td>
<td>MJPG yuvj422p</td>
</tr>
<tr>
<td>FHD 50.00P ALL-I</td>
<td>1920x1080</td>
<td>50</td>
<td>174 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
<td></td>
</tr>
<tr>
<td>FHD 50.00P IPB</td>
<td>1920x1080</td>
<td>50</td>
<td>59 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB h264 yuvj420p</td>
<td></td>
</tr>
<tr>
<td>FHD 25.00P ALL-I</td>
<td>1920x1080</td>
<td>25</td>
<td>88 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
<td></td>
</tr>
<tr>
<td>FHD 25.00P IPB</td>
<td>1920x1080</td>
<td>25</td>
<td>30 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB h264 yuvj420p</td>
<td></td>
</tr>
<tr>
<td>FHD 24.00P ALL-I</td>
<td>1920x1080</td>
<td>24</td>
<td>88 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
<td></td>
</tr>
<tr>
<td>FHD 24.00P IPB</td>
<td>1920x1080</td>
<td>24</td>
<td>30 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB h264 yuvj420p</td>
<td></td>
</tr>
<tr>
<td>HD 100.0P ALL-I</td>
<td>1280x720</td>
<td>100</td>
<td>154 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
<td></td>
</tr>
<tr>
<td>MP4</td>
<td>FHD 50.00P IPB</td>
<td>1920x1080</td>
<td>50</td>
<td>58 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB h264 yuvj420p</td>
</tr>
<tr>
<td></td>
<td>FHD 25.00P IPB</td>
<td>1920x1080</td>
<td>25</td>
<td>29 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB h264 yuvj420p</td>
</tr>
<tr>
<td></td>
<td>FHD 25.00P IPB</td>
<td>1920x1080</td>
<td>25</td>
<td>12 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB (&quot;Light&quot;, this is a stronger compression) h264 yuvj420p</td>
</tr>
<tr>
<td></td>
<td>FHD 24.00P IPB</td>
<td>1920x1080</td>
<td>24</td>
<td>29 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB h264 yuvj420p</td>
</tr>
</tbody>
</table>
## 17.2 All Canon 5D-Mark4 video modes for NTSC video system

<table>
<thead>
<tr>
<th>MOV / MP4</th>
<th>Movie rec. size</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>4K 29.97P MJPG</td>
<td>4096x2160</td>
<td>29.97</td>
<td>480 Mbps</td>
<td>4:2:2 / 8 bit</td>
<td>MJPG  yuvj422p</td>
</tr>
<tr>
<td>MOV</td>
<td>4K 23.98P MJPG</td>
<td>4096x2160</td>
<td>23.98</td>
<td>480 Mbps</td>
<td>4:2:2 / 8 bit</td>
<td>MJPG  yuvj422p</td>
</tr>
<tr>
<td>MOV</td>
<td>4K 24.00P MJPG</td>
<td>4096x2160</td>
<td>24</td>
<td>480 Mbps</td>
<td>4:2:2 / 8 bit</td>
<td>MJPG  yuvj422p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 59.94P ALL-I</td>
<td>1920x1080</td>
<td>59.94</td>
<td>174 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 59.94P IPB</td>
<td>1920x1080</td>
<td>59.94</td>
<td>59 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 29.97P ALL-I</td>
<td>1920x1080</td>
<td>29.97</td>
<td>88 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 29.97P IPB</td>
<td>1920x1080</td>
<td>29.97</td>
<td>30 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 23.98P ALL-I</td>
<td>1920x1080</td>
<td>23.98</td>
<td>88 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 23.98P IPB</td>
<td>1920x1080</td>
<td>23.98</td>
<td>30 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 24.00P ALL-I</td>
<td>1920x1080</td>
<td>24</td>
<td>88 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>FHD 24.00P IPB</td>
<td>1920x1080</td>
<td>24</td>
<td>30 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
<tr>
<td>MOV</td>
<td>HD 119.9P ALL-I</td>
<td>1280x720</td>
<td>119.9</td>
<td>154 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>ALL-I h264 yuvj420p</td>
</tr>
<tr>
<td>MP4</td>
<td>FHD 59.94P IPB</td>
<td>1920x1080</td>
<td>59.94</td>
<td>58 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
<tr>
<td>MP4</td>
<td>FHD 29.97P IPB</td>
<td>1920x1080</td>
<td>29.97</td>
<td>29 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
<tr>
<td>MP4</td>
<td>FHD 29.97P IPB</td>
<td>1920x1080</td>
<td>29.97</td>
<td>12 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB (&quot;Light&quot;, this is a stronger compression) h264 yuvj420p</td>
</tr>
<tr>
<td>MP4</td>
<td>FHD 23.98P IPB</td>
<td>1920x1080</td>
<td>23.98</td>
<td>29 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
<tr>
<td>MP4</td>
<td>FHD 24.00P IPB</td>
<td>1920x1080</td>
<td>24.00</td>
<td>29 Mbps</td>
<td>4:2:0 / 8 bit</td>
<td>IPB   h264 yuvj420p</td>
</tr>
</tbody>
</table>

Important note: If the size of a video exceeds 4GB, it can only be downloaded to the computer with "EOS Utility" software.
17.3 Video tutorials for Canon 5D-Mark4

The Canon 5D-Mark4 has a very good autofocus and is perfect for photography of fast moving objects (e.g. wildlife, birds).
I'm not a friend of video tutorials, but for the Canon 5D-Mark4 I found some tutorials that are indeed helpful. I will summarize the content below:

Grant Atkinson: Canon 5D Mk IV - Autofocus: Part 1/4 - Control Setup for Moving Subjects
https://www.youtube.com/watch?v=7iP60Np0lpw

<table>
<thead>
<tr>
<th>AF Operation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ONE SHOT</td>
<td>For non-moving objects</td>
</tr>
<tr>
<td>AI FOCUS</td>
<td>This decides automatically if the object is moving or not. Not recommended.</td>
</tr>
<tr>
<td>AI SERVO</td>
<td>For moving objects, recommended as default.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drive Mode</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Single shot</td>
</tr>
<tr>
<td>H</td>
<td>7 Pictures per second</td>
</tr>
<tr>
<td>-</td>
<td>3 Pictures per second</td>
</tr>
<tr>
<td>S</td>
<td>3 Pictures per second, silent mode</td>
</tr>
<tr>
<td>Clock Symbol</td>
<td>10 Seconds self timer</td>
</tr>
<tr>
<td>Clock Symbol 2</td>
<td>2 Seconds self timer</td>
</tr>
</tbody>
</table>

Orange Menu (second from right) --> 3 --> Custom Controls

<table>
<thead>
<tr>
<th>Shutter Button</th>
<th>leave as-is</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF_ON Button</td>
<td>set to AF_OFF, that means when you are in AF_SERVO mode you can hold the focus as long as you press this button.</td>
</tr>
<tr>
<td>* Button</td>
<td>set to ONE_SHOT/SERVO, that means by pressing this button you can toggle very fast between ONE_SHOT and AF_SERVO. Additionally you must press the INFO button and select the option to the right. But this function isn't very important, because you can work without ONE_SHOT. In another video he sets the * button also to AF_OFF, which is useful if you accidentally press the wrong button.</td>
</tr>
<tr>
<td>Multi_Controller</td>
<td>Set to &quot;Direct AF point selection&quot;</td>
</tr>
<tr>
<td>AF Area Selection Button</td>
<td>Set to &quot;Direct AF area selection&quot;</td>
</tr>
</tbody>
</table>
In another video he sets the SET button to "Exposure Compensation"

Grant Atkinson: Canon 5D Mk IV - Autofocus: Part 2/4 - The 7 Focus Modes
https://www.youtube.com/watch?v=4lPrjb5w1Zw

Pink AF Menü (second from left) --> 4 --> Select AF area selec. mode
Here you can select which of the 7 AF area selection modes you want to use. He chooses 2, 3 and 5.

<table>
<thead>
<tr>
<th>(1)</th>
<th>Spot AF (Square with Point)</th>
<th>Very small, good choice if you take pictures through branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2)</td>
<td>Single Point AF</td>
<td>This is the default setting, very precise and fast, if you manage to hold the point on the object.</td>
</tr>
<tr>
<td>(3)</td>
<td>Expand AF area (5 Points)</td>
<td>Recommended method for moving objects. The center point is prioritized and if this point looses focus, then one of the neighbor points is used. Place the center point on the eye of the object.</td>
</tr>
<tr>
<td>(4)</td>
<td>AF Expand Surround (9 Points)</td>
<td>Same as (3), but 8 neighbor points.</td>
</tr>
<tr>
<td>(5)</td>
<td>Zone AF (9 or 12 Points)</td>
<td>All selected points have the same weight. You don’t know which point is actually used. Don't use this method if you want to have the focus on the eye of the object.</td>
</tr>
<tr>
<td>(6)</td>
<td>Large Zone</td>
<td>Same as (5), but more points.</td>
</tr>
<tr>
<td>(7)</td>
<td>Auto AF Selection (all 61 Points)</td>
<td>This may be useful for birds in the sky, if there is sufficient depth of focus. You don’t know which point is actually used. Don’t use this method if you want to have the focus on the eye of the object.</td>
</tr>
</tbody>
</table>

Pink AF Menü (second from left) --> 4 --> Selectable AF Point
Here you can reduce the number of selectable points. His choice: 61 or 15, because then you can choose the best point very fast.

Grant Atkinson: Canon 5D Mk IV - Autofocus: Part 3/4 - Prioritizing Your Autofocus Options
https://www.youtube.com/watch?v=VOiiQs1UEi8

Pink AF Menü (second from left) --> 2
Here you can set the priorities for the first picture and for all subsequent pictures. His choice: 1th image: RELEASE, 2nd image: 0 to -2
Focus Priority | This means the first picture is taken not before the focus is found. This may lead to pauses, if no focus is found.
--- | ---
Speed Priority | This means that less time is used for focusing.

Grant Atkinson: Canon 5D Mk IV - Autofocus: Part 4/4 - AF Cases

[https://www.youtube.com/watch?v=vp8sHvGArgg](https://www.youtube.com/watch?v=vp8sHvGArgg)

Pink AF Menü (second from left) --> 1

The "cases" contain predefined settings. He doesn't use them, however he has put the three settings (Tracking Sensitivity, Accel/decel tracking and AF pt auto switching) into "MyMenu". This can be done as follows:
Press Q, until "My Menu" is selected.
Add My Menu Tab, OK
Configure MyMenu1
Select items to register.

Now select the three items that were mentioned above. They are now available in "My Menu".

<table>
<thead>
<tr>
<th>Tracking Sensitivity</th>
<th>This is by far the most important parameter! It describes how easily the focus can move away from the previously found focus.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accel/decel tracking</td>
<td>This is difficult to understand. Best if you leave it at 0.</td>
</tr>
<tr>
<td>AF pt auto switching</td>
<td>This describes, how fast the camera switches from one AF point to a neighbor AF point. He leaves it at 0, which means deactivated.</td>
</tr>
</tbody>
</table>
Grant Atkinson: Canon 5D Mark IV - Settings For Wildlife Photography
https://www.youtube.com/watch?v=yy_72JQ-QT4

**Red Camera Menu (first from left)**

<table>
<thead>
<tr>
<th>Page 1</th>
<th>Lens aberration correction</th>
<th>He switches all options off, so that the pictures can be saved faster.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 2</td>
<td>Auto Lighting Optimizer</td>
<td>OFF</td>
</tr>
<tr>
<td>Page 3</td>
<td>High ISO speed NR</td>
<td>OFF</td>
</tr>
<tr>
<td>Page 2</td>
<td>ISO speed settings</td>
<td>AUTO 100 - 12800 for both ranges</td>
</tr>
<tr>
<td>Page 1</td>
<td>Release Shutter without card</td>
<td>Disable</td>
</tr>
</tbody>
</table>

**Pink AF Menü (second from left)**

| Page 4  | Auto AF pt sel: EOS iTR AF  | OFF                                                                 |

Grant Atkinson: Shooting Canon 5D Mark IV in M mode with auto ISO
https://www.youtube.com/watch?v=Xmud7-O8HNs

You can use the M mode together with "Auto ISO". Exposure compensation is also possible in M mode.

Tony & Chelsea Northrup: How to Photograph Flying Birds
https://www.youtube.com/watch?v=GFghMNX9zrl

Shutter: 1/2000s TV, Auto ISO might be useful

<table>
<thead>
<tr>
<th>Birds in front of trees or water</th>
<th>Use a single AF point and hold it on the object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Birds in the sky</td>
<td>It's easier to use all AF points.</td>
</tr>
</tbody>
</table>
### 18.1 GH5S Record formats

<table>
<thead>
<tr>
<th>Record format</th>
<th>Bits</th>
<th>Video Codec</th>
<th>Audio Codec</th>
<th>Anamorphic</th>
<th>VFR</th>
<th>HLG</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVCHD</td>
<td>8</td>
<td>?</td>
<td>?</td>
<td>no</td>
<td>some</td>
<td>no</td>
<td>This data format is suitable for when playing back on a high-definition TV, etc.</td>
</tr>
<tr>
<td>MP4</td>
<td>8</td>
<td>?</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>This data format is suitable for when playing back on a PC, etc.</td>
</tr>
<tr>
<td>MP4 HEVC (High Efficiency Video Coding)</td>
<td>10</td>
<td>h.265</td>
<td>?</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>This data format is for HDR motion picture and suitable for playback on a HDR (HLG format)-compatible TV or recorder.</td>
</tr>
<tr>
<td>MP4 (LPCM)</td>
<td>8 or 10</td>
<td>h.264</td>
<td>LPCM (uncompressed)</td>
<td>possible</td>
<td>some</td>
<td>only 10 bit</td>
<td>The MP4 data format for image editing.</td>
</tr>
<tr>
<td>MOV</td>
<td>8 or 10</td>
<td>h.264</td>
<td>?</td>
<td>possible</td>
<td>some</td>
<td>only 10 bit</td>
<td>Data format for image editing.</td>
</tr>
</tbody>
</table>
18.2 GH5S Exposing for VLog-L

See also: https://business.panasonic.co.uk/professional-camera/sites/default/eu-files/professional-camera-2014/case_study_pdf/The%20DVX200%20Book.pdf (especially the diagram on page 93)

See also: https://pro-av.panasonic.net/en/dvx4k/pdf/ag-dvx200_tech_brief_vol6_en.pdf

The following is taken from the above links:

With VLOG-L, the brightest clipped highlights will display on the zebra and on the waveform at about 80 IRE. Nothing brighter than about 81 IRE will ever be displayed.

Exposing to the right (ETTR):
This is a technique based on using a histogram for exposure. The general idea behind ETTR is to expose the image as bright as you possibly can, so long as none of the video information “clips” off the top. If required, you can always push it back down to proper exposure in post. Clipping occurs at 80 IRE. If you set your zebras at 80 IRE, you are free to expose up until the zebras appear. Anywhere that the zebras are displayed, you’ve clipped the image and would need to back off your exposure. Do be aware though that at higher exposure levels, even though the luminance may not have clipped yet, an individual color channel may begin clipping before the zebras display. As such, you might want to back off a little more (by setting the zebras no higher than 75 IRE), to leave a little room to minimize any clipping of chroma channels. When exposing using ETTR, skin tones may end up being recorded brighter or darker in every scene, simply based on where the highlights happen to be in that particular shot, and every shot will need to be corrected to bring the skin tones back to a reasonably consistent level so that your footage will intercut cleanly and seamlessly. And, depending on just how bright the highlights are in any given scene, ETTR may result in a scenario where the skin tones and midtones are significantly underexposed in an effort to catch and preserve all the highlights. Generally, cinematography is (and should be) more about the subject than it should be about the highlights; excessive attention to the highlights may mean compromising other aspects of the footage, so a strict “ETTR” approach is not always going to provide the overall best results in a video project.
Exposing For Middle Gray:
An alternative method of exposure is to expose for middle gray = 18% gray. When exposing for middle gray, you'll find the zebras and the waveform monitor vastly more useful than the histogram. In VLOG-L, middle gray is properly exposed at 42 IRE. VLOG-L gamma curve maps the following brightness levels to the following IRE levels:

<table>
<thead>
<tr>
<th>Reflectance</th>
<th>IRE</th>
<th>10-bit code value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0% (black)</td>
<td>7.3</td>
<td>128</td>
</tr>
<tr>
<td>18% (middle gray)</td>
<td>42</td>
<td>433</td>
</tr>
<tr>
<td>90% (white)</td>
<td>61</td>
<td>602</td>
</tr>
<tr>
<td>absolute clipped superwhite</td>
<td>80 IRE</td>
<td></td>
</tr>
</tbody>
</table>

In VLOG-L, the curve is laid out so that there are 8 stops below middle gray, and 4 stops above middle gray. You can, of course, choose to modify that by underexposing middle gray some; if you underexpose by one stop, you'll then have 7 stops below middle gray and 5 stops above it. In all cases you'll get 12 stops of dynamic range; the recommended allocation is for middle gray to be at 42 IRE with 8 stops below and 4 stops above, but you can shift that on an as-needed basis, so long as you account for it in post. The advice is to expose middle gray at 42 IRE whenever possible.

Using Zebras and Waveform Monitor:
With VLOG-L placing middle gray at 42 IRE, 90% white at 61 IRE, and black at 7 IRE gives a wide exposure range that allows for 4 stops of exposure over middle gray, and 8 stops under middle gray. Using these general exposure levels, you'll find that properly-exposed highlights on skin tones will usually range between about 42 IRE for dark-skinned subjects up to a maximum of about 55 IRE for light-skinned subjects.
For VLOG-L, it's recommendes to set Zebra 1 at 55 IRE and Zebra 2 at 75 IRE. If you have your zebras higher than 80, they will never trigger.

Summary:
Exposing properly for VLOG-L is the key to getting the best results; aim to expose an 18% gray card at about 42 IRE, keep your Caucasian skin highlights to below 55 IRE, and set your Zebra 2 to 75 IRE to keep from clipping highlights.

Some people recommend to use +1 stop exposure compensation (which means one stop overexposed).

The formula for the VLog curve is given in the V-Log/V-Gamut Reference manual:
VLog table:

<table>
<thead>
<tr>
<th>Stops = \log_{2}(\text{in} / 0.18)</th>
<th>\text{in} = 0.18 \times 2^\text{Stops}</th>
<th>\text{out}</th>
<th>\text{out} \times 1023</th>
<th>\text{IRE} = -7.24 + 116.33 \times \text{out}</th>
<th>\text{curve}</th>
</tr>
</thead>
<tbody>
<tr>
<td>-\infty</td>
<td>0 (0% black)</td>
<td>0.1250</td>
<td>128.0</td>
<td>7.3</td>
<td>linear</td>
</tr>
<tr>
<td>-8</td>
<td>0.000703125</td>
<td>0.1290</td>
<td>131.9</td>
<td>7.8</td>
<td>linear</td>
</tr>
<tr>
<td>-7</td>
<td>0.00140625</td>
<td>0.1329</td>
<td>135.9</td>
<td>8.2</td>
<td></td>
</tr>
<tr>
<td>-6</td>
<td>0.0028125</td>
<td>0.1407</td>
<td>144.0</td>
<td>9.1</td>
<td></td>
</tr>
<tr>
<td>-5</td>
<td>0.005625</td>
<td>0.1565</td>
<td>160.1</td>
<td>11.0</td>
<td></td>
</tr>
<tr>
<td>-4.17</td>
<td>0.01</td>
<td>0.1810</td>
<td>185.2</td>
<td>13.8</td>
<td></td>
</tr>
<tr>
<td>-4.17</td>
<td>0.01</td>
<td>0.1810</td>
<td>185.2</td>
<td>13.8</td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td>0.01125</td>
<td>0.1878</td>
<td>192.1</td>
<td>14.6</td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>0.0225</td>
<td>0.2346</td>
<td>240.0</td>
<td>20.1</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>0.045</td>
<td>0.2915</td>
<td>298.2</td>
<td>26.7</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>0.09</td>
<td>0.3554</td>
<td>363.5</td>
<td>34.1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0.18 (18% middle gray)</td>
<td>0.4233</td>
<td>433.0</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.36</td>
<td>0.4936</td>
<td>504.9</td>
<td>50.2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.72</td>
<td>0.5650</td>
<td>578.0</td>
<td>58.5</td>
<td></td>
</tr>
<tr>
<td>2.32</td>
<td>0.90 (90% white)</td>
<td>0.5882</td>
<td>601.7</td>
<td>61</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.44</td>
<td>0.6371</td>
<td>651.7</td>
<td>66.9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.88</td>
<td>0.7095</td>
<td>725.8</td>
<td>75.3</td>
<td></td>
</tr>
<tr>
<td>4.559</td>
<td>4.24246</td>
<td>0.7500</td>
<td>767.2</td>
<td>80 (maximum value for VLog-L)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.76</td>
<td>0.7820</td>
<td>800.0</td>
<td>83.7 (only available in VLog)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>11.52</td>
<td>0.8546</td>
<td>874.3</td>
<td>92.2 (only available in VLog)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>23.04</td>
<td>0.9273</td>
<td>948.6</td>
<td>100.6 (only available in VLog)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>46.08</td>
<td>1.0000</td>
<td>1023</td>
<td>109.1 (only available in VLog)</td>
<td></td>
</tr>
</tbody>
</table>
The function for converting from linear signal to V-Log data is as follows.

With linear reflection as “in” and V-Log data as “out”,

\[
\text{out} = \begin{cases} 
5.6 \times \text{in} + 0.125 & (\text{in} < \text{cut1}) \\
\log_{10}(\text{in} + b) + d & (\text{in} \geq \text{cut1})
\end{cases}
\]

with \( \text{cut1} = 0.01, \ b = 0.00873, \ c = 0.241514, \ d = 0.598206, \ 0 \leq \text{out} \leq 1 \)

The function for reverting compressed V-Log data to linear reflection is as follows.

With V-Log data as “in” and linear reflection as “out”,

\[
\text{in} = \begin{cases} 
\frac{(\text{out} - 0.125)}{5.6} & (\text{out} < \text{cut2}) \\
10^{\frac{(\text{out} - d)}{c}} - b & (\text{out} \geq \text{cut2})
\end{cases}
\]

with \( \text{cut2} = 0.181, \ 0 \leq \text{out} \leq 1 \)

This batch file makes a 10-bit VLog test video with 18 vertical bars. The brightness levels (from left to right) are black and from -8 to +8 stops:

```
set "T=10"                   :: Duration in seconds
rem Make a 10-bit VLog video:
ffmpeg -f lavfi -i nullsrc=s=svga,format=gray16 -lavfi
geq=lum='st(0, trunc(18*X/W));64*(128*eq(ld(0),0)+132*eq(ld(0),1)+136*eq(ld(0),2)+144*eq(ld(0),3)+160*eq(ld(0),4)+192*eq(ld(0),5)+240*eq(ld(0),6)+298*eq(ld(0),7)+363*eq(ld(0),8)+433*eq(ld(0),9)+505*eq(ld(0),10)+578*eq(ld(0),11)+652*eq(ld(0),12)+726*eq(ld(0),13)+800*eq(ld(0),14)+874*eq(ld(0),15)+949*eq(ld(0),16)+1023*eq(ld(0),17))',oscilloscope=tw=1:s=1
-pix_fmt yuv444p10le -color_range pc -crf 10 -c:v h264 -t %T% -y VLog_10bit.mov
```

Note: If "-color_range pc" is omitted, the video still plays fine in FFplay but it doesn't play correctly in VLC Player. The two brightest bars are shown with the same brightness.

It seems that if "-color_range" is not specified in a video, by default FFplay is assuming it's "PC" however VLC is assuming it's "TV".

**Important:** Always specify "-color_range"!
This is the VLOG-L curve:

(Source: https://business.panasonic.co.uk/professional-camera/sites/default/eu-files/professional-camera-2014/case_study_pdf/The%20DVX200%20Book.pdf)

The green line is the limit for the DVX200 camera. The limit for the GH5S is a little bit higher (approximately at 768).
18.3 GH5S HLG (Hybrid Log Gamma)

See also: https://en.m.wikipedia.org/wiki/Hybrid_Log-Gamma

HLG is a nonlinear transfer curve in which the lower half of the signal values use a gamma curve and the upper half of the signal values use a logarithmic curve.

\[ E' = r \cdot \sqrt{E} \] for \( 0 \leq E \leq 1 \)
\[ E' = a \cdot \ln(E - b) + c \] for \( 1 < E \)

where

- \( E \) is the signal normalized by the reference white level and \( E' \) is the resulting nonlinear signal
- \( r \) is the reference white level and has a signal value of 0.5
- the constants \( a, b, \) and \( c \) are defined as \( a = 0.17883277, b = 0.28466892, \) and \( c = 0.55991073 \)

The signal value is 0.5 for the reference white level while the signal value for 1 has a relative luminance that is 12 times higher than the reference white level. ARIB STD-B67 has a nominal range of 0 to 12.

A free HLG to Rec709 LUT is available here: https://nickdriftwood.com/product/hlg-rec709

This LUT is quite large (almost 10 MB) because the numbers have 10 decimal places, which is totally unnecessary. More than 3 or 4 make no sense.
18.4 GH5S Metering Modes

The four possible metering modes are "Multiple", "Centre Weighted", "Spot" and "Highlight Weighted". The last one was added in firmware version 1.4 and is missing in the manual.

18.5 GH5S Autofocus

There are two possible ways how to get rid of the "AF" icon in the lower right corner of the screen:

- Totally disable the touch screen as follows: Custom Menu --> Operation --> Touch_Settings --> Touch_Screen = OFF
- Use a video mode that allows variable frame rate (VFR), switch VFR to ON and set the variable frame rate to the same value as the rec frame rate, for example 25/25. In this case the "AF" icon disappears, and the camera does record audio. However at all other variable frame rates, it doesn't record audio.
18.6 GH5S Recommended settings

<table>
<thead>
<tr>
<th></th>
<th>Cinelike-D</th>
<th>VLOG-L</th>
<th>HLG (Hybrid Log Gamma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Phote Style&quot; in Exif Data</td>
<td>Unknown (10)</td>
<td>Unknown (13)</td>
<td></td>
</tr>
<tr>
<td>Contrast</td>
<td>0</td>
<td>[NA]</td>
<td>[NA]</td>
</tr>
<tr>
<td>Sharpness (1)</td>
<td>-5 ?</td>
<td>-5 ?</td>
<td>-5 ?</td>
</tr>
<tr>
<td>Noise Reduction (2)</td>
<td>-5</td>
<td>-5</td>
<td>-5</td>
</tr>
<tr>
<td>Saturation</td>
<td>-5</td>
<td>[NA]</td>
<td>-5</td>
</tr>
<tr>
<td>Hue</td>
<td>0</td>
<td>[NA]</td>
<td>0</td>
</tr>
<tr>
<td>Luminance Level</td>
<td>0-1023 (0-255 for 8 bit)</td>
<td>fixed at 32-200 (128-800 for 10 bit) (3)</td>
<td>fixed at 0-1023</td>
</tr>
<tr>
<td>Zebras</td>
<td>100%</td>
<td>75%</td>
<td>90%</td>
</tr>
<tr>
<td>Exposure compensation</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Possible ISO range for &quot;Dual Native ISO Settings&quot; = Low</td>
<td>80 - 800</td>
<td>320 - 1600</td>
<td>320 - 1600</td>
</tr>
<tr>
<td>Possible ISO range for &quot;Dual Native ISO Settings&quot; = High</td>
<td>800 - 204800</td>
<td>1600 - 25600</td>
<td>1600 - 204800</td>
</tr>
<tr>
<td>Dynamic range [F-Stops]</td>
<td>10.5</td>
<td>11.58</td>
<td>11.5</td>
</tr>
<tr>
<td>Notes</td>
<td>Best choice for video post processing!</td>
<td>Best choice for night sky!</td>
<td></td>
</tr>
</tbody>
</table>

(1) At higher ISO values (for example 25600), the sharpness setting is quite irrelevant, as there is no big difference in videos taken with sharpness -5 and +5. I'm unsure if negative sharpness values are a low pass filter or not.

(2) Any setting larger than -5 will suppress fainter stars in the night sky!

(3) V-LOG L uses only the range [128..800] (or 128..768?) from the possible range [0..1023], which means it's closer to 9-bit than to 10-bit
18.7 GH5S Custom settings C1, C2, C3-1, C3-2, C3-3

Up to 5 settings can be saved in Menu -> Settings -> Cust.Set Mem.
They can be loaded by turning the wheel to C1, C2 or C3.
In case of C3, you must additionally press the menu button and then select C3-1, C3-2 or C3-3.

My own settings:

<table>
<thead>
<tr>
<th>Rec Format</th>
<th>Pixel</th>
<th>fps</th>
<th>ISO, Photo Style</th>
<th>Exposure Mode, Exposure time</th>
<th>System Frequency</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1 [4K/10bit/150M/25p] 422 / 10Bit / Long GOP</td>
<td>3840x2160 4K</td>
<td>25</td>
<td>400, STD</td>
<td>M, 1/50s</td>
<td>50.00Hz (PAL)</td>
<td>For 4K videos</td>
</tr>
<tr>
<td>C2 [C4K/10bit/150M/25p] 422 / 10Bit / Long GOP</td>
<td>4096x2160 C4K</td>
<td>25</td>
<td>Auto, STD</td>
<td>M, 1/50s</td>
<td>50.00Hz (PAL)</td>
<td>For C4K videos</td>
</tr>
<tr>
<td>C3-1 [4K/A/150M/25p] 422 / 10bit / Long GOP</td>
<td>3328x2496 Anamorphic</td>
<td>25</td>
<td>51200 HLG, NR=-5</td>
<td>M, 1/25s</td>
<td>50.00Hz (PAL)</td>
<td>For meteor astronomy with SpeedBooster and Nippon Kogaku 8mm f/2.8 fisheye lens</td>
</tr>
<tr>
<td>C3-2 [FHD/8bit/100M/25p] 420 / 8Bit / Long GOP</td>
<td>1920x1080 FHD</td>
<td>125</td>
<td>51200, STD</td>
<td>M, 1/125s</td>
<td>50.00Hz (PAL)</td>
<td>For video astronomy: Variable framerate: 125 Ex.Tele Conv is OFF, but can be set to ON</td>
</tr>
<tr>
<td>C3-3 [FHD/8bit/100M/25p] 420 / 8Bit / Long GOP</td>
<td>1920x1080 FHD</td>
<td>25</td>
<td>51200, STD</td>
<td>M, 1/25s (1/2s - 1/25s)</td>
<td>50.00Hz (PAL)</td>
<td>For video astronomy: Variable framerate: off Ex.Tele Conv is OFF, but can be set to ON</td>
</tr>
</tbody>
</table>

My function key settings:

Fn1 Sound Rec Level Adj.
Fn2 Histogram
Fn3 Waveform Monitor
Fn4 Zebras
Fn5 Ex. Tele Conv.
18.8 GH5S Luminance level

Motion Picture > Luminance Level
Select the luminance range to match the use of video. Settings: [0-255]/[16-235]/[16-255]

- If you set Rec Quality to a 10bit motion picture setting, the available options change to [0-1023], [64-940], and [64-1023].
- This function works only for motion pictures. Still pictures (including those you take during motion picture recording) will be taken with [0-255].
- When Rec Format is set to AVCHD or MP4, [0-255] in Luminance Level will switch to [16-255].
- When Photo Style is set to Hybrid Log Gamma, setting is fixed to [0-1023]. The manual says [64-640], but I think this is wrong.
- When Photo Style is set to V-Log L, setting is fixed to [32-200] or [128-800]. The manual says [0-255], but I think this is wrong.

18.9 GH5S Master pedestal level

Creative Video > Master Pedestal Level

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>This side creates a high contrast image with a crisp atmosphere.</td>
</tr>
<tr>
<td>0</td>
<td>Standard</td>
</tr>
<tr>
<td>+</td>
<td>This side creates a slightly misty atmosphere.</td>
</tr>
</tbody>
</table>

This function is not available when Photo Style is set to V-Log L

18.10 GH5S Video size

<table>
<thead>
<tr>
<th>Mode</th>
<th>Resolution</th>
<th>Read-out Chip Size</th>
<th>Diagonal Size</th>
<th>Number of Pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K</td>
<td>3820 x 2160 (16:9)</td>
<td>18.8mm x 10.6mm</td>
<td>21.6mm</td>
<td>8251200</td>
</tr>
<tr>
<td>C4K</td>
<td>4096 x 2160 (17:9)</td>
<td>19.2mm x 10.12mm</td>
<td>21.7mm</td>
<td>8847360</td>
</tr>
<tr>
<td>Anamorphic</td>
<td>3328 x 2496 (4:3)</td>
<td>17.3mm x 13.0mm</td>
<td>21.6mm</td>
<td>8306688</td>
</tr>
<tr>
<td>FHD</td>
<td>1920 x 1080 (16:9)</td>
<td>18.8mm x 10.6mm (1)</td>
<td>21.6mm</td>
<td>2062800</td>
</tr>
<tr>
<td>FHD with 2.1x Extra Tele Conversion</td>
<td>1920 x 1080 (16:9)</td>
<td>8.95mm x 5.05mm</td>
<td>10.3mm</td>
<td>2062800</td>
</tr>
</tbody>
</table>

(1) Read-out chip size is smaller when frame rate is larger than 200
18.11 GH5S Mechanical / electronic shutter

Rec > Shutter Type

<table>
<thead>
<tr>
<th>Shutter Type</th>
<th>ISO</th>
<th>Exposure Time Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanical shutter</td>
<td>100-204800</td>
<td>60s - 1/8000s</td>
</tr>
<tr>
<td>Electronic shutter</td>
<td>204800</td>
<td>1/30s - 1/16000s</td>
</tr>
<tr>
<td></td>
<td>102400</td>
<td>1/15s - 1/16000s</td>
</tr>
<tr>
<td></td>
<td>51200</td>
<td>1/8s - 1/16000s</td>
</tr>
<tr>
<td></td>
<td>25600</td>
<td>1/4s - 1/16000s</td>
</tr>
<tr>
<td></td>
<td>12800</td>
<td>1/2s - 1/16000s</td>
</tr>
<tr>
<td></td>
<td>100 - 6400</td>
<td>1s - 1/16000s</td>
</tr>
</tbody>
</table>

18.12 GH5S Longer exposure time than framerate allows

When making a 25fps video, exposure times longer than 1/25s up to 1/2s are possible. Duplicated frames are written to the SD card.

At least these settings are required (there may be more requirements that I don’t know):

-- Creative film mode
-- Exposure mode "M"
-- Autofocus must be switched off at the lens
-- "SS/Gain Operation" must be set to "SEC/ISO"
-- Not in variable framerate mode
## 18.13 GH5S Variable frame rate

<table>
<thead>
<tr>
<th>System Frequency</th>
<th>Rec Quality</th>
<th>Available Framerates</th>
</tr>
</thead>
<tbody>
<tr>
<td>59.94Hz (NTSC)</td>
<td>[4K/8bit/100M/30p]</td>
<td>2 15 26 28 30 32 34 45 60</td>
</tr>
<tr>
<td></td>
<td>[FHD/24M/30p]</td>
<td>2 15 26 28 30 32 34 45 60</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/60p]</td>
<td>2 30 56 58 60 62 64 90 120 150 180 210 240</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/30p]</td>
<td>2 15 26 28 30 32 34 45 60 75 90 105 120 135 150 165 180 195 210 225 240</td>
</tr>
<tr>
<td></td>
<td>[4K/8bit/100M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60</td>
</tr>
<tr>
<td></td>
<td>[FHD/24M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60 72 84 96 108 120 132 144 156 168 180 192 204 216 228 240</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60 72 84 96 108 120 132 144 156 168 180 192 204 216 228 240</td>
</tr>
<tr>
<td>50.00Hz (PAL)</td>
<td>[4K/8bit/100M/25p]</td>
<td>2 12 21 23 25 27 30 37 60</td>
</tr>
<tr>
<td></td>
<td>[FHD/24M/25p]</td>
<td>2 12 21 23 25 27 30 37 60</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/50p]</td>
<td>2 25 46 48 50 52 54 75 100 125 150 200 240</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/25p]</td>
<td>2 12 21 23 25 27 30 37 50 62 75 87 100 112 125 137 150 175 200 225 240</td>
</tr>
<tr>
<td>24.00Hz (CINEMA)</td>
<td>[4K/8bit/100M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60</td>
</tr>
<tr>
<td></td>
<td>[C4K/8bit/100M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60 72 84 96 108 120 132 144 156 168 180 192 204 216 228 240</td>
</tr>
<tr>
<td></td>
<td>[FHD/8bit/100M/24p]</td>
<td>2 12 20 22 24 26 28 36 48 60 72 84 96 108 120 132 144 156 168 180 192 204 216 228 240</td>
</tr>
</tbody>
</table>

Note: Normally the GH5S doesn't record any audio in VFR mode. But there is one exception: When you switch VFR to ON and set the variable frame rate to the same value as the rec frame rate. For example, if "Rec Quality" is set to [FHD/8bit/100M/25p] then you can record audio only if the variable frame rate is set to 25.
18.14 Recording duration on SD cards

<table>
<thead>
<tr>
<th>Mbps</th>
<th>MB/s</th>
<th>MB/min</th>
<th>128GB card</th>
<th>256GB card</th>
<th>512GB card</th>
<th>640GB card (128GB + 512GB)</th>
<th>1024GB card (512GB + 512GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>50</td>
<td>3000</td>
<td>43 min = 0.7 h</td>
<td>87 min = 1.4 h</td>
<td>174 min = 2.9 h</td>
<td>218 min = 3.6 h</td>
<td>349 min = 5.8 h</td>
</tr>
<tr>
<td>200</td>
<td>25</td>
<td>1500</td>
<td>87 min = 1.4 h</td>
<td>174 min = 2.9 h</td>
<td>349 min = 5.8 h</td>
<td>436 min = 7.2 h</td>
<td>699 min = 11.6 h</td>
</tr>
<tr>
<td>150</td>
<td>18.75</td>
<td>1125</td>
<td>116 min = 1.9 h</td>
<td>233 min = 3.9 h</td>
<td>466 min = 7.8 h</td>
<td>582 min = 9.7 h</td>
<td>932 min = 15.5 h</td>
</tr>
<tr>
<td>100</td>
<td>12.5</td>
<td>750</td>
<td>174 min = 2.9 h</td>
<td>349 min = 5.8 h</td>
<td>699 min = 11.6 h</td>
<td>873 min = 14.5 h</td>
<td>1398 min = 23.3 h</td>
</tr>
<tr>
<td>72</td>
<td>9</td>
<td>540</td>
<td>242 min = 4.0 h</td>
<td>485 min = 8.1 h</td>
<td>970 min = 16.2 h</td>
<td>1213 min = 20.2 h</td>
<td>1941 min = 32.3 h</td>
</tr>
<tr>
<td>28</td>
<td>3.5</td>
<td>210</td>
<td>624 min = 10.4 h</td>
<td>1248 min = 20.8 h</td>
<td>2496 min = 41.6 h</td>
<td>3120 min = 52.0 h</td>
<td>4993 min = 83.2 h</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>180</td>
<td>728 min = 12.1 h</td>
<td>1456 min = 24.3 h</td>
<td>2912 min = 48.5 h</td>
<td>3640 min = 60.6 h</td>
<td>5825 min = 97.0 h</td>
</tr>
<tr>
<td>20</td>
<td>2.5</td>
<td>150</td>
<td>873 min = 14.5 h</td>
<td>1747 min = 29.1 h</td>
<td>3495 min = 58.2 h</td>
<td>4369 min = 72.8 h</td>
<td>6990 min = 116.5 h</td>
</tr>
<tr>
<td>17</td>
<td>2.125</td>
<td>127.5</td>
<td>1028 min = 17.1 h</td>
<td>2056 min = 34.3 h</td>
<td>4112 min = 68.5 h</td>
<td>5397 min = 89.9 h</td>
<td>8224 min = 137.0 h</td>
</tr>
</tbody>
</table>

18.15 GH5S Cable remote trigger

The cable remote trigger has a 2.5mm connector with 4 contacts:

<table>
<thead>
<tr>
<th>Contact</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tip</td>
<td>not connected</td>
</tr>
<tr>
<td>2nd</td>
<td>not connected</td>
</tr>
<tr>
<td>3rd</td>
<td>not pressed: 38.5 kΩ to ground (Difference to &quot;half presses&quot; is 33 kΩ)</td>
</tr>
<tr>
<td></td>
<td>half pressed: 5.5 kΩ to ground (Difference to &quot;full pressed&quot; is 3.3 kΩ)</td>
</tr>
<tr>
<td></td>
<td>full pressed: 2.2 kΩ to ground</td>
</tr>
<tr>
<td>Outer</td>
<td>ground</td>
</tr>
</tbody>
</table>
18.16 GH5S Cheap chinese battery adapters

If a 10kΩ resistor is soldered between the "-" and "T" contacts, the GH5S will accept all voltages from 6.5 to 8.5 Volts without any error messages. Without this resistance, the input voltage is much more critical. The original Panasonic DMW-AC10E power supply is rated 8.4V at 2.5A and the voltage is about 9.1V without load.

18.17 GH5S Telescopic effect

Set the [Ex. Tele Conv.] parameter to [ON] for a fixed 2.1x telescopic effect. This is on page 2/5 in the motion pictures menu. This function is not available when [HDR] is set to [ON], or when motion pictures size is set to [C4K] or [4K] in [Rec Quality], or when a frame rate of 150fps or higher is set for [Variable Frame Rate].

18.18 GH5S External HDMI

It's impossible to use USB-C for control, HDMI for monitor, and internal monitor operate at the same time. You have to pick a combination of two of these.

It's possible to capture the HDMI output signal with a cheap chinese HDMI to USB converter. The input resolution can be up to 4K and the output resolution is 1920x1080 maximum. This converter also accepts anamorphic 3328 x 2496 (4:3) input and converts it to 1920x1080 output, with black borders added at the left and right sides. It can also convert the 3328 x 2496 (4:3) input to 1600 x 1200 (4:3) output.

How to disable the overlays on the external HDMI output signal: Go to Video menu --> HDMI_Rec_Output --> Info_Display --> OFF

18.19 GH5S Synchro Scan

This is a fine adjustment of shutter speed, used to reduce flickering and horizontal stripes. It's only available when [Exposure Mode] is set to either [S] or [M] in Creative Video Mode.

Flicker-free calculator: https://www.red.com/flicker-free-video
### 18.20 GH5S Field of view with and without SpeedBooster 0.64x

<table>
<thead>
<tr>
<th>Lens</th>
<th>Effective focal length and f/ratio with SpeedBooster 0.64x</th>
<th>Field of view GH5S 4K ((18.8\text{mm} \times 10.6\text{mm}))</th>
<th>Field of view GH5S 4K ((17.3\text{mm} \times 13.0\text{mm})) with SpeedBooster 0.64x</th>
<th>Field of view GH5S Anamorphic ((4:3)) 4K ((17.3\text{mm} \times 13.0\text{mm})) with SpeedBooster 0.64x</th>
<th>Field of view Full Frame ((36\text{mm} \times 24\text{mm}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigma EX DG 4.5mm f/2.8</td>
<td>2.9mm f/1.8</td>
<td>180.0° x 155.1°</td>
<td>180.0° x 160.0°</td>
<td>180.0° x 160.0°</td>
<td>180.0° x 180.0°</td>
</tr>
<tr>
<td>Meike 6-11mm f/3.5</td>
<td>3.8mm-7.0mm f/2.2</td>
<td>180.0° x 126.4° @ 6mm</td>
<td>180.0° x 180.0° @ 6mm</td>
<td>180.0° x 155.0° @ 6mm</td>
<td>180.0° x 180.0° @ 6mm</td>
</tr>
<tr>
<td>Nippon Kogaku 8mm f/2.8</td>
<td>5.1mm f/1.8</td>
<td>147.1° x 83.0°</td>
<td>180.0° x 129.6°</td>
<td>135.4° x 101.7°</td>
<td>180.0° x 159.0°</td>
</tr>
<tr>
<td>Sigma EX DG 8mm f/3.5</td>
<td>5.1mm f/2.2</td>
<td>149.1° x 84.1°</td>
<td>180.0° x 131.3°</td>
<td>137.2° x 103.1°</td>
<td>180.0° x 161.1°</td>
</tr>
<tr>
<td>Canon EF 8-15mm f/4.0</td>
<td>5.1mm f/2.5</td>
<td>147.8° x 83.3° @ 8mm</td>
<td>180.0° x 130.2° @ 8mm</td>
<td>136.0° x 102.2° @ 8mm</td>
<td>180.0° x 160.0° @ 8mm</td>
</tr>
<tr>
<td>Canon EF 11-24mm f/4.0</td>
<td>7.0mm-15.4mm f/2.8</td>
<td>81.0° x 24.9° - 42.8° x 24.9°</td>
<td>106.3° x 73.9° - 62.9° x 38.1°</td>
<td>76.4° x 61.2° - 39.6° x 30.3°</td>
<td>101.7° x 85.4° - 58.8° x 45.9°</td>
</tr>
<tr>
<td>Leica DG 12-60mm f/2.8-4.0</td>
<td>--</td>
<td>76.1° x 47.7° - 17.8° x 10.1°</td>
<td>--</td>
<td>71.6° x 56.9° - 16.4° x 12.4°</td>
<td>--</td>
</tr>
<tr>
<td>Sigma EX DG 14mm f/1.8</td>
<td>9.0mm f/1.1</td>
<td>67.8° x 41.5°</td>
<td>21.2° x 12.1°</td>
<td>63.4° x 49.8°</td>
<td>88.0° x 71.9°</td>
</tr>
<tr>
<td>Canon CN-E 24mm T1.5 L F</td>
<td>15.4mm T 0.96</td>
<td>42.8° x 24.9°</td>
<td>62.9° x 38.1°</td>
<td>39.6° x 30.3°</td>
<td>58.8° x 45.9°</td>
</tr>
<tr>
<td>Sigma 24mm f/1.4</td>
<td>15.4mm f/0.9</td>
<td>42.8° x 24.9°</td>
<td>62.9° x 38.1°</td>
<td>39.6° x 30.3°</td>
<td>58.8° x 45.9°</td>
</tr>
<tr>
<td>Laowa 24mm f/4</td>
<td>15.4mm f/4.0</td>
<td>42.8° x 24.9°</td>
<td>62.9° x 38.1°</td>
<td>39.6° x 30.3°</td>
<td>58.8° x 45.9°</td>
</tr>
<tr>
<td>Canon EF 24-70mm f/4.0</td>
<td>15.4mm-44.8mm f/2.5</td>
<td>42.8° x 24.9° - 15.3° x 8.66°</td>
<td>62.9° x 38.1° - 23.7° x 13.5°</td>
<td>39.6° x 30.3° - 14.1° x 10.6°</td>
<td>58.8° x 45.9° - 21.9° x 16.5°</td>
</tr>
<tr>
<td>Sigma 50mm f/1.4</td>
<td>32mm f/0.9</td>
<td>21.3° x 12.1°</td>
<td>8.9° x 4.7°</td>
<td>19.6° x 14.8°</td>
<td>30.3° x 23.0°</td>
</tr>
<tr>
<td>Canon CN-E 85mm T1.3 L F</td>
<td>54.4mm T 0.83</td>
<td>12.6° x 7.14°</td>
<td>19.8° x 11.1°</td>
<td>11.6° x 8.75°</td>
<td>18.1° x 13.6°</td>
</tr>
<tr>
<td>Canon EF 100mm f/2.8</td>
<td>64mm f/2.8</td>
<td>10.7° x 6.07°</td>
<td>16.7° x 9.47°</td>
<td>9.9° x 7.44°</td>
<td>15.4° x 11.6°</td>
</tr>
<tr>
<td>Canon EF 100-400mm f/4.5-5.6</td>
<td>64mm-256mm f/2.8-3.5</td>
<td>10.7° x 6.07° - 2.69° x 1.52°</td>
<td>16.7° x 9.47° - 4.21° x 2.37°</td>
<td>9.8° x 7.44° - 2.48° x 1.86°</td>
<td>15.4° x 11.6° - 3.87° x 2.91°</td>
</tr>
<tr>
<td>Canon EF 200mm f/2.0</td>
<td>128mm f/1.2</td>
<td>5.38° x 3.04°</td>
<td>8.40° x 4.74°</td>
<td>4.95° x 3.72°</td>
<td>7.73° x 5.81°</td>
</tr>
<tr>
<td>Canon EF 400mm f/2.8</td>
<td>256mm f/1.8</td>
<td>2.69° x 1.52°</td>
<td>4.21° x 2.37°</td>
<td>2.48° x 1.86°</td>
<td>3.87° x 2.91°</td>
</tr>
<tr>
<td>+ 1.4x Teleconverter 560mm f/4.0</td>
<td>--</td>
<td>1.92° x 1.08°</td>
<td>--</td>
<td>1.77° x 1.33°</td>
<td>--</td>
</tr>
<tr>
<td>+ 2x Teleconverter 800mm f/7.6</td>
<td>--</td>
<td>1.35° x 0.76°</td>
<td>--</td>
<td>1.24° x 0.93°</td>
<td>--</td>
</tr>
<tr>
<td>Canon EF 500mm f/4.0</td>
<td>320mm f/2.5</td>
<td>2.15° x 1.21°</td>
<td>3.37° x 1.90°</td>
<td>1.98° x 1.49°</td>
<td>3.10° x 2.33°</td>
</tr>
<tr>
<td>+ 1.4x Teleconverter 700mm f/5.6</td>
<td>--</td>
<td>1.54° x 0.87°</td>
<td>--</td>
<td>1.42° x 1.06°</td>
<td>--</td>
</tr>
<tr>
<td>+ 2x Teleconverter 1000mm f/8.8</td>
<td>--</td>
<td>1.08° x 0.61°</td>
<td>--</td>
<td>0.99° x 0.74°</td>
<td>--</td>
</tr>
</tbody>
</table>

Fisheye lenses: \(\text{Field of view in degrees} = 180° \ast \frac{x}{\text{Image circle diameter}}\)

Normal lenses: \(\text{Field of view in degrees} = 2 \ast \arctan(x/2f)\)

with \(x = \text{image width or height in mm}\)

393
### 18.21 GH5S, all 77 video modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[C4K/8bit/150M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>59.94p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/10bit/150M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>29.97p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/ALL-I/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>23.98p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[C4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/150M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>59.94p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/ALL-I/400M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/ALL-I/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/72M/30p]</td>
<td>MP4 HEVC</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>72 Mbps</td>
<td>4:2:0/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/72M/24p]</td>
<td>MP4 HEVC</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>72 Mbps</td>
<td>4:2:0/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/100M/30p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/100M/24p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/150M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>59.94p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/400M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>29.97p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/A/150M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>29.97p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>23.98p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>Format</td>
<td>Codec</td>
<td>Profile</td>
<td>Audio Format</td>
<td>Frame Rate</td>
<td>Resolution</td>
<td>Bit Rate</td>
<td>Bit Depth</td>
<td>GOP Type</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
<td>---------</td>
<td>--------------</td>
<td>------------</td>
<td>------------</td>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>[4K/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>23.98p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/28M/60p]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94i</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/17M/60i]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94i</td>
<td>17 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/30p]</td>
<td>AVCHD</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94i</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/24p]</td>
<td>AVCHD</td>
<td>yes</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/28M/60p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/20M/30p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>20 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/24p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/150M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>4096x2160</td>
<td>50.00p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/10bit/150M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>4096x2160</td>
<td>25.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>4096x2160</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/150M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>50.00p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/ALL-I/400M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>Resolution</td>
<td>Codec</td>
<td>Container</td>
<td>Audio</td>
<td>Video</td>
<td>Frame Rate</td>
<td>Resolution</td>
<td>Bit Rate</td>
<td>Video Bit Depth</td>
<td>Audio Bit Depth</td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td>-----------</td>
<td>-------</td>
<td>-------</td>
<td>------------</td>
<td>------------</td>
<td>----------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>[4K/72M/25p]</td>
<td>MP4 HEVC</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>72 Mbps</td>
<td>4:2:0/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/100M/25p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/150M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>50.00p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/400M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>25.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/A/150M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>25.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/28M/50p]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/17M/50i]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00i</td>
<td>17 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/25p]</td>
<td>AVCHD</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00i</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/8bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/28M/50p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/20M/25p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>20 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/ALL-I/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>4096x2160</td>
<td>24.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[C4K/10bit/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>4096x2160</td>
<td>24.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>4096x2160</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/ALL-I/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3840x2160</td>
<td>24.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3840x2160</td>
<td>24.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>3840x2160</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3328x2496</td>
<td>24.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/A/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3328x2496</td>
<td>24.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>Rec Quality</td>
<td>Rec Format</td>
<td>VFR</td>
<td>HLG</td>
<td>System frequency</td>
<td>Size</td>
<td>Frame rate</td>
<td>Bit rate</td>
<td>YUV/bit</td>
<td>Image compression</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------</td>
<td>-----</td>
<td>-------</td>
<td>------------------</td>
<td>--------------</td>
<td>------------</td>
<td>----------</td>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td>[4K/A/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>3328x2496</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/24p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>

### 18.22 GH5S, all C4K 8 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[C4K/8bit/150M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>59.94p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/150M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>4096x2160</td>
<td>50.00p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>4096x2160</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>4096x2160</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>

### 18.23 GH5S, all C4K 10 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[C4K/10bit/150M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>29.97p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/ALL-I/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>23.98p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[C4K/10bit/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>4096x2160</td>
<td>23.98p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/10bit/150M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>4096x2160</td>
<td>25.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[C4K/10bit/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>4096x2160</td>
<td>24.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>

397
# GH5S, all 4K 8 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format / Bit rate</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4K/8bit/150M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>59.94p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/100M/30p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/100M/24p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/150M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>50.00p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/100M/25p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>3840x2160</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>
### 18.25 GH5S, all 4K 10 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4K/ALL-I/400M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/ALL-I/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/72M/30p]</td>
<td>MP4 HEVC</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>29.97p</td>
<td>72 Mbps</td>
<td>4:2:0/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/72M/24p]</td>
<td>MP4 HEVC</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3840x2160</td>
<td>23.98p</td>
<td>72 Mbps</td>
<td>4:2:0/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/ALL-I/400M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/72M/25p]</td>
<td>MP4 HEVC</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3840x2160</td>
<td>25.00p</td>
<td>72 Mbps</td>
<td>4:2:0/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/ALL-I/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3840x2160</td>
<td>24.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/10bit/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3840x2160</td>
<td>24.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>

### 18.26 GH5S, all anamorphic 8 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4K/A/150M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>59.94p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/150M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>50.00p</td>
<td>150 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>3328x2496</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>
## GH5S, all anamorphic 10 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4K/A/400M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>29.97p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/A/150M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>29.97p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>23.98p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/A/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>3328x2496</td>
<td>23.98p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/400M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>25.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/A/150M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>3328x2496</td>
<td>25.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[4K/A/400M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3328x2496</td>
<td>24.00p</td>
<td>400 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[4K/A/150M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>3328x2496</td>
<td>24.00p</td>
<td>150 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>
### 18.28 GH5S, all FHD 8 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FHD/28M/60p]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94p</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/17M/60i]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94i</td>
<td>17 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/30p]</td>
<td>AVCHD</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94i</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/24p]</td>
<td>AVCHD</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/28M/60p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94p</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/20M/30p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>20 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/24p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/28M/50p]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/17M/50i]</td>
<td>AVCHD</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00i</td>
<td>17 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/25p]</td>
<td>AVCHD</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00i</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/28M/50p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>28 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/20M/25p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>20 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/8bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>yes</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/24M/24p]</td>
<td>MP4</td>
<td>no</td>
<td>no</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>24 Mbps</td>
<td>4:2:0/8 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>
### GH5S, all FHD 10 bit modes

<table>
<thead>
<tr>
<th>Rec Quality</th>
<th>Rec Format</th>
<th>VFR</th>
<th>HLG</th>
<th>System frequency</th>
<th>Size</th>
<th>Frame rate</th>
<th>Bit rate</th>
<th>YUV/bit</th>
<th>Image compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FHD/ALL-I/200M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/60p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>59.94p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/30p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>29.97p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>59.94Hz (NTSC)</td>
<td>1920x1080</td>
<td>23.98p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/50p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>50.00p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/25p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>50.00Hz (PAL)</td>
<td>1920x1080</td>
<td>25.00p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
<tr>
<td>[FHD/ALL-I/200M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>200 Mbps</td>
<td>4:2:2/10 bit</td>
<td>ALL-Intra</td>
</tr>
<tr>
<td>[FHD/10bit/100M/24p]</td>
<td>MP4 (LPCM) / MOV</td>
<td>no</td>
<td>yes</td>
<td>24.00Hz (CINEMA)</td>
<td>1920x1080</td>
<td>24.00p</td>
<td>100 Mbps</td>
<td>4:2:2/10 bit</td>
<td>Long GOP</td>
</tr>
</tbody>
</table>
19 PanoView XDV360 camera

This is a very cheap chinese camera with a 200° 1.1mm f/2.0 fisheye lens.

Some hints for using:

Change the mode (Video / Photo / Timelapse / Settings) by short pressing the "on/off" button.
You can go directly to the settings by pressing the "arrow down" button.
Scroll in the settings with "arrow down" and "arrow up" buttons.
Switch to the right for the next menu with the "on/off" button.
Select and confirm with "start/stop" button.
Possible square video resolutions: 2448 / 2048 / 1440 / 1072 with 30 fps or 1440 / 1072 with 60 fps
Recommended exposure correction for video:
-- If the sun is in the field of view, use 0
-- In the woods, but sun is not directly visible: use 0 to +3
-- If in doubt, you aren't wrong if you use 0.

<table>
<thead>
<tr>
<th>Field of View</th>
<th>Top and left border</th>
<th>Width and height</th>
</tr>
</thead>
<tbody>
<tr>
<td>180°</td>
<td>176</td>
<td>2104</td>
</tr>
<tr>
<td>185°</td>
<td>144</td>
<td>2168</td>
</tr>
<tr>
<td>190°</td>
<td>116</td>
<td>2224</td>
</tr>
<tr>
<td>195°</td>
<td>88</td>
<td>2280</td>
</tr>
<tr>
<td>200°</td>
<td>60</td>
<td>2336</td>
</tr>
</tbody>
</table>
Kodak PIXPRO SP360 4K camera

This is a small camera with a 235° 0.85mm f/2.8 fisheye lens. The maximum video size is 2880 x 2880 pixels and in this mode the filesize is about 7.5MB per second.


Possible video resolutions in fisheye mode:

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Top and left border</th>
<th>Width and height</th>
</tr>
</thead>
<tbody>
<tr>
<td>2880x2880</td>
<td>210</td>
<td>2456</td>
</tr>
<tr>
<td>2048x2048</td>
<td>174</td>
<td>2528</td>
</tr>
<tr>
<td>1440x1440</td>
<td>142</td>
<td>2592</td>
</tr>
<tr>
<td>1072x1072</td>
<td>106</td>
<td>2664</td>
</tr>
<tr>
<td>720x720</td>
<td>74</td>
<td>2728</td>
</tr>
</tbody>
</table>

Charging the battery with the external battery charger: Lamp is red while charging and becomes green when battery is full. The charging time is at least 4 hours for a completely empty battery.

Charging the battery in the camera: Just plug in the USB cable and don't switch the camera on. The lamp is blinking orange while charging and goes off when the battery is full.

Error in instruction manual: The lamp is not continuously orange when the battery is charging.

Warning: The battery is always empty when you need it. Charge before using!

When recording, the LED is blinking red.

Test report (german): https://www.digitaley.de/Tests/Testbericht_Kodak_Pixpro_SP360_4K_360-Grad-Actioncam/9819
This is a cheap Chinese panoramic camera with two 220° fisheye lenses.

Video resolution: 1920x960@30fps, 2880x1440@25fps, 3840x1920@15fps, there is no exposure compensation setting

Lenses: f=0.88mm F/2.0, distance between the two lenses is about 26mm

Audio: 44.1kHz, 16-bit, stereo

The bitrate is about 12Mbps at 1920x960@30fps (1.5MB per second)

After downloading the video from the Micro SD card, it is already an equirectangular video and can be viewed as-is with the VLC player. The stitching is already done in the camera and there is no postprocessing required.

**Warning:** The battery is always empty when you need it. Charge before using!

Charging via USB: Blue lamp is on while charging, and goes off when battery is full.

If you mount this camera on a selfie stick, the stick itself isn't visible in the video. But it's shadow is visible! So it's a good idea to choose the diameter of the stick as small as possible.

It's difficult to find now. Search terms for Google: "Amkov Amköv VR 360 Handheld 4K WiFi Panorama Camera"

My Checklist:

- Battery fully charged?
- SD card installed?
- Have you set the desired video resolution?

Problem: Nothing happens when you press the "Record" button. Solution: You forgot to install the SD card.

Small problem: The two cameras aren't running exactly synchrone, meaning that the two frames aren't exposed exactly at the same time. It's noticeable when the video contains fast moving objects.
Ricoh Theta V

The LED is green while the battery is charging, and goes off when the battery is full. Press the upper button to switch the camera on. The LED will become blue (or green if the USB cable is connected). If it's blinking, the battery is low. The MODE button toggles between the modes STILL, VIDEO and LIVE STREAMING (only when a USB cable is connected). The LED below the camera symbol is blinking red while a video is recorded. At the bottom is a LED that warns if the memory is almost full (blinking) or full (permanently on). Recording a video: Press the large RECORD button, and press it again to stop recording.

The bitrate is about 57Mbps at 3840x1920@30fps (7.2MB per second)

The required apps can be downloaded here: https://support.theta360.com/en/download/

This is the workflow for making a 360° video with ambisonic sound:

• Download the video from the camera. The filename is R001xxxx.MP4 where xxxx is a 4-digit number. This video contains two circular fisheye images from the two lenses and isn't yet stitched together. The sound is mono, this is the ambisonic R channel. The ambisonic X, Y and Z channels are hidden in the MP4 container. I don't know if they can be extracted with FFmpeg. If you know it (without the "Ricoh Theta Movie Converter App") please let me know.

• Drag and drop this video to the "Ricoh Theta Basic app". This app stitches the hemispheres together to an equirectangular video. The output filename is R001xxxx_er.MP4 and the sound is the same mono sound as before, with the X, Y and Z channels still hidden in the MP4 container.

• For converting the mono audio channel to 4 ambisonic channels RYZX, drag and drop the equirectangular *.MP4 video to the "Ricoh Theta Movie Converter App". This conversion is quite fast and the output video has the filename R001xxxx_er.mov. This video has 4 audio channels and these are the ambisonic channels R, Y, Z, and X in this order. This video can be played for example with VLC player, including ambisonic sound. Which means you can change the viewing direction with the mouse, and the sound will change accordingly. This video can also be uploaded to Facebook.

• The camera has 4 built-in microphones, but for true ambisonic sound you need the TA-1 microphone. Surprisingly there is absolutely no difference in the EXIF data. The only way for finding out if a video was taken with or without TA-1 seems to be to check with a player and headphones. If I'm wrong, please let me know.

The workflow is also described in this thread: https://community.theta360.guide/t/youtube-spatial-audio-support-now-available/1675
A helpful Youtube video: https://www.youtube.com/watch?v=w8q3sFmNN8Y
22.1 Ambisonics


This is a very good introduction to Ambisonics: [https://www.researchgate.net/publication/280010078_Introduction_to_Ambisonics](https://www.researchgate.net/publication/280010078_Introduction_to_Ambisonics).


You can use this batch file for extracting the four R, Y, Z, and X audio channels from the *.mov file:

```bash
set "IN=R0010009_er.mov" :: *.MOV Input video
ffmpeg -i %IN% -map_channel 0.1.0 -y R.wav -map_channel 0.1.1 -y Y.wav -map_channel 0.1.2 -y Z.wav -map_channel 0.1.3 -y X.wav
pause
```

I did use this batch file for verifying that the four channels are really different from each other. All difference files are indeed non-zero:

```bash
set "IN=R0010009_er.mov" :: *.MOV Input video
ffmpeg -i %IN% -af "aeval=val(0)-val(1)" diff01.wav
ffmpeg -i %IN% -af "aeval=val(0)-val(2)" diff02.wav
ffmpeg -i %IN% -af "aeval=val(0)-val(3)" diff03.wav
ffmpeg -i %IN% -af "aeval=val(1)-val(2)" diff12.wav
ffmpeg -i %IN% -af "aeval=val(1)-val(3)" diff13.wav
ffmpeg -i %IN% -af "aeval=val(2)-val(3)" diff23.wav
pause
```
This batch file shows the volume of the four R, Y, Z and X audio channels over time:

```
set "IN=R0010009_er.mov" :: *.MOV Input video

ffmpeg -i %IN% -lavfi showwavespic=split_channels=1:s=1024x800 -y waveform.png

pause
```

The coordinate system used in Ambisonics follows the right hand rule convention with positive X pointing forwards, positive Y pointing to the left and positive Z pointing upwards. Horizontal angles run anticlockwise from due front and vertical angles are positive above the horizontal, negative below. (Source: Wikipedia)
22.2 Making 360° test videos with ambisonic sound

This batch file reads an equirectangular input video and replaces the audio channel by an ambisonic test tone, which is coming from a specified direction.

```
set "IN=R0010008_er.mp4"     :: Equirectangular input video
set "T=10"                   :: Duration in seconds
set "F=440"                  :: Tone frequency in Hz
::
set "YAW=90"                 :: The yaw angle of the sound source is anticlockwise from the front:
::  0 is front, 90 is left, 180 is back, -90 is right
set "PITCH=45"               :: The pitch angle is positive upwards: 0 is front, 90 is up, -90 is down
::
set "SYCP=0.707"             :: sin(yaw) * cos(pitch)       Unfortunately you must manually calculate these
set "CYCP=0"                 :: cos(yaw) * cos(pitch)        values, because there are no expressions
set "SP=0.707"               :: sin(pitch)                   allowed in the options of the pan filter
::
set "OUT=test.mov"           :: Equirectangular output video with ambisonic sound

ffmpeg -i %IN% -f lavfi -i sine=f=%F%:r=48000 -lavfi [1]pan="4.0|c0=0.707*c0|c1=%SYCP%*c0|c2=%SP%*c0|c3=%CYCP%*c0"[a]
-map 0:0 -map [a] -c:v copy -t %T% -y %OUT%
```

Note: Before playing this video, you must inject the spatial metadata with the "Spatial Media Metadata Injector". Tick the boxes "My video is spherical (360)" and "My video has spatial audio (ambiX ACN/SN3D format)".

The injector is available here: https://github.com/google/spatial-media/releases/tag/v2.1
This batch file is similar to the previous example, but uses an equirectangular test video. The position of the sound source is marked in the equirectangular output video with the word "SOUND":

```plaintext
set "IN=1200.png"                   :: Test pattern from http://www.paulbourke.net/dome/testpattern/1200.png
set "T=10"                          :: Duration in seconds
set "F=440"                         :: Tone frequency in Hz

set "YAW=90"                        :: The yaw angle of the sound source is anticlockwise from the front:
:: 0 is front, 90 is left, 180 is back, -90 is right
set "PITCH=45"                      :: The pitch angle is positive upwards: 0 is front, 90 is up, -90 is down

set "SYCP=0.707"                    :: sin(yaw) * cos(pitch)       Unfortunately you must manually calculate these
set "CYCP=0"                        :: cos(yaw) * cos(pitch)        values, because there are no expressions
set "SP=0.707"                      :: sin(pitch)                   allowed in the options of the pan filter

set "OUT=ambisonic_test.mov"        :: Equirectangular output video with ambisonic sound

ffmpeg -loop 1 -i %IN% -lavfi -i color=black@0:s=2400x1200,format=rgba -f lavfi -i sine=f=%F%:r=48000 -lavfi [0]split[a][b];[a]transpose=1[left];[b]transpose=2,negate[right];[left] [right]hstack,v360=input=dfisheye:output=e:ih_fov=180:iv_fov=180:pitch=90[c];[2]pan="4.0|c0=0.707*c0|c1=%SYCP%*c0|c2=%SP %*c0|c3=%CYCP%*c0"[:1]drawtext="fontsize=80:text='SOUND':box=1:boxcolor=red:boxborderw=10:fontcolor=yellow:x=(w-text_w)/2:y=(h-text_h)/2",v360=e:e:rorder=pyr:pitch=-%PITCH%:yaw=%YAW%[d];[c][d]overlay -t %T% -y %OUT%

Note: Don't forget to inject the spatial metadata with the "Spatial Media Metadata Injector".
```

410
This batch file makes an ambisonic test video with a moving sound source. A mosquito is flying in the horizontal plane around the observer:

```plaintext
set "IN=equirectangular_test.png" :: Equirectangular image or video
set "F=550" :: Mosquito frequency in Hz (Sawtooth wave)
               :: (550Hz for female mosquitos, 600Hz for male mosquitos)
set "VOL=0.1" :: Volume
set "R=2" :: Time in seconds for the mosquito flying 360° around the observer
set "T=10" :: Duration in seconds
set "OUT=mosquito.mov" :: Equirectangular output video with ambisonic sound

ffmpeg -loop 1 -i %IN% -f lavfi -i aevalsrc='%VOL%*(0.5-mod(%F%*t,1)):c=mono:s=48000',aeval="0.707*val(0)|sin(2*PI/%R%*t)*val(0)|0|cos(2*PI/%R%*t)*val(0)" -ac 4 -t %T% -y %OUT%
```

Note: Don't forget to inject the spatial metadata with the "Spatial Media Metadata Injector".
22.3 Play ambisonic sound with 4 speakers

If you play a video with ambisonic sound for example with VLC player, the player will automatically handle the conversion from 4 ambisonic channels to 2 stereo channels for your headphones.

Things are more complicated if you want real ambisonic sound with four speakers, for example in a planetarium dome. I did buy a cheap USB audio device which supports the 5.1 channel layout.

This is one possible way to convert the W, Y, Z and X signals into signals for 4 speakers in the horizontal plane:

<table>
<thead>
<tr>
<th></th>
<th>( \frac{(W + X)}{2} )</th>
<th>( \frac{(W - Y)}{2} )</th>
<th>( \frac{(W - X)}{2} )</th>
<th>( \frac{(W + Y)}{2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>front</td>
<td>( ( W + X ) ) / 2</td>
<td>( ( W - Y ) ) / 2</td>
<td>( ( W - X ) ) / 2</td>
<td>( ( W + Y ) ) / 2</td>
</tr>
<tr>
<td>right</td>
<td>( ( W - Y ) ) / 2</td>
<td>( ( W + X ) ) / 2</td>
<td>( ( W + Y ) ) / 2</td>
<td>( ( W - X ) ) / 2</td>
</tr>
<tr>
<td>back</td>
<td>( ( W - X ) ) / 2</td>
<td>( ( W + Y ) ) / 2</td>
<td>( ( W + X ) ) / 2</td>
<td>( ( W - Y ) ) / 2</td>
</tr>
<tr>
<td>left</td>
<td>( ( W + Y ) ) / 2</td>
<td>( ( W - X ) ) / 2</td>
<td>( ( W - Y ) ) / 2</td>
<td>( ( W + X ) ) / 2</td>
</tr>
</tbody>
</table>

This batch file converts the W, Y, Z and X audio channels to Front, Left, Back and Right audio channels, which can be fed to four speakers. The Z channel is not used, because all four speakers are in the horizontal plane. In this example the division by 2 is omitted. That’s no problem if the volume isn’t too close to the upper limit. The 5.1 channel layout is used. This layout has 6 channels in this order: FL = front left, FR = front right, FC = front center, LFE = low frequency, BL = back left, BR = back right. The FC and LFE channels are unused and set to zero. Please note that the speakers must be positioned as follows: FL channel = front, FR channel = left, BL channel = back, BR channel = right.

```bash
set "IN=R0010013_er.mov" :: *.MOV Input video
ffmpeg -i %IN% -lavfi "pan=5.1|c0=c0+c3|c1=c0+c1|c2=0*c0|c3=0*c0|c4=c0-c3|c5=c0-c1" -c:v copy -y out.mp4
pause
```
This batch file converts an equirectangular input video with ambisonic sound to a fisheye output video with 5.1 sound for 4 speakers in the horizontal plane (FL channel = front, FR channel = left, BL channel = back, BR channel = right):

```
set "IN=R0010013_er.mov" :: Equirectangular input video
set "FOV=220" :: Output field of view in degrees
set "S=1200" :: Output width and height
set "OUT=fish.mov" :: Fisheye output video

ffmpeg -i %IN% -lavfi \
[0:0]v360=e:fisheye:h_fov=%FOV%:v_fov=%FOV%:pitch=90,scale=%S%:%S%;\[0:1\]pan="5.1|c0=c0+c3|c1=c0+c1|c2=0*c0|c3=0*c0|c4=c0-c3|c5=c0-c1" -y %OUT% \
pause
```

Note: In the previous examples theoretically you could also use the 4.0 channel layout, because anyway you don't need the FC and LFE channels. But then the player doesn't recognize that the file contains signals for 4 individual speakers which are 90° apart in the horizontal plane. That's why it's better to use the 5.1 channel layout and leave two of the channels unused.

This is another way for converting the W, Y, Z and X signals into signals for 4 speakers in the horizontal plane:

<table>
<thead>
<tr>
<th>Speaker</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>left front</td>
<td>( \frac{(W + 0.707 \times X + 0.707 \times Y)}{2} )</td>
</tr>
<tr>
<td>right front</td>
<td>( \frac{(W + 0.707 \times X - 0.707 \times Y)}{2} )</td>
</tr>
<tr>
<td>left back</td>
<td>( \frac{(W - 0.707 \times X + 0.707 \times Y)}{2} )</td>
</tr>
<tr>
<td>right back</td>
<td>( \frac{(W - 0.707 \times X - 0.707 \times Y)}{2} )</td>
</tr>
</tbody>
</table>

This batch file converts the W, Y, Z and X audio channels to FrontLeft, BackLeft, BackRight and FrontRight audio channels, which can be fed to four speakers. The Z channel is not used, because all four speakers are in the horizontal plane:

```
set "IN=R0010013_er.mov" :: * .MOV Input video

ffmpeg -i %IN% -lavfi pan="5.1|c0=c0+0.707*c1+0.707*c3|c1=c0+0.707*c1-0.707*c3|c2=0*c0|c3=0*c0|c4=c0-0.707*c1+0.707*c3|c5=c0-0.707*c1-0.707*c3" -c:v copy -y out.mp4 \
pause
```
Play an equirectangular video with ambisonic sound in a planetarium dome with 4 speakers. The output channel layout is 5.1 (FL, FR, FC, LFE, BL, BR), where FC and LFE are silent:

```
set "IN=mosquito.mov" :: Equirectangular input video with ambisonic sound
set "FOV=200" :: Field of view of fisheye projection
set "S=1080" :: Size of output video

ffmpeg -i %IN% -lavfi v360=e:fisheye:pitch=90:h_fov=%FOV%:v_fov=%FOV%:w=%S%:h=%S%;pan="5.1|
c0=c0+0.707*c1+0.707*c3|c1=c0+0.707*c1-0.707*c3|c2=0*c0|c3=0*c0|c4=c0-0.707*c1+0.707*c3|c5=c0-0.707*c1-0.707*c3" -q:v 2 -f nut - | ffplay -fs -autoexit -
```

This is the output of the USB 5.1 device on an oscilloscope, CH1 = FL, CH2 = FR, CH3 = BR, CH4 = BL:
This batch file creates an audio test file with 5.1 channel layout with 4 different sine frequencies for the FL, FR, BL and BR channels. The FC and LFE channels are unused and remain silent:

```plaintext
set "F1=250"                  :: Frequency for front left
set "F2=500"                  :: Frequency for front right
set "F3=1000"                 :: Frequency for back left
set "F4=2000"                 :: Frequency for back right
set "T=10"                    :: Duration in seconds
set "V=0.2"                   :: Volume
set "OUT=audio_test_4.wav"    :: Output filename

ffmpeg -lavfi "aevalsrc="%V%*sin(2*PI*%F1%*t)|%V%*sin(2*PI*%F2%*t)|0|0|%V%*sin(2*PI*%F3%*t)|%V%*sin(2*PI*%F4%*t):c=5.1" -t %T% -y %OUT%

ffmpeg -i %OUT% -lavfi "atrim=duration=0.025,showwavespic=split_channels=1:s=1024x800 -y audio_test_waveform.png"

pause
```
## Color temperature test with video lights

The L4500 video lights have an adjustable color temperature from 3200K to 5600K in 13 steps. At each color temperature I did take a picture of a white piece of paper with a Canon 5D-MK4, which was set to the same color temperature as the two lamps. Exposure time was 1/125s, f/5.6, ISO800. The lamps were set to 100% brightness and the diffusor was installed. The CR2 images were converted with IrfanView to 8-bit PNG and then the average value for the R, G and B channels were calculated with Fitswork.

<table>
<thead>
<tr>
<th>Color Temperature</th>
<th>Illuminance [lux] @ 1m</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>S = R + G + B</th>
<th>R / S</th>
<th>G / S</th>
<th>B / S</th>
</tr>
</thead>
<tbody>
<tr>
<td>3200 K</td>
<td>426 (Minimum)</td>
<td>140.4</td>
<td>132.5</td>
<td>137.3</td>
<td>410.2 (Minimum)</td>
<td>34.2% (Minimum)</td>
<td>32.3% (Maximum)</td>
<td>33.5%</td>
</tr>
<tr>
<td>3400 K</td>
<td>456</td>
<td>149.1</td>
<td>138.1</td>
<td>145.9</td>
<td>433.1</td>
<td>34.4%</td>
<td>31.9%</td>
<td>33.7% (Maximum)</td>
</tr>
<tr>
<td>3600 K</td>
<td>474</td>
<td>156.5</td>
<td>143.5</td>
<td>152.7</td>
<td>452.7</td>
<td>34.6%</td>
<td>31.7%</td>
<td>33.7% (Maximum)</td>
</tr>
<tr>
<td>3800 K</td>
<td>494</td>
<td>161.9</td>
<td>145.9</td>
<td>155.1</td>
<td>462.9</td>
<td>35.0%</td>
<td>31.5%</td>
<td>33.5%</td>
</tr>
<tr>
<td>4000 K</td>
<td>508</td>
<td>165.8</td>
<td>149.2</td>
<td>158.6</td>
<td>473.6</td>
<td>35.0%</td>
<td>31.5%</td>
<td>33.5%</td>
</tr>
<tr>
<td>4200 K</td>
<td>518</td>
<td>169.8</td>
<td>151.5</td>
<td>160.6</td>
<td>481.9</td>
<td>35.2%</td>
<td>31.4% (Minimum)</td>
<td>33.3%</td>
</tr>
<tr>
<td>4400 K</td>
<td>525</td>
<td>172.5</td>
<td>153.9</td>
<td>162.7</td>
<td>489.1</td>
<td>35.3%</td>
<td>31.5%</td>
<td>33.3%</td>
</tr>
<tr>
<td>4600 K</td>
<td>531</td>
<td>174.1</td>
<td>154.9</td>
<td>163.1</td>
<td>492.1</td>
<td>35.4%</td>
<td>31.5%</td>
<td>33.1%</td>
</tr>
<tr>
<td>4800 K</td>
<td>535 (Maximum)</td>
<td>175.9</td>
<td>156.5</td>
<td>163.3</td>
<td>495.7</td>
<td>35.5% (Maximum)</td>
<td>31.6%</td>
<td>32.9%</td>
</tr>
<tr>
<td>5000 K</td>
<td>535 (Maximum)</td>
<td>175.9</td>
<td>156.9</td>
<td>162.6</td>
<td>495.4</td>
<td>35.5%</td>
<td>31.7%</td>
<td>32.8% (Minimum)</td>
</tr>
<tr>
<td>5200 K</td>
<td>534</td>
<td>175.4</td>
<td>157.8</td>
<td>163.9</td>
<td>497.1 (Maximum)</td>
<td>35.3%</td>
<td>31.7%</td>
<td>33.0%</td>
</tr>
<tr>
<td>5400 K</td>
<td>530</td>
<td>174.5</td>
<td>158.0</td>
<td>164.5</td>
<td>497.0</td>
<td>35.1%</td>
<td>31.8%</td>
<td>33.1%</td>
</tr>
<tr>
<td>5600 K</td>
<td>502</td>
<td>169.8</td>
<td>156.1</td>
<td>164.5</td>
<td>490.4</td>
<td>34.6%</td>
<td>31.8%</td>
<td>33.5%</td>
</tr>
</tbody>
</table>

Hint: L4500 lights can be operated with one NP-F970 battery. It’s not required to connect two batteries simultaneously.

1 lux = 0.0929 fc (Foot-candle)  
1 fc = 10.76 lux

416
Same test for Neewer RGB660 lamp with diffusor installed, brightness is set to 100%:

<table>
<thead>
<tr>
<th>Color Temperature</th>
<th>Illuminance [lux] @ 1m</th>
<th>R</th>
<th>G</th>
<th>B</th>
<th>S = R + G + B</th>
<th>R / S</th>
<th>G / S</th>
<th>B / S</th>
</tr>
</thead>
<tbody>
<tr>
<td>3200 K</td>
<td>574</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3400 K</td>
<td>539 ???</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3600 K</td>
<td>597</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3800 K</td>
<td>662</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000 K</td>
<td>734</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4200 K</td>
<td>810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4400 K</td>
<td>892 (Maximum)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4600 K</td>
<td>805</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4800 K</td>
<td>730</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5000 K</td>
<td>661</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5200 K</td>
<td>597</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5400 K</td>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5600 K</td>
<td>468 (Minimum)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hue = 0°, red</td>
<td>373</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hue = 60°, yellow</td>
<td>684</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hue = 120°, green</td>
<td>568</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hue = 180°, cyan</td>
<td>799</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hue = 240°, blue</td>
<td>410</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hue = 300°, magenta</td>
<td>650</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## TASCAM DR-70D

<table>
<thead>
<tr>
<th>Main Menu</th>
<th>Sub Menu</th>
<th>Recommended Setting</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC</td>
<td>RECORD</td>
<td>ON / OFF</td>
<td>Choose the channels you want to use</td>
</tr>
<tr>
<td></td>
<td>PAN</td>
<td></td>
<td>Balance for monitoring the inputs, this doesn't affect the record</td>
</tr>
<tr>
<td></td>
<td>GAIN</td>
<td>LOW / MID / HIGH / HI+PLUS</td>
<td>Choose the input gain</td>
</tr>
<tr>
<td></td>
<td>INPUT</td>
<td>XLR/TRS</td>
<td>Choose the input</td>
</tr>
<tr>
<td>MONITOR</td>
<td>MIX</td>
<td></td>
<td>Don't care, this is only for the monitor output</td>
</tr>
<tr>
<td>INPUT</td>
<td>INPUT GAIN</td>
<td>MIC+PHANTOM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIMITER</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LOWCUT</td>
<td>OFF</td>
<td>High pass filter</td>
</tr>
<tr>
<td></td>
<td>DELAY</td>
<td>0</td>
<td>Delay time to channel 1</td>
</tr>
<tr>
<td></td>
<td>PHASE</td>
<td>OFF</td>
<td>Reverses the polarity</td>
</tr>
<tr>
<td>RECORD</td>
<td>FILE TYPE</td>
<td>STEREO</td>
<td>One or two stereo files will be written</td>
</tr>
<tr>
<td></td>
<td>FORMAT</td>
<td>WAV 24bit</td>
<td>Best quality</td>
</tr>
<tr>
<td></td>
<td>SAMPLE</td>
<td>44.1kHz / 48kHz / 96kHz / 192kHz</td>
<td>Use 96kHz for ultrasound conversion</td>
</tr>
<tr>
<td></td>
<td>DUAL REC</td>
<td>OFF or -1db to -12dB</td>
<td>This is only possible if channels 3 and 4 are deactivated</td>
</tr>
<tr>
<td>SLATE</td>
<td></td>
<td></td>
<td>Slate signal</td>
</tr>
<tr>
<td>MIC</td>
<td>MS MODE 1/2</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MS MODE 3/4</td>
<td>OFF</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PHANTOM VOLT</td>
<td>48V</td>
<td>Phantom voltage for Rode MT-1 microphones</td>
</tr>
<tr>
<td>OTHERS</td>
<td>SYSTEM --&gt; FORMAT</td>
<td></td>
<td>Formatting the SD card</td>
</tr>
<tr>
<td></td>
<td>BATTERY</td>
<td>NIMH / ALKAL</td>
<td>Battery type</td>
</tr>
<tr>
<td></td>
<td>DATE / TIME</td>
<td></td>
<td>Setting date and time</td>
</tr>
</tbody>
</table>
I typically make either 4-channel records, or 2-channel records with DUAL REC -10dB. These settings must be changed:

<table>
<thead>
<tr>
<th>Application</th>
<th>BASIC --&gt; RECORD CH3+4</th>
<th>RECORD --&gt; DUAL RECORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-Channel Recording</td>
<td>ON</td>
<td>OFF</td>
</tr>
<tr>
<td>2-Channel Recording with DUAL REC -10dB</td>
<td>OFF</td>
<td>-10dB</td>
</tr>
</tbody>
</table>

WAV 24bit 44.1 kHz, maximum recording length with 2GB SD card: 3h 22m

Always power the recorder with an external powerbank. The internal batteries are much too small, especially if phantom voltage is used.
<table>
<thead>
<tr>
<th>Menu</th>
<th>Recommended Setting</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 1/19 INPUT | GAIN: LINE / LOW / MID / HI / HI+  
SEL: IN 1-2 IN 3-4                                                                         | Choose the input gain and which inputs are used  
LOW: +20dB, MID: +40dB, HI: +52dB, HI+: +64dB                         |
| 2/19 MIXER | LVL: 100, 100, 100, 100  
PAN: L12, R12, L12, R12  
MS: OFF                                                                 | Important: If you set PAN to "C", both stereo channels are equal!   |
| 3/19 PHASE / DELAY | 0, OFF                                         | Reverse the polarity, set a delay time                                |
| 4/19 LEVEL CONTROL | OFF                                            |                                                                      |
| 5/19 TRIM GANG | GRP 1: all ON  
GRP2: all OFF                                                                 | Adjust all channels simultaneously with channel 1 knob               |
| 6/19 OUTPUT LEVEL | CAMERA: 30db  
LINE: 0db                                                                   | Set the output levels                                                 |
| 7/19 MIC POWER | PHAN: all ON, VOLTAGE : 48V  
PLUGIN: OFF                                                                       | Use 48V for RODE NT1 microphones  
PLUGIN is the supply voltage for microphones at the EXT IN 1/2 input |
| 8/19 RECORD | CH1, CH2, CH3, CH4: ON  
MIX: OFF  
DUAL: OFF or 1-2, -12dB                                                       | When DUAL mode is used, channels 3 and 4 are automatically deselected |
| 9/19 REC SETTING | FILE TYPE: STEREO  
FORMAT: WAV 24bit  
SAMPLE: 44.1kHz / 48kHz / 96kHz / 192kHz                                      | One or two stereo files will be written  
Use 44.1kHz or 48kHz for normal sound, or 96kHz for ultrasound         |
| 10/19 FILE | NAME TYPE: DATE  
WORD: TASCAM                                                                 |                                                                      |
| 11/19 MEDIA | FORMAT                                                                                     | Here you can format the SD card                                       |
| 12/19 TIME CODE |                                                                                              |                                                                      |
| 13/19 SLATE TONE | AUTO: OFF  
OSCILLATOR                                                                        | Use the OSCILLATOR feature for generating a -20dB test tone          |
<table>
<thead>
<tr>
<th>14/19 HDMI AUDIO ASSIGN</th>
<th>OFF</th>
</tr>
</thead>
<tbody>
<tr>
<td>15/19 AMBISONICS</td>
<td>OFF</td>
</tr>
</tbody>
</table>
| 16/19 METER/TRIM        | PEAK HOLD: 2sec  
                           | TRIM MIN: MIN |
| 17/19 POWER MANAGEMENT  | BATTERY TYPE: ALKALI  
                           | AUTO PWR SAVE: 30min  
                           | BACKLIGHT: 10sec |
| 18/19 REMOTE            |     |
| 19/19 SYSTEM            | DATE / TIME | Setting date and time |

I typically make either 4-channel records, or 2-channel records with DUAL REC -10dB. For toggling between these modes, only one setting must be changed: Set RECORD / DUAL to OFF or 1-2.

Always power the recorder with an external powerbank. The internal batteries are much too small, especially if phantom voltage is used.

Pinout of 3.5mm stereo connectors: Tip contact is left channel, middle contact is right channel, outer contact is ground.
25.1 Matching the DR-701D's output level to the GH5S' input level

The output level of the TASCAM DR-701D camera output can be set in the menu OUTPUT LEVEL / CAMERA in the range -24dB to +42dB. There are hardware switches between 0dB and 1dB, between 12dB and 13dB and between 30dB and 31dB.

A 1kHz test tone can be generated in the menu SLATE TONE / OSCILLATOR, with level -18dB or -20dB. The reference level seems to be about 62mV without load.

Output level at the TASCAM's camera output (measured with high impedance):

<table>
<thead>
<tr>
<th>OUTPUT LEVEL / CAMERA</th>
<th>Output voltage (OSCILLATOR = -18dB)</th>
<th>Output voltage (OSCILLATOR = -20dB)</th>
<th>Maximum 1kHz sine output voltage, just before clipping occurs in the output signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0dB</td>
<td>7.5 mV_rms</td>
<td>6.2 mV_rms</td>
<td>62 mV_rms</td>
</tr>
<tr>
<td>12dB</td>
<td>30.3 mV_rms</td>
<td>24.0 mV_rms</td>
<td>240 mV_rms</td>
</tr>
<tr>
<td>20dB</td>
<td>79.0 mV_rms</td>
<td>62.0 mV_rms</td>
<td>620 mV_rms</td>
</tr>
<tr>
<td>30dB</td>
<td>249.6 mV_rms</td>
<td>200.0 mV_rms</td>
<td>2.00 V_rms</td>
</tr>
<tr>
<td>40dB</td>
<td>795 mV_rms</td>
<td>622 mV_rms</td>
<td>3.35 V_rms</td>
</tr>
<tr>
<td>42dB</td>
<td>993 mV_rms</td>
<td>795 mV_rms</td>
<td>3.35 V_rms</td>
</tr>
</tbody>
</table>

The output level of the TASCAM DR-701D line output can be set in the menu OUTPUT LEVEL / LINE in the range -12dB to +12dB. There is a hardware switch between 0dB and 1dB.

Output level at the TASCAM's line output (measured with high impedance):

<table>
<thead>
<tr>
<th>OUTPUT LEVEL / LINE</th>
<th>Output voltage (OSCILLATOR = -18dB)</th>
<th>Output voltage (OSCILLATOR = -20dB)</th>
<th>Maximum 1kHz sine output voltage, just before clipping occurs in the output signal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>-12dB</td>
<td>62 mV_rms</td>
<td>49 mV_rms</td>
<td>0.5 V_rms</td>
</tr>
<tr>
<td>-3dB</td>
<td>175 mV_rms</td>
<td>139 mV_rms</td>
<td>1.41 V_rms</td>
</tr>
<tr>
<td>0dB</td>
<td>248 mV_rms</td>
<td>197 mV_rms</td>
<td>2.0 V_rms</td>
</tr>
<tr>
<td>12dB</td>
<td>990 mV_rms</td>
<td>785 mV_rms</td>
<td>3.27 V_rms</td>
</tr>
</tbody>
</table>
The input level of the Panasonic LUMIX GH5S can be set to "LINE" in the menu Motion_Picture --> Mic_Socket.
The Motion_Picture --> Sound_Rec_Level_Adj. parameter can be set in the -12dB to +6dB range.
For measuring the clipping voltage, make sure that Motion_Picture --> Sound_Rec_Level_Limiter is OFF.

<table>
<thead>
<tr>
<th>Sound Rec Level Adj.</th>
<th>Input voltage when level indicator is at -12dB mark</th>
<th>Maximum sine voltage before clipping occurs</th>
<th>Maximum peak voltage before clipping occurs</th>
</tr>
</thead>
<tbody>
<tr>
<td>-12dB</td>
<td>1050 mV_rms</td>
<td>4.88 V_rms</td>
<td>+/- 6.90 V</td>
</tr>
<tr>
<td>-6dB</td>
<td>525 mV_rms</td>
<td>2.44 V_rms</td>
<td>+/- 3.45 V</td>
</tr>
<tr>
<td>0dB</td>
<td>262 mV_rms</td>
<td>1.22 V_rms</td>
<td>+/- 1.73 V</td>
</tr>
<tr>
<td>+6dB</td>
<td>131 mV_rms</td>
<td>0.61 V_rms</td>
<td>+/- 0.86 V</td>
</tr>
</tbody>
</table>

So after all these measurements, what's a good match between the output level of the TASCAM and the input level of the GH5S?

- **TASCAM DR-701D** ↔ **Panasonic LUMIX GH5S**
  - Camera output 27dB or line output -3dB
  - Set microphone input to "LINE" and Sound_Rec_Level_Adj. to 0dB

Or alternatively:

- **TASCAM DR-701D** ↔ **Panasonic LUMIX GH5S**
  - Camera output 30dB or line output 0dB
  - Set microphone input to "LINE" and Sound_Rec_Level_Adj. to -3dB

With these settings both recorders get the same amplitude and clipping occurs at the same level.
The Apprehension Engine: Sound effects for horror films

This is a machine for creating sound effects for horror films. It was envisioned by movie composer Mark Korven and created by guitar maker Tony Duggan-Smith. [http://apprehensionengine.com/](http://apprehensionengine.com/)
The apprehension engine was used to make the sounds for the movie "The Witch": [https://en.wikipedia.org/wiki/The_Witch_(2015_film)](https://en.wikipedia.org/wiki/The_Witch_(2015_film))

Some videos by Jakob Balogh showing what you can do with this engine:
The Apprehension Engine - First Look Part 01 (Horror Machine) [https://www.youtube.com/watch?v=dSVzFD6bDwQ](https://www.youtube.com/watch?v=dSVzFD6bDwQ)
The Apprehension Engine - First Look Part 02 (Horror Machine) [https://www.youtube.com/watch?v=61Cw5vApw-o](https://www.youtube.com/watch?v=61Cw5vApw-o)
The Apprehension Engine - First Look Part 03 (Horror Machine) [https://www.youtube.com/watch?v=n5nAXLdBc40](https://www.youtube.com/watch?v=n5nAXLdBc40)

Other videos showing how to use the engine and similar instruments:
The Apprehension Engine - Horror Suite Part 1 [https://www.youtube.com/watch?v=QUYFMHM3wns](https://www.youtube.com/watch?v=QUYFMHM3wns)
The Apprehension Engine - Horror Suite Part 2 [https://www.youtube.com/watch?v=K9xE1UHDoLU](https://www.youtube.com/watch?v=K9xE1UHDoLU)
Apprehension Engine: Sound check in Chicago [https://www.youtube.com/watch?v=cgp76wROgxY](https://www.youtube.com/watch?v=cgp76wROgxY)
Horror Musical Instrument - The Apprehension Engine [https://www.youtube.com/watch?v=lzk-I8Gm0MY](https://www.youtube.com/watch?v=lzk-I8Gm0MY)
DIY Apprehension Engine 1 - Metallic Bones (Rulers) [https://www.youtube.com/watch?v=_q-yMks1NYg](https://www.youtube.com/watch?v=_q-yMks1NYg)
DIY Apprehension Engine 2 - Glass Shards (Wine Glass) [https://www.youtube.com/watch?v=1arnzEoAuAk](https://www.youtube.com/watch?v=1arnzEoAuAk)
DIY Apprehension Engine 3 - Gates of Hell (Spring Reverb) [https://www.youtube.com/watch?v=v7ygTQu1Eo](https://www.youtube.com/watch?v=v7ygTQu1Eo)
DIY Apprehension Engine 4 - Heavy Metal (Guitar) [https://www.youtube.com/watch?v=gh77BusFy4](https://www.youtube.com/watch?v=gh77BusFy4)
Latest build: "Horror Box 1.0" - (demo) - spring box w/ piezo microphone [https://www.youtube.com/watch?v=tEgUXJiDEIg](https://www.youtube.com/watch?v=tEgUXJiDEIg)

Here is a series of "How to Build The Apprehension Engine" videos by Michael Freudenberg on Youtube:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Youtube Link</th>
<th>Material needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 - HISTORY</td>
<td><a href="https://www.youtube.com/watch?v=xHXUEycuAMY">https://www.youtube.com/watch?v=xHXUEycuAMY</a></td>
<td>Pine wood: Width 40cm x Length 1.2 meters</td>
</tr>
<tr>
<td>#2 - The Base</td>
<td><a href="https://www.youtube.com/watch?v=uwZmU44Lp10">https://www.youtube.com/watch?v=uwZmU44Lp10</a></td>
<td>5 Ply board: (for the base) Width 40cm x Length 1.2 meters</td>
</tr>
<tr>
<td>#3 - The Sides</td>
<td><a href="https://www.youtube.com/watch?v=eZXM-Dj7IQw">https://www.youtube.com/watch?v=eZXM-Dj7IQw</a></td>
<td>Pine wood: Width 4.2 cm x Depth 1.9 cm x Length 1.2 meters</td>
</tr>
<tr>
<td>#4 - Finishing the Sides</td>
<td>[<a href="https://www.youtube.com/watch?v=JfEjMSTJ">https://www.youtube.com/watch?v=JfEjMSTJ</a> Ts](<a href="https://www.youtube.com/watch?v=JfEjMSTJ">https://www.youtube.com/watch?v=JfEjMSTJ</a> Ts)</td>
<td>MDF 16mm thick</td>
</tr>
<tr>
<td>#5 - Attaching rear support</td>
<td><a href="https://www.youtube.com/watch?v=SkLuiXLvTgw">https://www.youtube.com/watch?v=SkLuiXLvTgw</a></td>
<td></td>
</tr>
<tr>
<td>#6 - The Hurdy Gurdy Wheel</td>
<td><a href="https://www.youtube.com/watch?v=cL1_YBDax5s">https://www.youtube.com/watch?v=cL1_YBDax5s</a></td>
<td></td>
</tr>
</tbody>
</table>
#7 - The Hurdy Gurdy bearing  https://www.youtube.com/watch?v=xU8ES5OLxak
#8 - Installing the Front Frame  https://www.youtube.com/watch?v=3tSBeRqQSOE
#9 - Installing the Rear Frame  https://www.youtube.com/watch?v=XWddP1lneco
#10 - Guitar Neck Supports  https://www.youtube.com/watch?v=az5uFla6qBq
#11 - Left side Soundboard  https://www.youtube.com/watch?v=HhuUxrRij5E
#12 - Front and Right side Soundboard  https://www.youtube.com/watch?v=wlzUkMgLDhM
#13 - Installing Top Soundboard  https://www.youtube.com/watch?v=DBW_x_KKEdM  3 mm to 5 mm Ply wood (3 ply) 18mm x 18mm square pine wood
#14 - Installing the Hurdy Gurdy  https://www.youtube.com/watch?v=t1JzJRfPtW0
#15 - Making the Guitar Necks  https://www.youtube.com/watch?v=84oymSJ6L1w  Hard wood 65mm x 18mm x 1.2m Hard wood 40mm x 18mm x 1.2 meters
#16 - Making the Guitar Necks Part II  https://www.youtube.com/watch?v=lRAJwj0ZpoU
#17 - FINISHING THE BOX  https://www.youtube.com/watch?v=OJtZyos_ZaE
#18 – The Electronics and parts  https://www.youtube.com/watch?v=Ke14onBnITs  Two cello strings (G and C) for the large neck, and either three electric guitar strings or violin strings for the small neck. A pack of rosin. A bow with horsehair. An adjustable rosewood bridge for mandolin. A cigar box hard tail bridge saddle for 3 string guitar. Two 6” metal rulers and two 12” metal rulers. Blend 301 piezo preamp pickup mic EQ tuner for acoustic guitar. Pickup piezo transducer prewired volume. 3 guitar pickup piezo for acoustic guitar / ukelele / violin / mandolin.
#19 - Installing The Guitar Tuners  https://www.youtube.com/watch?v=oNs79OUh3AU  3 left and 3 right Guitar tuners, 3 string cigar box guitar bridge
#20 - The String Bridges  https://www.youtube.com/watch?v=8h9N7T8jA50
#21 - Installing the Humbucker pickup  https://www.youtube.com/watch?v=mRA0JK5aCFk  Circuit Wiring Harness Twin-coil Pickup HUMBUCKER 3-way switch Electric Guitar ebay has them for $11 or buy a 3 string cigar box pickup (like one shown in video). https://www.ebay.com.au/itm/Circuit-Wiring-Harness-Twin-coil-
<table>
<thead>
<tr>
<th>#22 - Installing the Piezo Contact Pickup</th>
<th><a href="https://www.youtube.com/watch?v=N1BU6MSp8Xs">https://www.youtube.com/watch?v=N1BU6MSp8Xs</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>#23 - Installing the Reverb Tank</td>
<td><a href="https://www.youtube.com/watch?v=keRG7eUaOww">https://www.youtube.com/watch?v=keRG7eUaOww</a></td>
</tr>
<tr>
<td>#24 - The Final Tutorial</td>
<td><a href="https://www.youtube.com/watch?v=QSJOtxwgd8Y">https://www.youtube.com/watch?v=QSJOtxwgd8Y</a></td>
</tr>
<tr>
<td>Learn how to add the cotton to the Hurdy Gurdy strings here:</td>
<td><a href="https://www.youtube.com/watch?v=0TTi5FoNKw8">https://www.youtube.com/watch?v=0TTi5FoNKw8</a></td>
</tr>
</tbody>
</table>

Reverb tank 4FB3A1A: Input 1475 Ω, Output 2250 Ω, Long decay 2.75s - 4s, Input grounded, Output grounded
This is the corrected block diagram of the Betagear FX82USB mixer:
27 Synthesizers and Midi

27.1 The keyboard

![Diagram of a piano keyboard with notes C to A# (ish)]
## 27.2 Frequencies of the keys

Factor between two notes: \(2^{\frac{1}{12}} = 1.05946\)

<table>
<thead>
<tr>
<th>Octave</th>
<th>c</th>
<th>cis/des</th>
<th>d</th>
<th>dis/es</th>
<th>e</th>
<th>f</th>
<th>fis/ges</th>
<th>g</th>
<th>gis/as</th>
<th>a</th>
<th>ais/b</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>32.703</td>
<td>34.648</td>
<td>36.708</td>
<td>38.891</td>
<td>41.203</td>
<td>43.654</td>
<td>46.249</td>
<td>48.999</td>
<td>51.913</td>
<td>55.000</td>
<td>58.270</td>
<td>61.735</td>
</tr>
<tr>
<td>3</td>
<td>65.406</td>
<td>69.296</td>
<td>73.416</td>
<td>77.782</td>
<td>82.507</td>
<td>87.307</td>
<td>92.499</td>
<td>97.999</td>
<td>103.83</td>
<td>110.00</td>
<td>116.54</td>
<td>123.47</td>
</tr>
<tr>
<td>4</td>
<td>130.81</td>
<td>138.59</td>
<td>146.83</td>
<td>155.56</td>
<td>164.81</td>
<td>174.61</td>
<td>185.00</td>
<td>196.00</td>
<td>207.65</td>
<td>220.00</td>
<td>233.08</td>
<td>246.94</td>
</tr>
<tr>
<td>5</td>
<td>261.63</td>
<td>277.18</td>
<td>293.66</td>
<td>311.13</td>
<td>329.63</td>
<td>349.23</td>
<td>369.99</td>
<td>392.00</td>
<td>415.30</td>
<td>440.00</td>
<td>466.14</td>
<td>493.88</td>
</tr>
<tr>
<td>6</td>
<td>523.25</td>
<td>554.37</td>
<td>587.33</td>
<td>622.25</td>
<td>659.26</td>
<td>698.46</td>
<td>739.99</td>
<td>783.99</td>
<td>830.61</td>
<td>880.00</td>
<td>932.33</td>
<td>987.77</td>
</tr>
<tr>
<td>7</td>
<td>1046.5</td>
<td>1108.7</td>
<td>1174.7</td>
<td>1244.5</td>
<td>1318.5</td>
<td>1396.9</td>
<td>1480.0</td>
<td>1568.0</td>
<td>1661.2</td>
<td>1760.0</td>
<td>1864.7</td>
<td>1975.5</td>
</tr>
<tr>
<td>8</td>
<td>2093.0</td>
<td>2217.5</td>
<td>2349.3</td>
<td>2489.0</td>
<td>2637.0</td>
<td>2793.8</td>
<td>2960.0</td>
<td>3136.0</td>
<td>3322.4</td>
<td>3520.0</td>
<td>3727.3</td>
<td>3951.1</td>
</tr>
<tr>
<td>9</td>
<td>4186.0</td>
<td>4434.9</td>
<td>4698.6</td>
<td>4978.0</td>
<td>5274.0</td>
<td>5587.7</td>
<td>5919.9</td>
<td>6271.9</td>
<td>6644.9</td>
<td>7040.0</td>
<td>7458.6</td>
<td>7902.1</td>
</tr>
<tr>
<td></td>
<td>8372.0</td>
<td>8869.8</td>
<td>9397.3</td>
<td>9956.1</td>
<td>10548</td>
<td>11175</td>
<td>11840</td>
<td>12544</td>
<td>13290</td>
<td>14080</td>
<td>14917</td>
<td>15804</td>
</tr>
</tbody>
</table>

Matrixbrute: \(4.0879 \text{ Hz} \rightarrow 16744 \text{ Hz}\)  
Subsequent 37: \(8.1768 \text{ Hz} \rightarrow 8372.0 \text{ Hz}\)  
Microbrute: \(32.703 \text{ Hz} \rightarrow 2093.0 \text{ Hz}\)

Midi numbers: from 0 (C-1) to 127 (G9)

In electronic music, pitch is often given by MIDI number \(m\), which is 69 for note A4 and increases by one for each equal tempered semitone, so this gives us these simple conversions between frequencies and MIDI numbers:

\[ m = 12 \log_2(f_m/440 \text{ Hz}) + 69 \]

\[ f_m = 440 \text{ Hz} \times 2^{(m-69)/12} \]

\[ \log_2(x) = \ln(x) / \ln(2) \]
27.3 Midi

There are 128 midi controllers. Controllers from 0 to 31 are 14-bit, the MSB is at addresses 0-31 and the LSB is at addresses 32-63.

27.4 USB/MIDI Adapters

The device name can be different, for example “UBB2.0-MIDI” or “USB A”.
Connect the “OUT” connector to “IN” at the synthesizer, and vice versa.
In rare cases I had transmission errors with the “USB2.0-MIDA” adapter. It seems better to use the direct USB cable to the MatrixBrute.

27.5 Synthesizer Abbreviations

CV = Control Voltage
DAW = Digital Audio Workstation
ENV = Envelope
LFO = Low Frequency Oscillator
MCC = Midi Control Center
PW = Pulse Width
VCA = Voltage Controlled Amplifier
VCO = Voltage Controlled Oscillator
VCF = Voltage Controlled Filter
### 27.6 Presets / Patches

<table>
<thead>
<tr>
<th>Matrixbrute</th>
<th>Subsequent 37</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initialize a preset with a basic sound</strong></td>
<td>Hold PANEL and then press PRESET.</td>
</tr>
<tr>
<td><strong>Load a preset</strong></td>
<td>Press PRESET if it's not already illuminated, then press one of the matrix buttons. Press the “BANK“ button and then one of the 16 “PRESET“ buttons to select a bank. Then press one of the 16 “PRESET“ buttons to load a preset.</td>
</tr>
</tbody>
</table>

### 27.7 Potentiometer behaviour

<table>
<thead>
<tr>
<th>Behaviour</th>
<th>Matrixbrute</th>
<th>Subsequent 37</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absolute mode, the value changes immediately when you turn the potentiometer.</td>
<td>In MIDI Control Center, device tab: Set 2pot mode2 to “Jump”. On MatrixBrute: Press PRESET+SEQ+MOD, then press G1.</td>
<td>Press “GLOBAL”, scroll down to “POT MODE”, then select “ABS”.</td>
<td>Recommended for creating new sounds.</td>
</tr>
<tr>
<td>Value changes relative to potentiometer adjustment. x Then you must first turn it to the other extreme position.</td>
<td>In MIDI Control Center, device tab: Set 2pot mode2 to “Scaled”. On MatrixBrute: Press PRESET+SEQ+MOD, then press G3. <strong>Drawback:</strong> If the potentiometer is at the extreme position, you cannot move it further.</td>
<td>Press “GLOBAL”, scroll down to “POT MODE”, then select “RLTV”.</td>
<td>Recommended for life performance.</td>
</tr>
</tbody>
</table>

### 27.8 MatrixBrute: Oscillator tuning

Wait at least 5 minutes after warm-up. Hold “Panel” and then press “Kbd Track”. You should see “Tun” in the display.
27.9 MatrixBrute power saving mode

In the system settings editor (PRESET + SEQ + MOD), row “P”, column “4” is “OFF” and column “5” is “ON”.

27.10 MatrixBrute AUDIO MOD Section

It’s unclear what type of modulation the “VCO1>VCO2” and “VCO1<VCO3>VCO2” potentiometers are doing. It appears to be a combination of amplitude, frequency and phase modulation.

27.11 MatrixBrute Control Voltages I/O

<table>
<thead>
<tr>
<th>Voltage</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCO1 Pitch</td>
<td>0 - 10V</td>
</tr>
<tr>
<td>VCO1 Ultra Saw</td>
<td>+/- 5V</td>
</tr>
<tr>
<td>VCO1 Pulse Width</td>
<td>+/- 5V</td>
</tr>
<tr>
<td>VCO1 Metalizer</td>
<td>+/- 5V</td>
</tr>
<tr>
<td>VCO2 Pitch</td>
<td>0 - 10V</td>
</tr>
<tr>
<td>VCO2 Ultra Saw</td>
<td>+/- 5V</td>
</tr>
<tr>
<td>VCO2 Pulse Width</td>
<td>+/- 5V</td>
</tr>
<tr>
<td>VCO2 Metalizer</td>
<td>+/- 5V</td>
</tr>
<tr>
<td>Steiner Cutoff</td>
<td>0 - 10V</td>
</tr>
<tr>
<td>Ladder Cutoff</td>
<td>0 - 10V</td>
</tr>
<tr>
<td>LFO1 Amount</td>
<td>0 - 10V</td>
</tr>
<tr>
<td>VCA</td>
<td>0 - 10V</td>
</tr>
</tbody>
</table>
27.12  MatrixBrute: Which Modules are unused?

<table>
<thead>
<tr>
<th>Module ...</th>
<th>is unused, if ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCO1</td>
<td>([VCO1&gt;VCO2] == 0 and ([MIXER_VCO1] == 0 or [MIXER_VCO1_Filter] == none) or ([VCO1_Saw] == 0 and [VCO1_Square] == 0 and [VCO1_Tri] == 0) ) or ([VCO1_Sub] == 0 and [VCO1_Saw] == 0 and [VCO1_Square] == 0 and [VCO1_Tri] == 0)</td>
</tr>
<tr>
<td>VCO2</td>
<td>([VCO_SYNC] == off and ([MIXER_VCO2] == 0 or [MIXER_VCO2_Filter] == none) or ([VCO2_Sub] == 0 and [VCO2_Saw] == 0 and [VCO2_Square] == 0 and [VCO2_Tri] == 0) )</td>
</tr>
<tr>
<td>VCO3-LFO3</td>
<td>([VCO1&lt;VCO3&gt;VCO2] == 0 and [VCF1&lt;VCO3&gt;VCF2] == 0 and (Row_G has no active elements) and ([MIXER_VCO3] == 0 or [MIXER_VCO3_Filter] == none) )</td>
</tr>
<tr>
<td>NOISE</td>
<td>([VCO1&lt;Noise&gt;VCF1] == 0 and ([MIXER_Noise] == 0 or [MIXER_Noise_Filter] == none) )</td>
</tr>
<tr>
<td>VCF1_STEINER</td>
<td>[Steiner_Out] == 0</td>
</tr>
<tr>
<td>VCF2_LADDER</td>
<td>[Ladder_Out] == 0</td>
</tr>
<tr>
<td>LFO1</td>
<td>Row_E has no active elements</td>
</tr>
<tr>
<td>LFO2</td>
<td>Row_F has no active elements</td>
</tr>
<tr>
<td>ENV1</td>
<td>([VCF1_STEINER_Env_Amt] == 0 and [VCF2_LADDER_Env_Amt] == 0 and (Row_A has no active elements) )</td>
</tr>
<tr>
<td>ENV2</td>
<td>(ENV2 is always used)</td>
</tr>
<tr>
<td>ENV3</td>
<td>Row_C has no active elements</td>
</tr>
<tr>
<td>ANALOG_EFFECTS</td>
<td>[Dry/Wet] == 0</td>
</tr>
</tbody>
</table>
This table lists the number of pictures and the video duration at 30fps, depending on interval and recording time.

<table>
<thead>
<tr>
<th>Interval</th>
<th>1h</th>
<th>2h</th>
<th>3h</th>
<th>4h</th>
<th>5h</th>
<th>6h</th>
<th>8h</th>
<th>12h</th>
<th>24h</th>
</tr>
</thead>
<tbody>
<tr>
<td>2s</td>
<td>1800</td>
<td>60s</td>
<td>3600</td>
<td>120s</td>
<td>5400</td>
<td>150s</td>
<td>7200</td>
<td>240s</td>
<td>9400</td>
</tr>
<tr>
<td>3s</td>
<td>1200</td>
<td>40s</td>
<td>2400</td>
<td>80s</td>
<td>3600</td>
<td>120s</td>
<td>4800</td>
<td>160s</td>
<td>6000</td>
</tr>
<tr>
<td>4s</td>
<td>900</td>
<td>30s</td>
<td>1800</td>
<td>60s</td>
<td>2700</td>
<td>90s</td>
<td>3600</td>
<td>120s</td>
<td>4500</td>
</tr>
<tr>
<td>5s</td>
<td>720</td>
<td>24s</td>
<td>1440</td>
<td>48s</td>
<td>2160</td>
<td>72s</td>
<td>2880</td>
<td>96s</td>
<td>3600</td>
</tr>
<tr>
<td>6s</td>
<td>600</td>
<td>20s</td>
<td>1200</td>
<td>40s</td>
<td>1800</td>
<td>60s</td>
<td>2400</td>
<td>80s</td>
<td>3000</td>
</tr>
<tr>
<td>8s</td>
<td>450</td>
<td>15s</td>
<td>900</td>
<td>30s</td>
<td>1350</td>
<td>45s</td>
<td>1800</td>
<td>60s</td>
<td>2350</td>
</tr>
<tr>
<td>10s</td>
<td>360</td>
<td>12s</td>
<td>720</td>
<td>24s</td>
<td>1080</td>
<td>36s</td>
<td>1440</td>
<td>48s</td>
<td>1800</td>
</tr>
<tr>
<td>12s</td>
<td>300</td>
<td>10s</td>
<td>600</td>
<td>20s</td>
<td>900</td>
<td>30s</td>
<td>1200</td>
<td>40s</td>
<td>1500</td>
</tr>
<tr>
<td>15s</td>
<td>240</td>
<td>8s</td>
<td>480</td>
<td>16s</td>
<td>720</td>
<td>24s</td>
<td>960</td>
<td>32s</td>
<td>1200</td>
</tr>
<tr>
<td>20s</td>
<td>180</td>
<td>6s</td>
<td>360</td>
<td>12s</td>
<td>540</td>
<td>18s</td>
<td>720</td>
<td>24s</td>
<td>900</td>
</tr>
<tr>
<td>24s</td>
<td>150</td>
<td>5s</td>
<td>300</td>
<td>10s</td>
<td>450</td>
<td>15s</td>
<td>600</td>
<td>20s</td>
<td>750</td>
</tr>
<tr>
<td>30s</td>
<td>120</td>
<td>4s</td>
<td>240</td>
<td>8s</td>
<td>360</td>
<td>12s</td>
<td>480</td>
<td>16s</td>
<td>600</td>
</tr>
<tr>
<td>40s</td>
<td>90</td>
<td>3s</td>
<td>180</td>
<td>6s</td>
<td>270</td>
<td>9s</td>
<td>360</td>
<td>12s</td>
<td>450</td>
</tr>
<tr>
<td>60s</td>
<td>60</td>
<td>2s</td>
<td>120</td>
<td>4s</td>
<td>180</td>
<td>6s</td>
<td>240</td>
<td>8s</td>
<td>300</td>
</tr>
<tr>
<td>120s</td>
<td>30</td>
<td>1s</td>
<td>60</td>
<td>2s</td>
<td>90</td>
<td>3s</td>
<td>120</td>
<td>4s</td>
<td>150</td>
</tr>
</tbody>
</table>
Zhiyun Crane 3S

See also: https://www.zhiyun-tech.com/crane3s/en

<table>
<thead>
<tr>
<th>Mode</th>
<th>Pan axis (left / right)</th>
<th>Tilt axis (up / down)</th>
<th>Roll axis (rotate image)</th>
<th>How to activate this mode</th>
<th>Joystick</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PF</td>
<td>following</td>
<td>stabilized</td>
<td>stabilized</td>
<td>Press MODE button</td>
<td>Pan + Tilt</td>
<td>Pan following mode (This is the default mode after startup) Note: The &quot;MODE&quot; button is only on the main body.</td>
</tr>
<tr>
<td>L</td>
<td>stabilized</td>
<td>stabilized</td>
<td>stabilized</td>
<td>Press MODE button</td>
<td>Pan + Tilt</td>
<td>Locking mode Note: The &quot;MODE&quot; button is only on the main body.</td>
</tr>
<tr>
<td>POV</td>
<td>following</td>
<td>following</td>
<td>following</td>
<td>Press POV button</td>
<td>--</td>
<td>Point of view mode, press FOV button again to return to previous mode</td>
</tr>
<tr>
<td>V</td>
<td>following</td>
<td>following</td>
<td>following</td>
<td>Press POV button twice</td>
<td>Pan</td>
<td>I don't understand this mode. If you can explain it, please let me know.</td>
</tr>
<tr>
<td>F</td>
<td>following</td>
<td>following</td>
<td>stabilized</td>
<td>Press F button</td>
<td>Roll</td>
<td>Following mode, press F button again to return to previous mode</td>
</tr>
<tr>
<td>GO</td>
<td>fast</td>
<td>fast</td>
<td>stabilized</td>
<td>Press GO button</td>
<td>--</td>
<td>Following mode, similar to &quot;F&quot; mode but faster following. Press GO button again to return to previous mode</td>
</tr>
</tbody>
</table>

Firmware upgrade: In the manual is described that you need "Zhiyun Gimbal Tools", but you won't find this on the website. Search for "Calibration Upgrade Tool" instead.

If the joystick and the buttons on the main body don't work: Move the "FN" switch to the left side.

Enter or exit standby mode: Long press the MODE button.

Note: GH5S must be set to USB Mode "PC(Tether)". Don't forget to set it back to "PC(Storage)" when you want to read out the SD card.

When using the gimbal, should the stabilization in the lens be activated or not?

I did make a test in PF mode with GH5S and Leica DG 12-60mm f/2.8-4.0 lens. The result is better if image stabilization in the lens is activated.
Balance adjustment table:

<table>
<thead>
<tr>
<th>Camera + Lens</th>
<th>Camera left/right</th>
<th>Tilt axis</th>
<th>Roll axis</th>
<th>Pan axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>GH5S + Leica DG 12-60mm f/2.8-4.0</td>
<td>22mm</td>
<td>48mm (maximum)</td>
<td>27mm</td>
<td>54mm</td>
</tr>
<tr>
<td>GH5S + Canon CN-E 24mm T1.5 L F with 0.64x SpeedBooster and variable ND filter and focus motor</td>
<td>22mm</td>
<td>27mm</td>
<td>8.5mm</td>
<td>43mm</td>
</tr>
</tbody>
</table>

Note: The extension arm is not required.

Note for Canon CN-E 24mm lens: Set the focus position to 0.7m before powering on the gimbal. Then the focus range is from 0.4m to infinite.

How to use the AB settings for limiting the focus range:

Press DOWN button, rotate to select "wheel", press RIGHT, rotate to select "Abpoint", press RIGHT.

Rotate the wheel to the first focus point, then select "A" and press RIGHT. Rotate the wheel to the second focus point, then select "B" and press RIGHT.

It doesn't care if the larger distance is A or B. Now the focus setting is limited between A and B. This setting is not permanently saved. You can delete it with the "clear" function in the same menu.
This is a small device that connects to a camera and controls the exposure time, aperture and ISO automatically, so that day-to-night or night-to-day timelapses are possible.

https://www.timelapseplus.com/

Important notes:

- Set the camera to manual (M) mode
- Use a native ISO setting (not Auto ISO)
- Save as RAW (not RAW + JPG)
- Manual focus (no autofocus)
- Disable image stabilization
- Check all parameters before using
- Don't rely on the internal battery, use an external powerbank
- Save the images in the camera, not in the Timelapse+ View
- The "Night Exposure" parameter describes how much darker the video shall become at night. Typical values are -0.5 to -0.75. Please note that the unit of this parameter isn't specified. These are not exposure compensation values! I did try -2 and the resulting video was much too dark in the night (about -11 exposure values).
This is a C# project I did program in 2016, before the "Timelapse+ View" became available. It's a program for controlling the exposure values of a Canon 6D camera for day-to-night or night-to-day timelapses. It should also work with other Canon cameras. The software calculates the altitude of the sun above or below the horizon, and based on this angle chooses a "light value". This "light value" is then translated into a combination of exposure time, aperture and ISO settings. All this is defined in the file "default.cfg" which is an ASCII text file that can be edited.

Screenshot:
default.cfg:

30                Interval in seconds
-6.02099          Geographic longitude, west = positive, decimal separator "."  Puimichel
43.98021          Geographic latitude, north = positive, decimal separator "."  Puimichel
83                Light_value for altitude -18° (deep night)
75                Light_value for altitude -15°
65                Light_value for altitude -12°
51                Light_value for altitude -9°
38                Light_value for altitude -6°
30                Light_value for altitude -3°
27                Light_value for altitude 0° (sunset)
27                Light_value for altitude +3°
28                Light_value for altitude +6°
28                Light_value for altitude +9°
28                Light_value for altitude +12°
28                Light_value for altitude +15°
28                Light_value for altitude +18° (daytime)
10 0 10 0    Light_value = exposure_code + aperture_code + ISO_code   1/4000s f/10 ISO100   10 = 0 + 10 + 0
11 1 10 0
12 2 10 0
13 3 10 0
14 4 10 0
15 5 10 0    Typical day: 1/1250s f/10 ISO100   15 = 5 + 10 + 0
16 6 10 0
17 7 10 0
18 8 10 0
19 9 10 0
20 10 10 0
21 11 10 0
22 12 10 0
23 13 10 0
24 14 10 0
25 15 10 0
26 16 10 0    1/100s f/10 ISO100   26 = 16 + 10 + 0
27 16 10 1
28 16 10 2
29 16 10 3
30 16 10 4
31 16 10 5
32 16 10 6
33 16 10 7
34 16 10 8
35 16 10 9
36 16 10 10    1/100s f/10 ISO1000   36 = 16 + 10 + 10
37 17 10 10
38 18 10 10
39 19 10 10
40 20 10 10
41 21 10 10
50  30  10  10  1/4s f/10 ISO1000  50 = 30 + 10 + 10
51  31  10  10
52  32  10  10
53  33  10  10
54  34  10  10
55  35  10  10
56  36  10  10
57  37  10  10
58  38  10  10
59  39  10  10
60  40  10  10  2.5s f/10 ISO1000  60 = 40 + 10 + 10
61  40  11  10
62  40  12  10
63  40  13  10
64  40  14  10
65  40  15  10
66  40  16  10
67  40  17  10
68  40  18  10  2.5s f/4 ISO1000  68 = 40 + 18 + 10
69  41  18  10
70  42  18  10
71  43  18  10
72  44  18  10
73  45  18  10
74  46  18  10
75  47  18  10
76  48  18  10
77  49  18  10
78  50  18  10  25s f/4 ISO1000  78 = 50 + 18 + 10
79  50  18  11
80  50  18  12
81  50  18  13
82  50  18  14
83  50  18  15  Typical night 25s f/4 ISO3200  83 = 50 + 18 + 15
84  50  18  16
85  50  18  17
86  50  18  18
87  50  18  19
88  50  18  20
89  50  18  21
90  50  18  22  25s f/4 ISO16000  90 = 50 + 18 + 22
-10.34433  Geographic longitude, west = positive, decimal separator ","  Herzberg
51.64833  Geographic latitude, north = positive, decimal separator ","  Herzberg
-10.52612  Geographic longitude, west = positive, decimal separator ","  St. Andreasberg
51.73166  Geographic latitude, north = positive, decimal separator ","  St. Andreasberg
-7.44621  Geographic longitude, west = positive, decimal separator ","  Gurnigel
46.73165  Geographic latitude, north = positive, decimal separator ","  Gurnigel
-6.02099  Geographic longitude, west = positive, decimal separator ","  Puimichel
43.98021  Geographic latitude, north = positive, decimal separator ","  Puimichel

Exposure codes:

<table>
<thead>
<tr>
<th>Number</th>
<th>Exposure Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1/4000s</td>
</tr>
<tr>
<td>1</td>
<td>1/3200s</td>
</tr>
<tr>
<td>2</td>
<td>2/2500s</td>
</tr>
<tr>
<td>3</td>
<td>3/2000s</td>
</tr>
<tr>
<td>4</td>
<td>4/1600s</td>
</tr>
<tr>
<td>5</td>
<td>5/1250s</td>
</tr>
<tr>
<td>6</td>
<td>6/1000s</td>
</tr>
<tr>
<td>7</td>
<td>7/800s</td>
</tr>
<tr>
<td>8</td>
<td>8/640s</td>
</tr>
<tr>
<td>9</td>
<td>9/500s</td>
</tr>
</tbody>
</table>

Aperture codes:

<table>
<thead>
<tr>
<th>Number</th>
<th>Aperture Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>f/32</td>
</tr>
<tr>
<td>1</td>
<td>f/29</td>
</tr>
<tr>
<td>2</td>
<td>f/25</td>
</tr>
<tr>
<td>3</td>
<td>f/22</td>
</tr>
<tr>
<td>4</td>
<td>f/20</td>
</tr>
<tr>
<td>5</td>
<td>f/18</td>
</tr>
<tr>
<td>6</td>
<td>f/16</td>
</tr>
<tr>
<td>7</td>
<td>f/14</td>
</tr>
<tr>
<td>8</td>
<td>f/13</td>
</tr>
<tr>
<td>9</td>
<td>f/11</td>
</tr>
<tr>
<td>10</td>
<td>f/10</td>
</tr>
<tr>
<td>11</td>
<td>f/9</td>
</tr>
<tr>
<td>12</td>
<td>f/8</td>
</tr>
<tr>
<td>13</td>
<td>f/7.1</td>
</tr>
<tr>
<td>14</td>
<td>f/6.3</td>
</tr>
<tr>
<td>15</td>
<td>f/5.6</td>
</tr>
<tr>
<td>16</td>
<td>f/5</td>
</tr>
<tr>
<td>17</td>
<td>f/4.5</td>
</tr>
<tr>
<td>18</td>
<td>f/4</td>
</tr>
<tr>
<td>19</td>
<td>f/3.5</td>
</tr>
<tr>
<td>20</td>
<td>f/3.2</td>
</tr>
<tr>
<td>21</td>
<td>f/2.8</td>
</tr>
<tr>
<td>22</td>
<td>f/2.5</td>
</tr>
<tr>
<td>23</td>
<td>f/2.2</td>
</tr>
<tr>
<td>24</td>
<td>f/2</td>
</tr>
<tr>
<td>25</td>
<td>f/1.8</td>
</tr>
<tr>
<td>26</td>
<td>f/1.6</td>
</tr>
<tr>
<td>27</td>
<td>f/1.4</td>
</tr>
<tr>
<td>28</td>
<td>f/1.2</td>
</tr>
<tr>
<td>29</td>
<td>f/1.1</td>
</tr>
<tr>
<td>30</td>
<td>f/1</td>
</tr>
</tbody>
</table>

ISO codes:

<table>
<thead>
<tr>
<th>Number</th>
<th>ISO Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ISO100</td>
</tr>
<tr>
<td>1</td>
<td>ISO125</td>
</tr>
<tr>
<td>2</td>
<td>ISO160</td>
</tr>
<tr>
<td>3</td>
<td>ISO200</td>
</tr>
<tr>
<td>4</td>
<td>ISO250</td>
</tr>
<tr>
<td>5</td>
<td>ISO320</td>
</tr>
<tr>
<td>6</td>
<td>ISO400</td>
</tr>
<tr>
<td>7</td>
<td>ISO500</td>
</tr>
<tr>
<td>8</td>
<td>ISO640</td>
</tr>
<tr>
<td>9</td>
<td>ISO800</td>
</tr>
<tr>
<td>10</td>
<td>ISO1000</td>
</tr>
<tr>
<td>11</td>
<td>ISO1250</td>
</tr>
<tr>
<td>12</td>
<td>ISO1600</td>
</tr>
<tr>
<td>13</td>
<td>ISO2000</td>
</tr>
<tr>
<td>14</td>
<td>ISO2500</td>
</tr>
<tr>
<td>15</td>
<td>ISO3200</td>
</tr>
<tr>
<td>16</td>
<td>ISO4000</td>
</tr>
<tr>
<td>17</td>
<td>ISO5000</td>
</tr>
<tr>
<td>18</td>
<td>ISO6400</td>
</tr>
<tr>
<td>19</td>
<td>ISO8000</td>
</tr>
<tr>
<td>20</td>
<td>ISO10000</td>
</tr>
<tr>
<td>21</td>
<td>ISO12800</td>
</tr>
<tr>
<td>22</td>
<td>ISO16000</td>
</tr>
<tr>
<td>23</td>
<td>ISO20000</td>
</tr>
<tr>
<td>24</td>
<td>ISO25600</td>
</tr>
</tbody>
</table>

MainForm.cs:

```csharp
using System;
using System.IO;
using System.Drawing;
using System.Windows.Forms;
using System.Collections.Generic;
using EDSDKLib;

// add "CameraClasses.cs" and "EDSDK.cs" to the project, these
// are in the folder "EDSDKLib" from Canon

namespace Timelapse_Control_V2
```

441
public partial class MainForm : Form {
    SDKHandler CameraHandler;
    List<Camera> CamList;

    int ErrCount;
    object ErrLock = new object();

    string[] lines;
    string[] words;
    double latitude, longitude;
    int interval;
    int[] lv = new int[32];
    int[] ex = new int[92];
    int[] ap = new int[92];
    int[] iso = new int[92];
    double altitude;
    Boolean bluedot = false;
    int bluedot_lv;
    double bluedot_altitude;
    Boolean taking_pictures = false;
    double offset = 0;
    int dark_limit = 90;
    int picture_counter = 0;

    public MainForm()
    {
        try
        {
            InitializeComponent();
            CameraHandler = new SDKHandler();
            CameraHandler.CameraAdded += new SDKHandler.CameraAddedHandler(SDK_CameraAdded);
            CameraHandler.CameraHasShutdown += SDK_CameraHasShutdown;
            RefreshCamera();
        }
        catch (DllNotFoundException) { ReportError("Canon DLLs not found!", true); }
        catch (Exception ex) { ReportError(ex.Message, true); }

        lines = File.ReadAllLines("default.cfg"); // read the config file
        words = lines[0].Split(' ');  // interval
        interval = System.Convert.ToInt32(words[0]);
        progressBar1.Maximum = interval;
        textBox5.Text = interval.ToString() + "s";
        //richTextBox1.AppendText(interval.ToString() + "\n");
        info.NumberDecimalSeparator = ".";
    }
}
info.NumberGroupSeparator = ",";

words = lines[1].Split(new char[] { ' ' }); // geographic longitude
longitude = System.Convert.ToDouble(words[0], info);
// richTextBox1.AppendText(longitude.ToString() + "\n");
textBox7.Text = longitude.ToString("F3") + "°";
longitude = longitude * Math.PI / 180;

words = lines[2].Split(new char[] { ' ' }); // geographic latitude
latitude = System.Convert.ToDouble(words[0], info);
// richTextBox1.AppendText(latitude.ToString() + "\n");
textBox6.Text = latitude.ToString("F3") + "°";
latitude = latitude * Math.PI / 180;

for (int i = 0; i < 13; i++)
{
    words = lines[i + 3].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries); // light value
    lv[i] = System.Convert.ToInt32(words[0]);
    // richTextBox1.AppendText(lv[i].ToString() + "\n");
}

for (int i = 0; i < 81; i++)
{
    words = lines[i + 16].Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);
    ex[i + 10] = System.Convert.ToInt32(words[1]); // exposure code
    ap[i + 10] = System.Convert.ToInt32(words[2]); // aperture code
    // richTextBox1.AppendText(ex[i + 10].ToString() + " " + ap[i + 10].ToString() + " " + iso[i + 10].ToString() + "\n");
}

private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
{
    try
    {
        if (CameraHandler != null) CameraHandler.Dispose();
    }
    catch (Exception ex) { ReportError(ex.Message, false); }
}

private void SDK_CameraAdded()
{
    try
    {
        RefreshCamera();
    }
    catch (Exception ex) { ReportError(ex.Message, false); }
}

private void SDK_CameraHasShutdown(object sender, EventArgs e)
{
    try
    {
        CloseSession();
    }
    catch (Exception ex) { ReportError(ex.Message, false); }
}
private void SessionButton_Click(object sender, EventArgs e)
{
    try
    {
        if (CameraHandler.CameraSessionOpen) CloseSession();
        else OpenSession();
    }
    catch (Exception ex) { ReportError(ex.Message, false); }
}

private void CloseSession()
{
    CameraHandler.CloseSession();
    SessionButton.Text = "Open Session";
    SessionLabel.Text = "No open session";
    RefreshCamera(); // Closing the session invalidates the current camera pointer
}

private void RefreshCamera()
{
    CameraListBox.Items.Clear();
    CamList = CameraHandler.GetCameraList();
    foreach (Camera cam in CamList) CameraListBox.Items.Add(cam.Info.szDeviceDescription);
    if (CameraHandler.CameraSessionOpen) CameraListBox.SelectedIndex = CamList.FindIndex(t => t.Ref == CameraHandler.MainCamera.Ref);
    else if (CamList.Count > 0) CameraListBox.SelectedIndex = 0;
}

private void OpenSession()
{
    if (CameraListBox.SelectedIndex >= 0)
    {
        CameraHandler.OpenSession(CamList[CameraListBox.SelectedIndex]);
        SessionButton.Text = "Close Session";
        string cameraname = CameraHandler.MainCamera.Info.szDeviceDescription;
        SessionLabel.Text = cameraname;
    }
}

private void ReportError(string message, bool lockdown)
{
    int errc;
    lock (ErrLock) { errc = ++ErrCount; }
    if (lockdown) EnableUI(false);
    if (errc < 4) MessageBox.Show(message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    else if (errc == 4) MessageBox.Show("Many errors happened!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    lock (ErrLock) { ErrCount--; }
}
private void EnableUI(bool enable)
{
    if (InvokeRequired) Invoke((Action)delegate { EnableUI(enable); });
}

private string GetExposureString(int lightvalue)
{
    switch (lightvalue)
    {
        case 0: return ("1/4000s");
        case 1: return ("1/3200s");
        case 2: return ("1/2500s");
        case 3: return ("1/2000s");
        case 4: return ("1/1600s");
        case 5: return ("1/1250s");
        case 6: return ("1/1000s");
        case 7: return ("1/800s");
        case 8: return ("1/640s");
        case 9: return ("1/500s");
        case 10: return ("1/400s");
        case 11: return ("1/320s");
        case 12: return ("1/250s");
        case 13: return ("1/200s");
        case 14: return ("1/160s");
        case 15: return ("1/125s");
        case 16: return ("1/100s");
        case 17: return ("1/80s");
        case 18: return ("1/60s");
        case 19: return ("1/50s");
        case 20: return ("1/40s");
        case 21: return ("1/30s");
        case 22: return ("1/25s");
        case 23: return ("1/20s");
        case 24: return ("1/15s");
        case 25: return ("1/13s");
        case 26: return ("1/10s");
        case 27: return ("1/8s");
        case 28: return ("1/6s");
        case 29: return ("1/5s");
        case 30: return ("1/4s");
        case 31: return ("0.3s");
        case 32: return ("0.4s");
        case 33: return ("0.5s");
        case 34: return ("0.6s");
        case 35: return ("0.8s");
        case 36: return ("1s");
        case 37: return ("1.3s");
        case 38: return ("1.6s");
        case 39: return ("2s");
    }
}
case 40: return ("2.5s");
case 41: return ("3.2s");
case 42: return ("4s");
case 43: return ("5s");
case 44: return ("6s");
case 45: return ("8s");
case 46: return ("10s");
case 47: return ("13s");
case 48: return ("15s");
case 49: return ("20s");
case 50: return ("25s");
case 51: return ("30s");
default: return ("");
}
}

private string GetApertureString(int lightvalue)
{
    switch (lightvalue)
    {
        case 3: return ("f/22");
        case 4: return ("f/20");
        case 5: return ("f/18");
        case 6: return ("f/16");
        case 7: return ("f/14");
        case 8: return ("f/13");
        case 9: return ("f/11");
        case 10: return ("f/10");
        case 11: return ("f/9");
        case 12: return ("f/8");
        case 13: return ("f/7.1");
        case 14: return ("f/6.3");
        case 15: return ("f/5.6");
        case 16: return ("f/5");
        case 17: return ("f/4.5");
        case 18: return ("f/4");
        case 19: return ("f/3.5");
        case 20: return ("f/3.2");
        case 21: return ("f/2.8");
        case 22: return ("f/2.5");
        case 23: return ("f/2.2");
        case 24: return ("f/2");
        case 25: return ("f/1.8");
        case 26: return ("f/1.6");
        case 27: return ("f/1.4");
        default: return ("");
    }
}

private string GetIsoString(int lightvalue)
```csharp
{ switch (lightvalue) {
    case 0: return ("ISO100");
    case 1: return ("ISO125");
    case 2: return ("ISO160");
    case 3: return ("ISO200");
    case 4: return ("ISO250");
    case 5: return ("ISO320");
    case 6: return ("ISO400");
    case 7: return ("ISO500");
    case 8: return ("ISO640");
    case 9: return ("ISO800");
    case 10: return ("ISO1000");
    case 11: return ("ISO1250");
    case 12: return ("ISO1600");
    case 13: return ("ISO2000");
    case 14: return ("ISO2500");
    case 15: return ("ISO3200");
    case 16: return ("ISO4000");
    case 17: return ("ISO5000");
    case 18: return ("ISO6400");
    case 19: return ("ISO8000");
    case 20: return ("ISO10000");
    case 21: return ("ISO12800");
    case 22: return ("ISO16000");
    case 23: return ("ISO20000");
    case 24: return ("ISO25600");
    default: return ("");
}

private int GetLightValue(double altitude)
{
    double y = 0;
    if (altitude < -15) y = lv[0] + Math.Round((altitude + 18) * (lv[1] - lv[0]) / 3);
    if (altitude >= 3 && altitude < 6)
```
if (altitude >= 6 && altitude < 9)
if (altitude >= 9 && altitude < 12)
if (altitude >= 12 && altitude < 15)
if (altitude >= 15)
y = lv[0] - (lv[0] - y) * (1.0 + offset);
if (y < 10) y = 10;
if (y > dark_limit) y = dark_limit;
return ((int)y);

private void timer1_Tick(object sender, EventArgs e)
{
  int lightvalue;
  uint AEMode = 0;
  int cam_ap = 0;
  int cam_ex = 0;
  int cam_iso = 0;
  if (checkBox1.Checked == false)
    altitude = getSunAltitude();
  if (altitude < -18) altitude = -18;
  if (altitude > 18) altitude = 18;
  bluedot = false;
  if (CameraHandler.CameraSessionOpen)
  {
    string AE = "";
    try
    {
      AEMode = CameraHandler.GetSetting(EDSDK.PropID_AEMode); // get and show the camera mode
      switch (AEMode)
      {
        case 0: AE = "P"; break;
        case 1: AE = "Tv"; break;
        case 2: AE = "Av"; break;
        case 3: AE = "M"; break;
        case 4: AE = "B"; break;
      }
      label4.Text = AE;
    }
    catch (Exception) { }
  }
}
if (AEMode != 3) // only if the mode is not "M"
{
    try
    {
        CameraHandler.PressShutterButton(EDSDK.EdsShutterButton.CameraCommand_ShutterButton_Halfway_NonAF);
        CameraHandler.PressShutterButton(EDSDK.EdsShutterButton.CameraCommand_ShutterButton_OFF);
    }
    catch (Exception) { }

    try
    {
        cam_ap = (int)CameraHandler.GetSetting((uint)EDSDK.PropID_Av);
        cam_ex = (int)CameraHandler.GetSetting((uint)EDSDK.PropID_Tv);
        cam_iso = (int)CameraHandler.GetSetting((uint)EDSDK.PropID_ISOSpeed);
    }
    catch (Exception) { }

    textBox8.Text = GetExposureString((153 - cam_ex) * 3 / 8);  // show the camera values
    textBox9.Text = GetApertureString((89 - cam_ap) * 3 / 8);
    textBox10.Text = GetIsoString((cam_iso - 71) * 3 / 8);

    if (lightvalue < 10) lightvalue = 10;
    if (lightvalue > 90) lightvalue = 90;
    bluedot = true;
    bluedot_lv = lightvalue;
    bluedot_altitude = altitude;
}
else
{
    int lv = GetLightValue(altitude);

    textBox8.Text = GetExposureString(ex[lv]);  // show the calculated values and send them to the camera
    uint ex_value = (uint)(152 - (8 * ex[lv] + 1) / 3);
    try
    {
        CameraHandler.SetSetting(EDSDK.PropID_Tv, ex_value);
    }
    catch (Exception) { }

    textBox9.Text = GetApertureString(ap[lv]);
    uint ap_value = (uint)(88 - (8 * ap[lv] + 1) / 3);
    try
    {
        CameraHandler.SetSetting(EDSDK.PropID_Av, ap_value);
    }
    catch (Exception) { }

    textBox10.Text = GetIsoString(iso[lv]);
    uint iso_value = (uint)(72 + (8 * iso[lv] + 1) / 3);
try {
    CameraHandler.SetSetting(EDSDK.PropID_ISOSpeed, iso_value);
} catch (Exception) { }

if (progressBar1.Value == progressBar1.Maximum && taking_pictures == true) // take a picture
{
    try {
        CameraHandler.PressShutterButton(EDSDK.EdsShutterButton.CameraCommand_ShutterButton_Completely_NonAF);
        CameraHandler.PressShutterButton(EDSDK.EdsShutterButton.CameraCommand_ShutterButton_OFF);
    } catch (Exception) { }
    picture_counter++; textBox11.Text = picture_counter.ToString();
}

if (progressBar1.Value == progressBar1.Maximum) // refresh the progress bar
    progressBar1.Value = 1;
else progressBar1.Value++;

pictureBox1.Refresh();

private double getSunAltitude() // get sun altitude in degrees
{
    double CR = Math.PI / 180;

    double xs, ys, zs, xp, yp, zp;
    double deklination, rektaszension;

    double ba, bb, bc;

    // TimeSpan ts = DateTime.UtcNow - new DateTime(2000, 1, 1, 11, 58, 56, DateTimeKind.Utc);  // should be correct?
    TimeSpan ts = DateTime.UtcNow - new DateTime(2000, 1, 1, 12, 01, 06, DateTimeKind.Utc);
    double jh = Convert.ToDouble(ts.TotalDays) / 36525; /* time in centuries */

    double mean_longitude = CR * 100.466449 + jh * CR * 35999.3728519;
    double inclination = CR * 0.00000000 + jh * CR * 0.0130546;
    double ascending_node = CR * 174.873174 + jh * CR * -0.2410908;
    double longitude_perihel = CR * 102.937348 + jh * CR * 0.3225557;
    double excentricity = 0.01670862 + jh * -0.000042037;

    ba = mean_longitude - longitude_perihel;
    bb = ba;
do
{
    bc = ba;
    ba = bb + excentricity * Math.Sin(ba);
}
while (Math.Abs(ba - bc) > 1e-6);
if (Math.Abs(ba) == 3.141592654)
    bc = ba;
else
    bc = 2.0 * Math.Atan(Math.Tan(ba / 2.0) * Math.Sqrt((1.0 + excentricity) / (1.0 - excentricity)));
ba = bc + longitude_perihel - ascending_node;
xp = Math.Cos(ba) * Math.Cos(ascending_node) - Math.Sin(ba) * Math.Sin(ascending_node) * Math.Cos(inclination);
yp = Math.Cos(ba) * Math.Sin(ascending_node) + Math.Sin(ba) * Math.Cos(ascending_node) * Math.Cos(inclination);
zp = Math.Sin(ba) * Math.Sin(inclination);
xs = -xp;
ys = -yp * 0.917469448 + zp * 0.397806184;
zs = -zp * 0.917469448 - yp * 0.397806184;
deklination = Math.Atan(zs / Math.Sqrt(xs * xs + ys * ys));
rektaszension = Math.Atan2(ys, xs);
if (rektaszension < 0) rektaszension += 2 * Math.PI;

double sternzeit_greenwich = CR * (280.46061837 + 360.98564736629 * jh * 36525 + 0.000387933 * jh * jh - jh * jh * jh / 38710000);
double sternzeit_greenwich = CR + CR;
double Stundenwinkel = sternzeit_greenwich - longitude - rektaszension;
altitude /= CR;
azimut /= CR;
textBox1.Text = (altitude).ToString("F3") + "°";
textBox2.Text = (180 + azimut).ToString("F3") + "°";
textBox3.Text = (rektaszension / Math.PI * 12).ToString("F3") + "h";
textBox4.Text = (deklination / CR).ToString("F3") + "°";
return (altitude);
}

private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen pen1 = new Pen(Color.Gray);
    pen1.Width = 1;
    Pen pen2 = new Pen(Color.Yellow);
}
pen2.Width = 1;
Pen pen3 = new Pen(Color.OrangeRed);
pen3.Width = 1;
Pen pen4 = new Pen(Color.LightGreen);
pen4.Width = 1;
Pen pen5 = new Pen(Color.White);
pen5.Width = 1;
Pen pen6 = new Pen(Color.CornflowerBlue);
pen6.Width = 4;
Pen pen7 = new Pen(Color.White);
pen7.Width = 4;
Pen pen8 = new Pen(Color.OrangeRed);
pen8.Width = 3;
Pen pen9 = new Pen(Color.Yellow);
pen9.Width = 4;
Pen pen10 = new Pen(Color.OrangeRed);
pen10.Width = 4;
Pen pen11 = new Pen(Color.LightGreen);
pen11.Width = 4;
Font font1 = new Font("Arial", 8, FontStyle.Bold);

for (int lv = 10; lv < 91; lv++) // horizontal lines with annotation
{
    g.DrawLine(pen1, 10, 5 * lv - 40, 298, 5 * lv - 40);
g.DrawLine(pen1, 535, 5 * lv - 40, 761, 5 * lv - 40);
    if (lv % 5 == 0)
        g.DrawString(lv.ToString(), font1, new SolidBrush(Color.Gray), 317, 5 * lv - 47);
}
for (int h = -18; h < 19; h += 3) // vertical lines with annotation
{
    g.DrawLine(pen1, 154 + 8 * h, 10, 154 + 8 * h, 410);
g.DrawString(h.ToString() + "°", font1, new SolidBrush(Color.Gray), 145 + 8 * h, 412);
    g.DrawString("Total Darkness", font1, new SolidBrush(Color.Gray), 0, 425);
g.DrawString("Sun Altitude", font1, new SolidBrush(Color.Gray), 119, 425);
g.DrawString("Full Daylight", font1, new SolidBrush(Color.Gray), 238, 425);
}
for (int x = 0; x < 52; x += 3) // vertical lines (right side) with annotation
{
    g.DrawLine(pen1, 353 + 8 * x, 10, 353 + 8 * x, 410);
}
g.DrawString("1/4000s", font1, new SolidBrush(Color.Yellow), 331, 412);
g.DrawString("1/1000s", font1, new SolidBrush(Color.Yellow), 379, 412);
g.DrawString("1/250s", font1, new SolidBrush(Color.Yellow), 431, 412);
g.DrawString("1/60s", font1, new SolidBrush(Color.Yellow), 481, 412);
g.DrawString("1/15s", font1, new SolidBrush(Color.Yellow), 528, 412);
g.DrawString("1/4s", font1, new SolidBrush(Color.Yellow), 578, 412);
g.DrawString("1s", font1, new SolidBrush(Color.Yellow), 633, 412);
private void checkBox1_CheckedChanged(object sender, EventArgs e) // simulation of the altitude
{
}
if (checkBox1.Checked)
{
    altitude = 0;
    button2.Enabled = true;
    button3.Enabled = true;
}
else
{
    altitude = getSunAltitude();
    button2.Enabled = false;
    button3.Enabled = false;
}
textBox1.Text = (altitude).ToString("F3") + "°";

private void button2_Click(object sender, EventArgs e)
{
    altitude -= 1.0;
    if (altitude < -18) altitude = -18;
    textBox1.Text = (altitude).ToString("F3") + "°";
    pictureBox1.Refresh();
}

private void button3_Click(object sender, EventArgs e)
{
    altitude += 1.0;
    if (altitude > 18) altitude = 18;
    textBox1.Text = (altitude).ToString("F3") + "°";
    pictureBox1.Refresh();
}

private void button1_Click(object sender, EventArgs e)
{
    if (taking_pictures == true)
    {
        taking_pictures = false;
        button1.Text = "Start Taking Pictures";
    }
    else
    {
        if (CameraHandler.CameraSessionOpen)
        {
            taking_pictures = true;
            button1.BackColor = Color.Tomato;
            button1.Text = "Stop Taking Pictures";
        }
    }
}
private void button4_Click(object sender, EventArgs e)
{
    offset += 0.01;
    pictureBox1.Refresh();
}

private void button5_Click(object sender, EventArgs e)
{
    offset -= 0.01;
    pictureBox1.Refresh();
}

private void button6_Click(object sender, EventArgs e)
{
    dark_limit -= 1;
    if (dark_limit < 10) dark_limit = 10;
    pictureBox1.Refresh();
}

private void button7_Click(object sender, EventArgs e)
{
    dark_limit += 1;
    if (dark_limit > 90) dark_limit = 90;
    pictureBox1.Refresh();
}

namespace Timelapse_Control_V2
{
    partial class MainForm
    {
        /// <summary>MainForm</summary>
        /// Erforderliche Designervariable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>MainForm</summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        /// <param name="disposing">True, wenn verwaltete Ressourcen gelöscht werden sollen; andernfalls False.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
            }
        }
    }
}
{  
    components.Dispose();
}  
base.Dispose(disposing);

#endregion  

/Vom Windows Form-Designer generierter Code
/// <summary>
/// Erforderliche Methode für die Designerunterstützung.
/// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
/// </summary>
private void InitializeComponent() {
    this.components = new System.ComponentModel.Container();
    this.CameraListBox = new System.Windows.Forms.ListBox();
    this.SessionLabel = new System.Windows.Forms.Label();
    this.STBothButton = new System.Windows.Forms.RadioButton();
    this.STComputerButton = new System.Windows.Forms.RadioButton();
    this.timer1 = new System.Windows.Forms.Timer(this.components);
    this.label4 = new System.Windows.Forms.Label();
    this.textBox1 = new System.Windows.Forms.TextBox();
    this.textBox2 = new System.Windows.Forms.TextBox();
    this.label5 = new System.Windows.Forms.Label();
    this.label6 = new System.Windows.Forms.Label();
    this.textBox3 = new System.Windows.Forms.TextBox();
    this.textBox4 = new System.Windows.Forms.TextBox();
    this.label7 = new System.Windows.Forms.Label();
    this.label8 = new System.Windows.Forms.Label();
    this.pictureBox1 = new System.Windows.Forms.PictureBox();
    this.textBox5 = new System.Windows.Forms.TextBox();
    this.textBox6 = new System.Windows.Forms.TextBox();
    this.textBox7 = new System.Windows.Forms.TextBox();
    this.label9 = new System.Windows.Forms.Label();
    this.label10 = new System.Windows.Forms.Label();
    this.label11 = new System.Windows.Forms.Label();
    this.checkBox1 = new System.Windows.Forms.CheckBox();
    this.button2 = new System.Windows.Forms.Button();
    this.button3 = new System.Windows.Forms.Button();
    this.progressBar1 = new System.Windows.Forms.ProgressBar();
    this.button1 = new System.Windows.Forms.Button();
    this.label1 = new System.Windows.Forms.Label();
}
this.button4 = new System.Windows.Forms.Button();
this.button5 = new System.Windows.Forms.Button();
this.label2 = new System.Windows.Forms.Label();
this.button6 = new System.Windows.Forms.Button();
this.button7 = new System.Windows.Forms.Button();
this.label3 = new System.Windows.Forms.Label();
this.textBox11 = new System.Windows.Forms.TextBox();
((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
this.SuspendLayout();
//
// CameraListBox
//
this.CameraListBox.FormattingEnabled = true;
this.CameraListBox.ItemHeight = 14;
this.CameraListBox.Location = new System.Drawing.Point(6, 26);
this.CameraListBox.Name = "CameraListBox";
this.CameraListBox.Size = new System.Drawing.Size(107, 46);
this.CameraListBox.TabIndex = 6;
//
// SessionButton
//
this.SessionButton.Location = new System.Drawing.Point(6, 79);
this.SessionButton.Name = "SessionButton";
this.SessionButton.Size = new System.Drawing.Size(97, 27);
this.SessionButton.TabIndex = 7;
this.SessionButton.Text = "Open Session";
this.SessionButton.UseVisualStyleBackColor = true;
this.SessionButton.Click += new System.EventHandler(this.SessionButton_Click);
//
// SessionLabel
//
this.SessionLabel.AutoSize = true;
this.SessionLabel.Location = new System.Drawing.Point(6, 9);
this.SessionLabel.Name = "SessionLabel";
this.SessionLabel.Size = new System.Drawing.Size(100, 14);
this.SessionLabel.TabIndex = 8;
this.SessionLabel.Text = "No open session";
//
// STBothButton
//
this.STBothButton.Location = new System.Drawing.Point(0, 0);
this.STBothButton.Name = "STBothButton";
this.STBothButton.Size = new System.Drawing.Size(104, 24);
this.STBothButton.TabIndex = 0;

// STComputerButton
//
this.STComputerButton.Location = new System.Drawing.Point(0, 0);
this.STComputerButton.Name = "STComputerButton";
this.STComputerButton.Size = new System.Drawing.Size(104, 24);
this.STComputerButton.TabIndex = 0;

// STCameraButton
//
this.STCameraButton.Location = new System.Drawing.Point(0, 0);
this.STCameraButton.Name = "STCameraButton";
this.STCameraButton.Size = new System.Drawing.Size(104, 24);
this.STCameraButton.TabIndex = 0;

// timer1
//
this.timer1.Enabled = true;
this.timer1.Interval = 1000;
this.timer1.Tick += new System.EventHandler(this.timer1_Tick);

// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(97, 113);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(31, 29);
this.label4.TabIndex = 13;
this.label4.Text = "--";

// textBox1
//
this.textBox1.Location = new System.Drawing.Point(101, 296);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(58, 20);
this.textBox1.TabIndex = 15;

// textBox2
//
this.textBox2.Location = new System.Drawing.Point(101, 322);
this.textBox2.Name = "textBox2";
this.textBox2.Size = new System.Drawing.Size(58, 20);
this.textBox2.TabIndex = 18;
this.textBox2.Size = new System.Drawing.Size(58, 20);
this.textBox2.TabIndex = 16;

// label5
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(6, 299);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(74, 14);
this.label5.TabIndex = 17;
this.label5.Text = "Sun Altitude";

// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(6, 325);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(70, 14);
this.label6.TabIndex = 18;
this.label6.Text = "Sun Azimuth";

// textBox3
//
this.textBox3.Location = new System.Drawing.Point(101, 348);
this.textBox3.Name = "textBox3";
this.textBox3.Size = new System.Drawing.Size(58, 20);
this.textBox3.TabIndex = 19;

// textBox4
//
this.textBox4.Location = new System.Drawing.Point(101, 374);
this.textBox4.Name = "textBox4";
this.textBox4.Size = new System.Drawing.Size(58, 20);
this.textBox4.TabIndex = 20;

// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(6, 351);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(52, 14);
this.label7.TabIndex = 21;
this.label7.Text = "Sun R.A.";
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(6, 377);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(52, 14);
this.label8.TabIndex = 22;
this.label8.Text = "Sun DEC";

// pictureBox1
//
this.pictureBox1.Location = new System.Drawing.Point(178, 9);
this.pictureBox1.Name = "pictureBox1";
this.pictureBox1.Size = new System.Drawing.Size(768, 460);
this.pictureBox1.TabStop = false;
this.pictureBox1.Paint += new System.Windows.Forms.PaintEventHandler(this.pictureBox1_Paint);

// textBox5
//
this.textBox5.Location = new System.Drawing.Point(101, 218);
this.textBox5.Name = "textBox5";
this.textBox5.Size = new System.Drawing.Size(58, 20);
this.textBox5.TabIndex = 25;

// textBox6
//
this.textBox6.Location = new System.Drawing.Point(101, 244);
this.textBox6.Name = "textBox6";
this.textBox6.Size = new System.Drawing.Size(58, 20);
this.textBox6.TabIndex = 26;

// textBox7
//
this.textBox7.Location = new System.Drawing.Point(101, 270);
this.textBox7.Name = "textBox7";
this.textBox7.Size = new System.Drawing.Size(58, 20);
this.textBox7.TabIndex = 27;
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(6, 221);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(48, 14);
this.label9.TabIndex = 28;
this.label9.Text = "Interval";
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(6, 247);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(52, 14);
this.label10.TabIndex = 29;
this.label10.Text = "Latitude";
//
// label11
//
this.label11.AutoSize = true;
this.label11.Location = new System.Drawing.Point(6, 273);
this.label11.Name = "label11";
this.label11.Size = new System.Drawing.Size(63, 14);
this.label11.TabIndex = 30;
this.label11.Text = "Longitude";
//
// textBox8
//
this.textBox8.Location = new System.Drawing.Point(101, 400);
this.textBox8.Name = "textBox8";
this.textBox8.Size = new System.Drawing.Size(58, 20);
this.textBox8.TabIndex = 31;
//
// textBox9
//
this.textBox9.Location = new System.Drawing.Point(101, 426);
this.textBox9.Name = "textBox9";
this.textBox9.Size = new System.Drawing.Size(58, 20);
this.textBox9.TabIndex = 32;
//
// textBox10
//
this.textBox10.Location = new System.Drawing.Point(101, 452);
this.textBox10.Name = "textBox10";
this.textBox10.Size = new System.Drawing.Size(58, 20);
this.textBox10.TabIndex = 33;

// label12
this.label12.AutoSize = true;
this.label12.Location = new System.Drawing.Point(6, 403);
this.label12.Name = "label12";
this.label12.Size = new System.Drawing.Size(60, 14);
this.label12.TabIndex = 34;
this.label12.Text = "Exp. Time";

// label13
this.label13.AutoSize = true;
this.label13.Location = new System.Drawing.Point(6, 429);
this.label13.Name = "label13";
this.label13.Size = new System.Drawing.Size(57, 14);
this.label13.TabIndex = 35;
this.label13.Text = "Aperture";

// label14
this.label14.AutoSize = true;
this.label14.Location = new System.Drawing.Point(6, 455);
this.label14.Name = "label14";
this.label14.Size = new System.Drawing.Size(58, 14);
this.label14.TabIndex = 36;
this.label14.Text = "ISO Value";

// checkBox1
this.checkBox1.AutoSize = true;
this.checkBox1.Location = new System.Drawing.Point(9, 159);
this.checkBox1.Name = "checkBox1";
this.checkBox1.Size = new System.Drawing.Size(74, 32);
this.checkBox1.TabIndex = 38;
this.checkBox1.Text = "Simulate\r\nAltitude";

462
this.checkBox1.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
this.checkBox1.UseVisualStyleBackColor = true;
this.checkBox1.CheckedChanged += new System.EventHandler(this.checkBox1_CheckedChanged);

// // button2
//
this.button2.Enabled = false;
this.button2.Location = new System.Drawing.Point(97, 159);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(30, 27);
this.button2.TabIndex = 39;
this.button2.Text = "<";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);

// // button3
//
this.button3.Enabled = false;
this.button3.Location = new System.Drawing.Point(133, 159);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(30, 27);
this.button3.TabIndex = 40;
this.button3.Text = ">";
this.button3.UseVisualStyleBackColor = true;
this.button3.Click += new System.EventHandler(this.button3_Click);

// // progressBar1
//
this.progressBar1.Location = new System.Drawing.Point(6, 478);
this.progressBar1.MarqueeAnimationSpeed = 0;
this.progressBar1.Name = "progressBar1";
this.progressBar1.Size = new System.Drawing.Size(940, 19);
this.progressBar1.Step = 1;
this.progressBar1.TabIndex = 41;

// // button1
//
this.button1.Location = new System.Drawing.Point(6, 112);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(85, 41);
this.button1.TabIndex = 43;
this.button1.Text = "Start Taking Pictures";
this.button1.UseVisualStyleBackColor = false;
this.button1.Click += new System.EventHandler(this.button1_Click);

//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(130, 9);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(41, 14);
this.label1.TabIndex = 44;
this.label1.Text = "Offset";
//
// button4
//
this.button4.Location = new System.Drawing.Point(141, 26);
this.button4.Name = "button4";
this.button4.Size = new System.Drawing.Size(22, 22);
this.button4.TabIndex = 45;
this.button4.Text = "^";
this.button4.UseVisualStyleBackColor = true;
this.button4.Click += new System.EventHandler(this.button4_Click);
//
// button5
//
this.button5.Location = new System.Drawing.Point(141, 54);
this.button5.Name = "button5";
this.button5.Size = new System.Drawing.Size(22, 22);
this.button5.TabIndex = 46;
this.button5.Text = "v";
this.button5.UseVisualStyleBackColor = true;
this.button5.Click += new System.EventHandler(this.button5_Click);
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(109, 85);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(63, 14);
this.label2.TabIndex = 47;
this.label2.Text = "Dark Limit";
//
// button6
//
this.button6.Location = new System.Drawing.Point(141, 103);
this.button6.Name = "button6";
this.button6.Size = new System.Drawing.Size(22, 22);
this.button6.TabIndex = 48;
this.button6.Text = "^";
this.button6.UseVisualStyleBackColor = true;
this.button6.Click += new System.EventHandler(this.button6_Click);

// button7

this.button7.Location = new System.Drawing.Point(141, 131);
this.button7.Name = "button7";
this.button7.Size = new System.Drawing.Size(22, 22);
this.button7.Text = "v";
this.button7.UseVisualStyleBackColor = true;
this.button7.Click += new System.EventHandler(this.button7_Click);

// label3

this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(6, 195);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(94, 14);
this.label3.TabIndex = 50;
this.label3.Text = "Picture Counter";

// textBox11

this.textBox11.Location = new System.Drawing.Point(101, 192);
this.textBox11.Name = "textBox11";
this.textBox11.Size = new System.Drawing.Size(58, 20);
this.textBox11.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

// MainForm

this.AutoScaleDimensions = new System.Drawing.SizeF(7F, 14F);
this.ClientSize = new System.Drawing.Size(953, 504);
this.Controls.Add(this.textBox11);
this.Controls.Add(this.label3);
this.Controls.Add(this.button7);
this.Controls.Add(this.button6);
this.Controls.Add(this.label2);
this.Controls.Add(this.button5);
this.Controls.Add(this.button4);
this.Controls.Add(this.button3);
this.Controls.Add(this.label1);

this.Controls.Add(this.button1);
this.Controls.Add(this.progressBar1);
this.Controls.Add(this.button3);
this.Controls.Add(this.button2);
this.Controls.Add(this.checkBox1);
this.Controls.Add(this.textBox10);
this.Controls.Add(this.textBox9);
this.Controls.Add(this.textBox8);
this.Controls.Add(this.label14);
this.Controls.Add(this.label13);
this.Controls.Add(this.label12);
this.Controls.Add(this.textBox11);
this.Controls.Add(this.pictureBox1);
this.Controls.Add(this.label11);
this.Controls.Add(this.label10);
this.Controls.Add(this.label9);
this.Controls.Add(this.textBox7);
this.Controls.Add(this.textBox6);
this.Controls.Add(this.textBox5);
this.Controls.Add(this.label8);
this.Controls.Add(this.label7);
this.Controls.Add(this.textBox4);
this.Controls.Add(this.textBox3);
this.Controls.Add(this.label6);
this.Controls.Add(this.label5);
this.Controls.Add(this.textBox2);
this.Controls.Add(this.textBox1);
this.Controls.Add(this.label4);
this.Controls.Add(this.SessionButton);
this.Controls.Add(this.CameraListBox);
this.MaximizeBox = false;
this.MinimizeBox = false;
this.MinimumSize = new System.Drawing.Size(683, 320);
this.Name = "MainForm";
this.Text = "Micha\'s Timelapse Control V2.0";
this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.MainForm_FormClosing);
((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();
}
#endregion
private System.Windows.Forms.ListBox CameraListBox;
private System.Windows.Forms.Label SessionLabel;
private System.Windows.Forms.Timer timer1;
private System.Windows.Forms.Label label14;
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.TextBox textBox2;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.TextBox textBox3;
private System.Windows.Forms.TextBox textBox4;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.PictureBox pictureBox1;
private System.Windows.Forms.TextBox textBox5;
private System.Windows.Forms.TextBox textBox6;
private System.Windows.Forms.TextBox textBox7;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.Label label11;
private System.Windows.Forms.TextBox textBox8;
private System.Windows.Forms.TextBox textBox9;
private System.Windows.Forms.TextBox textBox10;
private System.Windows.Forms.Label label12;
private System.Windows.Forms.Label label13;
private System.Windows.Forms.Label label14;
private System.Windows.Forms.TextBox textBox11;
}
32  Guide 9.1

This chapter may be a little bit off-topic in this document, because Guide 9.1 is an astronomy program. But I didn't want to create a new document for my notes about it.

Website: https://www.projectpluto.com/

32.1  Install Guide 9.1

• Insert the Guide 9.0 DVD and open the folder, then run "setup.exe". This will install Guide 9.0 very fast, but most of the data is still on the DVD. Which means it does only work if you let the DVD in the drive.

• If you have enough space on the harddisk, it's recommended to install all on the harddisk. Run Guide 9.0 and click on Extras / Install_on_hard_drive. It's best if you select all, except those languages that you don't need.

• It's highly recommended to install the upgrade to Guide 9.1, which is available here: https://www.projectpluto.com/ This upgrade is required for communication with a telescope over the serial port, and also for downloading the latest comet orbit data.

32.2  Control a LX200-compatible telescope

• If your computer doesn't have a RS232 port, then use a USB / RS232 adapter. Plug this adapter into a free USB port and find out which COM number was assigned to the adapter, e.g. COM10. Use always the same USB port. Otherwise the COM number will change.

• In Guide 9.1 click on Settings / Scope_Control. here you choose the COM number and as telescope type you use "LX200". Then click on "OK".

• Now there is a new menu "Scope Pad". When you click on it, a small window opens. Here you can control the telescope. It's described in the FS2 manual.

• USB adapters don't work with Guide 9.0. You must install the Guide 9.1 upgrade.
32.3 Add new comets to Guide 9.1

The "Add MPC Comets / Asteroids" function does no longer work. You can use this workaround:


and save this file in your Guide folder as soft02cm.txt (this is used for Guide) and also as comets.dat or cometg.dat (this is used for Charon, use the filename that already exists).

The broken "Add MPC Comets / Asteroids" function in Guide9.1 can be repaired if you copy and paste the following content to the "add_mpc.hee" file. In german installations the filename may be "add_mpc.hed". This doesn't work with Guide 9.0, the upgrade to Guide 9.1 is required. (Thanks to Larry Wood who posted this in the Guide user group, September 18, 2019)

<table>
<thead>
<tr>
<th>The Minor Planet Center (MPC)</th>
<th><a href="https://www.minorplanetcenter.net/cfa/ps/mpc.html">https://www.minorplanetcenter.net/cfa/ps/mpc.html</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>and the IMCCE</td>
<td><a href="http://www.imcce.fr/fr">http://www.imcce.fr/fr</a></td>
</tr>
<tr>
<td>provide orbital elements for comets. Guide updates its list of comets using both sources; MPC gives currently-observable comets, IMCCE all comets since about 1995. (Data for historical comets is already built into the Guide DVD.) You can click on the following to download some of these files, getting orbital data for newly-found objects and improving orbits for already known objects. About 600 KBytes will be downloaded.</td>
<td></td>
</tr>
</tbody>
</table>

^Click to download updated comet data and add it to Guide^  
[dhttp://astro.vanbuitenen.nl/cometelements?format=guide soft02cm.txt]  
[dhttps://www.projectpluto.com/eltdat.txt eltdat.txt;a2789]

Guide can also import other orbital elements if they're provided in the "eight-line format", or the "one-line format" used for Daily Orbit Updates. You wouldn't normally do this, but if you have generated an orbit using Find_Orb, for example, you could import the resulting file of orbital elements using the following command.

^Add MPC asteroids/comets//!2052^  

Please note that the long line in the middle must be written in one line and there is a space character between "guide" and "soft02cm".
32.4 Add ephemerides to Guide 9.1

The path of those comets or asteroids which have a close encounter with other objects (e.g. planets) can’t be described by orbital elements for a longer time. If you want to add the ephemeride of such an object point-wise into Guide 9.1, follow these instructions:

Go to this MPC website: http://www.minorplanetcenter.net/iau/MPEph/MPEph.html

and write the name of the object in the large white field (e.g. 2012DA14). Then fill in some more fields (use your own data, of course):

Ephemeris start date: e.g. 2013 02 15 19:00
Number of dates to output: e.g. 400
Ephemeris interval: e.g. 1 minute
Longitude: e.g. 10.3454
Latitude: e.g. 51.3829
Altitude: e.g. 257

Display R.A./Decl. positions in: full sexagesimal

Tick the box "Suppress output if sun above local horizon" if that makes sense for your object.

Then click on "Get Ephemerides/HTML page". Now copy and paste the data lines (without the header) to an editor. It should look like this:

<table>
<thead>
<tr>
<th>Date</th>
<th>Right Ascension</th>
<th>Declination</th>
<th>RA Error</th>
<th>Dec Error</th>
<th>Longitude</th>
<th>Latitude</th>
<th>Altitude</th>
<th>RA Offset</th>
<th>Dec Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013 02 15</td>
<td>19 00 00</td>
<td>12 01 19.4</td>
<td>0.00026</td>
<td>0.988</td>
<td>127.0</td>
<td>52.9</td>
<td>8.4</td>
<td>2279.58</td>
<td>0.31 147 +36</td>
</tr>
<tr>
<td>2013 02 15</td>
<td>19 01 00</td>
<td>12 01 30.2</td>
<td>0.00026</td>
<td>0.988</td>
<td>127.5</td>
<td>52.5</td>
<td>8.3</td>
<td>2300.72</td>
<td>0.31 147 +35</td>
</tr>
<tr>
<td>2013 02 15</td>
<td>19 02 00</td>
<td>12 01 41.1</td>
<td>0.00025</td>
<td>0.988</td>
<td>128.0</td>
<td>52.0</td>
<td>8.3</td>
<td>2321.73</td>
<td>0.31 147 +35</td>
</tr>
<tr>
<td>2013 02 15</td>
<td>19 03 00</td>
<td>12 01 51.9</td>
<td>0.00025</td>
<td>0.988</td>
<td>128.5</td>
<td>51.5</td>
<td>8.3</td>
<td>2342.59</td>
<td>0.31 148 +35</td>
</tr>
<tr>
<td>2013 02 15</td>
<td>19 04 00</td>
<td>12 02 02.7</td>
<td>0.00025</td>
<td>0.988</td>
<td>128.9</td>
<td>51.1</td>
<td>8.3</td>
<td>2363.24</td>
<td>0.31 148 +35</td>
</tr>
</tbody>
</table>

and so on...

Save this file as "2012DA14.dat" to your Guide folder (e.g. C:/GUIDE9).

It's absolutely required that all data are in the correct columns, as shown above.
Now create another file "2012DA14.tdf" and save it in the same folder. This is the content:

```
file 2012DA14.dat
title Asteroid 2012DA14
RA H   19   2
RA M   22   2
RA S   25   4
de d   30   3
de m   34   2
de s   37   2
mag    70   4
text   12   4
pref 2012DA14
epoch 2000
type scl:e0,0,30;   #green circle, 30 pixels diameter
shown 1
end
```

That's all. Start Guide and the positions will be shown.

### 32.5 Update the position of Jupiter's great red spot

The longitude of Jupiter's great red spot must be updated from time to time. To do this, open the file "grs_long.txt" from the Guide folder with an editor. Then insert a new line near the top, for example:

```
2019  6  1 311  (user)
```

In this example the longitude is 311° for date 2019 June 1th.

Save the file with the same filename.
32.6 Add a user-defined horizon

Either measure the horizon heights with a azimuthal telescope, or make a 180° fisheye image of your location. The lens is pointing to the zenith. Convert this image to an equirectangular image as follows:

```bash
set "FF=c:\ffmpeg\ffmpeg"     :: Path to FFmpeg
set "IN=s3.jpg"               :: Input fisheye image, south at the bottom
set "SQ=3648"                 :: Height of input image (width doesn't care)
set "OUT=s3.png"              :: Stereographic output image 360x90 pixel,
                              :: north at the left, south in the center, horizon at the top, zenith at the bottom

%FF% -i %IN% -lavfi
"crop=ih:ih,scale=180:180,pad=w=2*iw,v360=input=dfisheye:output=e:rorder='rpy':roll=180:pitch=90,crop=ih:ih/2:y=0,vflip"
-y %OUT%
```

Open the output image with IrfanView. The X coordinate is the azimuth angle in degrees, and the Y coordinate is the height over the horizon in degrees. Now you can manually read out the pixel positions of the horizon line for all azimuth angles in 5- or 10-degree steps.

Then you can insert the horizon line at the beginning of the horizon.dat file:

```
hor 32 0 0   ; these are the RGB colors
0  25
10 21
20 20
30 16
40 22
50 18
60 16
70 22
80 19
90 16
100 22
```

472
32.7 Switch between several user-defined horizon files

If you have several horizon files and you want to switch between them, you can create one batch file for each horizon, with this content:

```plaintext
copy d:\guide9\my_horizon_1.dat d:\guide9\horizon.dat
```

Put the batch files on the desktop and execute one of them by double-clicking. It will automatically overwrite the horizon.dat file with your own horizon file. You can change the horizon file while Guide is running. After overwriting the horizon file, just click in the Guide window to refresh the graphics and then you see the new horizon.

32.8 Install the Gaia2 catalog for Guide 9.1

Thanks to Jost Jahn who made it possible to download the Gaia2 catalog for Guide 9.1 here:

http://www.gaia2.de/index.html

Simply follow the instructions there.

32.9 Set up Guide 9.1 to show only the Gaia2 catalog

This is useful for making realistic timelapse videos of the proper motion of stars over 10000's of years. The trick is to create a second Guide folder which contains a minimal installation of Guide 9.1, with only those files that are absolutely required. No star catalogs are present in this folder.

- Create a new folder "guide_gaia_only"
• Copy the following files and folders from the original guide folder to the new folder: cache (folder), ngcic (folder), astnum, bitfont, cometg.dat, constbnd.ove, constlab.ove, gaia-std.tdf, gscdata2.idx, guide.dat, guide9.exe, hotkey.dat, lunar.dll, marks.nam, maximum.dat, messier.hee, overlays.nam, startup.mar, strings.dat, tdf_list.dat, temp_mar.txt, vsop.bin, win_meng.dat and win_menu.dat

• Open the file "gaia-std.tdf" with an editor and search/replace "file !:\STD\" to "file D:\Guide\STD\" using the actual path to the Gaia catalog. You don't want to have this catalog on your harddisk twice. There are 180 instances in the file that must be changed.

• Start Guide in the new folder.
How to create equirectangular images of the night sky:

http://paulbourke.net/dome/stellariumsphere/
DeepSkyStacker 4.2.3 can be downloaded here:  http://deepskystacker.free.fr/english/index.html
Support group:  https://groups.io/g/DeepSkyStacker

A big advantage of version 4.2.2 and later is that it can read the RAW files from the Canon 5D-MK4.

The language used by DeepSkyStacker is automatically set from the language used in the operating system. If you want to force another language you can change it from the "About" box.

Known problems:
When you open the light / dark / flat / offset images, unfortunately DSS always opens by default the folder from the last session. Same problem when you save the file list. Take care that you don't accidentally overwrite the file list from the last session! There is no known workaround to fix this problem. You have to select five times the same new folder!

34.1 How to align images without stacking them

Let's assume you want to align several images of star fields, for example if you want to make a timelapse from them. This is possible with DSS if you go to Options --> Settings --> Stacking settings --> Intermediate Files and then tick the box at "Create a registered / calibrated file for each light frame". The resulting intermediate images are calibrated, shifted and de-rotated. You can use FFmpeg to make a timelapse from them.
34.2 How to stack on comets with known motion

Normally the comet must be marked in at least 3 images: The first, the last and the reference image. If the first or last image is the reference image, then two images are sufficient. Marking the comet is simple if the comet is clearly visible in the images.

However things are getting difficult if either the comet is invisible (because it's too faint and hidden in the noise) or if the comet is so diffuse that it's difficult to define it's center. In these cases you can proceed as follows:

- It's required that north is up in all images, and that all images are already registered.
- Use the first or the last image as reference image. Use that one with the higher score. That means you have to mark the comet only in two images.
- Mark the same star as a comet in the first and last image. It's best to choose a star near the comet.
- These two images must also be checked in the leftmost column in the file list.
- Save the file list. It's not required to close DSS.
- The motion of a comet can be found for example in Guide9.1, if you make a right click on the comet and then click on "More info". The RA and DE motions are given in the last line in degrees/day. Example: "Motion is -0.42 degrees/day in RA, -0.36 degrees/day in dec"
- Calculate the time difference between the first and last image in the unit "days". Example: 2h55m37s - 2h40m57s = 0.0102d
- Calculate how far the comet has moved in RA and De during this time. Example: -0.42°/d * 0.0102d = -0.00428°, -0.35°/d * 0.012d = -0.00367°
- Calculate the image scale in degrees/pixel. Example for a full frame camera with 36mm image width, 6744 horizontal pixels and a 400mm lens: \( \arctan(36\text{mm} / (6744 \times 400\text{mm})) = 0.000765°/\text{pixel} \)
- Now you can calculate how man pixels the comet has moved between the two images. Example: \( X = 0.00428° / 0.000765°/\text{pixel} = -5.60 \text{ pixel (to the left)} \), \( Y = -0.00367° / 0.000765°/\text{pixel} = -4.80 \text{ pixel (downwards)} \)
- Open the file "last_image.info.txt" with an editor. In the 6th line from the top is the position of the comet, for example: Comet = 4022.09, 1957.80
- Now modify these coordinates. To the X coordinate you add the value that you calculated above \( 4022.09 + (-5.60) = 4016.49 \) and from the Y coordinate you subtract the value from above. That's because the direction of the Y axis is top down. \( 1957.80 - (-4.80) = 1962.60 \)
- Save the file with the same filename.
- Open the file list in DSS.
- Check that in the first image the comet is still marked at the same star as before. However in the last image the violet circle must have moved
with respect to the star. Check that it has moved in the correct direction of the comet's movement.

- Check if under Settings / Stacking_Settings / Comet the box at "Comet Stacking" is ticked, and then start stacking.

Print out this form for the calculations:

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA_speed = RA_speed</td>
<td>RA comet motion in degrees per day =</td>
</tr>
<tr>
<td>DE_speed = DE_speed</td>
<td>DE comet motion in degrees per day =</td>
</tr>
<tr>
<td>T1 = Time of first image</td>
<td></td>
</tr>
<tr>
<td>T2 = Time of last image</td>
<td></td>
</tr>
<tr>
<td>dT = Time difference in days = T2 - T1 =</td>
<td></td>
</tr>
<tr>
<td>RA_deg = RA_deg</td>
<td>RA comet movement in degrees = RA_speed * dT =</td>
</tr>
<tr>
<td>DE_deg = DE_deg</td>
<td>DE comet movement in degrees = DE_speed * dT =</td>
</tr>
<tr>
<td>S = Image scale in degrees per pixel =</td>
<td>For Canon 5D-MK4 with 400mm lens: 0.000765°/pixel</td>
</tr>
<tr>
<td>RA_pix = RA_pix</td>
<td>RA comet movement in pixels = RA_deg / S =</td>
</tr>
<tr>
<td>DE_pix = DE_pix</td>
<td>DE comet movement in pixels = DE_deg / S =</td>
</tr>
<tr>
<td>X_old = Original X value</td>
<td></td>
</tr>
<tr>
<td>Y_old = Original Y value</td>
<td></td>
</tr>
<tr>
<td>X_new = New X value</td>
<td>X_new = X_old + RA_pix =</td>
</tr>
<tr>
<td>Y_new = New Y value</td>
<td>Y_new = Y_old - DE_pix =</td>
</tr>
</tbody>
</table>
From time to time I have to deal with C# programming to stay in practice. This year’s task is: A digital topographic map is available for an area of approx. 10km x 10km. In addition, digital elevation data is available, which is to be superimposed on the map. Virtual noise sources are to be placed along the roads on the map. Then those places should be found that are affected as little as possible by the noise sources. It should be assumed that the sound propagates in a straight line and can be shadowed by mountains. We are therefore looking for places from which all noise sources are hidden behind mountains. These are the places where I want to record nature sounds.

35.1 Where is the best place for recording nature sounds?

That's more difficult than you might think, because you have to find a place without any disturbing noise:

- Road and rail traffic requires a distance of several kilometres. It’s helpful if there is no direct line of sight, i.e. mountains in between are advantageous.
- If you don’t want to record the sound of running water, you have to avoid valleys.
- Wind noise is disturbing already at quite small wind speeds despite fur windshield. Wind noise can be attenuated by a high pass filter. However on days with strong wind it’s wasted time to make a record.
- Airplanes cause that approx. 50% of the sound recordings are unusable (even in the Harz Mountains in Germany, where there is no large airfield within a radius of 80km).
Digital topographic maps

A very good source for free digital topographic maps is OpenTopoMap, the web version is here: https://opentopomap.org/#map=13/51.66473/10.42482

On this page is briefly mentioned (in german) how tiles of size 256x256 pixels can be downloaded: https://opentopomap.org/about

This is a sample download of one tile (this is the place where I live): https://a.tile.opentopomap.org/13/4331/2718.png

In this case the zoom level is 13, the X value is 4331 and the Y value is 2718.

This page contains very detailed explanations about the folder structure and mathematics of the zoom levels and coordinates: https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames

Coordinate transformation from latitude and longitude to X and Y:

\[
\begin{align*}
n & = 2^\text{zoom} \\
x & = n \times ((\text{lon} \text{ deg} + 180) \div 360) \\
y & = n \times (1 - (\log(\tan(\text{lat} \text{ rad}) + \sec(\text{lat} \text{ rad}))) \div \pi)) \div 2
\end{align*}
\]

Coordinate transformation from X and Y to latitude and longitude:

\[
\begin{align*}
n & = 2^\text{zoom} \\
\text{lon deg} & = X \div n \times 360.0 - 180.0 \\
\text{lat rad} & = \arctan(\sinh(\pi \times (1 - 2 \times Y \div n))) \\
\text{lat deg} & = \text{lat rad} \times 180.0 \div \pi
\end{align*}
\]

This returns the north west (top left) corner of the tile. Use X+1 and/or Y+1 to get the other corners. Use X+0.5 and Y+0.5 to get the coordinates of the tile’s center.

Calculate the resolution:

\[
\begin{align*}
\text{resolution} & = 156543.03 \text{ meters/pixel} \times \cos(\text{latitude}) \div (2^\text{zoom})
\end{align*}
\]

Note: You can create your own layers in the french version of OpenStreetMap: umap.openstreetmap.fr
35.3 Digital elevation data

I found several sources for free digital elevation data:


- NASA Earthdata Search: [https://search.earthdata.nasa.gov/search](https://search.earthdata.nasa.gov/search) After registering you can download free elevation data with 1 arcsec resolution (which is about 27m in latitude, and less than 27m in longitude). Search for the ASTER Digital Elevation Model (AST14DEM): [https://lpdaac.usgs.gov/products/ast14demv003/](https://lpdaac.usgs.gov/products/ast14demv003/) However the elevation data seems to be inaccurate, as I found some peaks in the mountains about 40m too low.

- [https://opendem.info/download_srtm.html](https://opendem.info/download_srtm.html) This is a very large 272MB GeoTiff file of Germany with 13201x10801 pixels. The resolution is 1200 pixels per degree. You can choose between surface data (including buildings and vegetation) and terrain data (without buildings and vegetation). The elevation data isn’t perfect, as I found some peaks in the mountains up to 17m too low. But for my project that’s good enough and I did use the terrain data file.

This really large 16-bit GeoTiff file contains Germany from longitude 5° to 16° and from latitude 47° to 56°. It covers a 11° longitude range with 13201 pixels and a 9° latitude range with 10801 pixels. The top left corner is at 5° longitude and 56° latitude. Please note that the resolution (in Meter) is different for longitude and latitude. The pixels aren’t square. The pixel size is about 92.63m x 149.74m.

Of course we need only a small part of the map, so how to crop it? GeoTiff’s can be read the same way as Tiff’s, but the software must be able to read and write 16-bit data.

- IrfanView can read 16-bit Tiff, but converts it internally to 8-bit.
- Fitswork can read and write 16-bit Tiff and crop the region, but it can’t write PGM files.
- Gimp can read and write 16-Bit Tiff. You can crop the region as follows: Make a double click on the "Rectangle Select Tool". Draw a rectangle in the picture. Now fill in the values for position and size. Then use "Image / Crop_to_Selection". Gimp can also save the output image as 16-bit ASCII PGM (P2 Portable Gray Map) file, which is easy to read by C# code.
- FFmpeg can read and write 16-bit Tiff and also crop the region. It can write binary PGM (P5) files, but unfortunately it can’t write ASCII PGM (P2) files.
Here is an example for cropping a region of the GeoTiff with FFmpeg:

```rem
Crop a large GeoTiff file and save it as Tiff or binary PGM file

set "FF=c:\ffmpeg\ffmpeg" :: Path to FFmpeg
set "IN=srtm_germany_dtm.tif" :: Input GeoTiff file
set "WIDTH=264" :: Width
set "HEIGHT=131" :: Height
set "LEFT=6340" :: Left edge
set "TOP=5128" :: Top edge
set "OUT=elevation.tif" :: Output Tiff or binary PGM file; it isn't possible to write an ASCII PGM file

%FF% -i %IN% -vf format=pix_fmts=gray16le,crop=%WIDTH%:%HEIGHT%:%LEFT%:%TOP% -y %OUT%

pause```

However, it's also possible to read the GeoTiff file directly with C# code, using the external BitMiracle.LibTiff.NET library. I did use this library in my C# code.


This is the link to the library: [https://bitmiracle.com/libtiff/](https://bitmiracle.com/libtiff/)

Here is the source code:

**Form1.cs:**

```csharp
class Form1 : Form
{
    // Global variables
    Bitmap b1 = new Bitmap(4096, 4096); // for 16x16 tiles
}
```
Graphics g1;
Int16[,] elevation = new Int16[4096, 4096];
List<NoiseSource> NoiseList = new List<NoiseSource>();

int map_zoom = 13;
int map_center_x = 4332;
int map_center_y = 2718;
int tiles_x;
int tiles_y;
double resolution;
bool view_all = false;

CultureInfo invC = CultureInfo.InvariantCulture;
SolidBrush brush_red_transparent = new SolidBrush(Color.FromArgb(96, Color.Red)); // red with transparency
SolidBrush brush_blue_transparent = new SolidBrush(Color.FromArgb(96, Color.Blue)); // blue with transparency

public class NoiseSource
{
    private double longitude, latitude;
    private int radius;
    private string type;
    private int x, y;

    public double Longitude { get { return longitude; } set { longitude = value; } }
    public double Latitude { get { return latitude; } set { latitude = value; } }
    public int Radius { get { return radius; } set { radius = value; } }
    public string Type { get { return type; } set { type = value; } }
    public int X { get { return x; } set { x = value; } }
    public int Y { get { return y; } set { y = value; } }

    public NoiseSource(double longitude, double latitude, int radius, string type) // Constructor
    {
        this.longitude = longitude;
        this.latitude = latitude;
        this.radius = radius;
        this.type = type;
    }
}

public Form1()
{
    InitializeComponent();
    g1 = Graphics.FromImage(b1);
    splitContainer1.IsSplitterFixed = true;
    splitContainer1.FixedPanel = FixedPanel.Panel2;
    splitContainer2.IsSplitterFixed = true;
splitContainer2.FixedPanel = FixedPanel.Panel2;

tiles_x = (int)numericUpDown4.Value;
tiles_y = (int)numericUpDown5.Value;

pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
pictureBox1.Size = new Size(256 * tiles_x, 256 * tiles_y);
panel1.AutoScroll = true;

this.Text = "Find the most silent places in the mountains";
richTextBox2.Text = "Map data: © OpenStreetMap-Mitwirkende, SRTM | Kartendarstellung: © OpenTopoMap (CC-BY-SA)\n" +
    "Elevation data: Open Database License: http://opendatacommons.org/licenses/odbl/1.0/. " +
    "Any rights in individual contents of the database are licensed under the Database Contents License: " +
    " http://opendatacommons.org/licenses/dbcl/1.0/";
}

private void draw_map()
{
    textBox1.Text = (256 * tiles_x).ToString() + "x" + (256 * tiles_y).ToString();
    resolution = 156543.03 * Math.Cos(xyz2latitude(map_center_y, map_zoom) / 180.0 * Math.PI) / Math.Pow(2, map_zoom);
    textBox4.Text = resolution.ToString("F2", invC) + " m/Pixel";

    WebClient wc = new WebClient();
    ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
    for (int x = 0; x < tiles_x; x++) // Load topographic map tiles of size 256x256 pixels each
    {
        int xx = x + map_center_x - tiles_x / 2;
        for (int y = 0; y < tiles_y; y++)
        {
            int yy = y + map_center_y - tiles_y / 2;
            string path_filename = map_zoom.ToString(); // "13"
            DirectoryInfo di1 = new DirectoryInfo(path_filename);
            if (di1.Exists == false)
                di1.Create();
            path_filename += "/" + xx.ToString(); // "13/4330"
            DirectoryInfo di2 = new DirectoryInfo(path_filename);
            if (di2.Exists == false)
                di2.Create();
            path_filename += "/" + yy.ToString() + ".png"; // "13/4330/2716.png"
            FileInfo fi = new FileInfo(path_filename);
            if (fi.Exists == false) // if the file doesn't already exist, load it from the internet
            {
                try
                {
                    wc.DownloadFile("https://a.tile.openstopomap.org/" + path_filename, path_filename);
                }
                catch
                {
                    richTextBox3.AppendText("Error: Map can't be loaded\n");
                }
            }
        }
    }
}

485
try {
    g1.DrawImage(new Bitmap(path_filename), 256 * x, 256 * y);
} catch {
    // draw a green rectangle, if the file can't be loaded
    g1.FillRectangle(new SolidBrush(Color.FromName("lime")), 256*x, 256*y, 256, 256);
}

foreach (NoiseSource n in NoiseList) // draw all noisessources in the map
{
    double lon = n.Longitude;
    double lat = n.Latitude;
    string type = n.Type;
    double xd = longitude2xyz(lon, map_zoom);
    double yd = latitude2xyz(lat, map_zoom);
    int x = (int)(256 * (xd - map_center_x + tiles_x / 2));
    int y = (int)(256 * (yd - map_center_y + tiles_y / 2));
    g1.FillEllipse(new SolidBrush((type == "T") ? Color.Red : Color.Blue), x - 6, y - 6, 12, 12);
}

if (view_all)
{
    panel1.AutoScroll = false;
    pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;
    pictureBox1.Size = panel1.ClientRectangle.Size;
    Bitmap b2 = new Bitmap(256 * tiles_x, 256 * tiles_y);
    Graphics g2 = Graphics.FromImage(b2);
    g2.DrawImageUnscaledAndClipped(b1, new Rectangle(0, 0, 256 * tiles_x, 256 * tiles_y));
    pictureBox1.Image = b2;
}
else
{
    panel1.AutoScroll = true;
    pictureBox1.SizeMode = PictureBoxSizeMode.Normal;
    pictureBox1.Size = new Size(256 * tiles_x, 256 * tiles_y);
    pictureBox1.Image = b1;
}

Point p = new Point();
p.X = (pictureBox1.Size.Width - panel1.ClientRectangle.Width) / 2;  // center the scroll bars
p.Y = (pictureBox1.Size.Height - panel1.ClientRectangle.Height) / 2;
panel1.AutoScrollPosition = p;
string elevation_file = "srtm_germany_dtm.tif"; // get the elevations from this file
FileInfo fi2 = new FileInfo(elevation_file);
if (!fi2.Exists)
    MessageBox.Show("File not found: " + elevation_file);

using (Tiff tiff = Tiff.Open("srtm_germany_dtm.tif", "r"))
{
    int height = tiff.GetField(TiffTag.IMAGELENGTH)[0].ToInt();
   getFieldValue() modelPixelScaleTag = tiff.GetField((TiffTag)33550);
    FieldValue[] modelTiepointTag = tiff.GetField((TiffTag)33922);

    byte[] modelPixelScale = modelPixelScaleTag[1].GetBytes();
    double pixelSizeX = BitConverter.ToDouble(modelPixelScale, 0);
    double pixelSizeY = BitConverter.ToDouble(modelPixelScale, 8) * -1;

    byte[] modelTransformation = modelTiepointTag[1].GetBytes();
    double originLon = BitConverter.ToDouble(modelTransformation, 24);
    double originLat = BitConverter.ToDouble(modelTransformation, 32);

    double startLat = originLat + (pixelSizeY / 2.0);
    double startLon = originLon + (pixelSizeX / 2.0);

    // Loading i th Line
    tiff.ReadScanline(scanline0, i);
    tiff.ReadScanline(scanline1, i + 1);
}

for (int y = 0; y < 256 * tiles_y; y++)
{
    double double_i = (xyz2latitude((double)(map_center_y - tiles_y / 2) + ((double)y / 256), map_zoom) - startLat) / pixelSizeY;
    int i = (int) double_i;
    double fraction_i = double_i - i;
    try
    {
        tiff.ReadScanline(scanline0, i); // Loading i th Line
        tiff.ReadScanline(scanline1, i + 1); // Loading i+1 th Line
    }
for (int x = 0; x < 256 * tiles_x; x++)
{
    double double_j = xyz2longitude((double)(map_center_x - tiles_x / 2) + ((double)x / 256), map_zoom) - startLon) / pixelSizeX;
    int j = (int) double_j;
    double fraction_j = double_j - j;
    if ((j >= 0) && (j + 3 < scanline0.Length)) // Interpolate the elevation between 4 points
    {
        short ele_topleft = (short)(scanline0[2 * j] + 256 * (short)scanline0[2 * j + 1]);
        short ele_topright = (short)(scanline0[2 * j + 2] + 256 * (short)scanline0[2 * j + 3]);
        double ele_top = ele_topleft + fraction_j * (ele_topright - ele_topleft);
        short ele_bottomleft = (short)(scanline1[2 * j] + 256 * (short)scanline1[2 * j + 1]);
        double ele_bottom = ele_bottomleft + fraction_j * (ele_bottomright - ele_bottomleft);
        elevation[x, y] = (short)(ele_top + fraction_i * (ele_bottom - ele_top));
    }
    else
    {
        elevation[x, y] = 0; // Elevation file doesn't cover this column, set elevation to zero
    }
}
}
catch
{
    for (int x = 0; x < 256 * tiles_x; x++)
    {
        elevation[x, y] = 0; // Elevation file doesn't cover this row, set elevation to zero
    }
}
}

public void calculate()
{
    int noise_height = (int)numericUpDown1.Value;
    int microphone_height = (int)numericUpDown2.Value;
    int raster = (Int)numericUpDown6.Value;

    foreach (NoiseSource n in NoiseList)
    {
        n.X = (int)(256 * (longitude2xyz(n.Longitude, map_zoom) - map_center_x + tiles_x / 2));
        n.Y = (int)(256 * (latitude2xyz(n.Latitude, map_zoom) - map_center_y + tiles_y / 2));
    }

    for (int py = 0; py < 256 * tiles_y; py += raster)
    {
        for (int px = 0; px < 256 * tiles_x; px += raster)
```java
{
    Boolean traffic = false;
    Boolean running_water = false;

    foreach (NoiseSource n in NoiseList)
    {
        int nx = n.X;
        if ((nx < 0) || (nx >= 256 * tiles_x))
            continue;

        int ny = n.Y;
        if ((ny < 0) || (ny >= 256 * tiles_y))
            continue;

        if (distance_in_pixels == 0)
            distance_in_pixels++;

        int distance_in_meters = (int)(resolution * distance_in_pixels);

        int ne = noise_height + elevation[nx, ny]; // elevation of the noise source
        int pe = microphone_height + elevation[px, py]; // elevation of the point px,py

        Boolean sight_of_line = true;
        if (distance_in_meters < n.Radius)
        {
            for (int i = 1; i < distance_in_pixels; i += raster)
            {
                double d = (double)i / distance_in_pixels; // 0...1
                int x = (int)(nx + d * (px - nx));
                int y = (int)(ny + d * (py - ny));
                int h = (int)(ne + d * (pe - ne));
                if (elevation[x, y] > h)
                {
                    sight_of_line = false;
                    break;
                }
            }
            if (sight_of_line == true)
            {
                if (n.Type == "T")
                    traffic = true;
                else
                    running_water = true;
                break;
            }
        }
    }

    if (traffic == true)
    {
        // Further processing...
    }
}
```
private void button1_Click(object sender, EventArgs e)  // Calculate the noise map
    {
        draw_map();
        calculate();
    }

private void button2_Click(object sender, EventArgs e)  // Save the map as PNG
    {
        Bitmap b2 = new Bitmap(256 * tiles_x, 256 * tiles_y);
        Graphics g2 = Graphics.FromImage(b2);
        g2.DrawImageUnscaledAndClipped(b1, new Rectangle(0, 0, 256 * tiles_x, 256 * tiles_y));
        if (saveFileDialog2.ShowDialog() == DialogResult.OK)
            b2.Save(saveFileDialog2.FileName);
    }

class xyz2longitude
    {
        public double xz(longitude, int z)
        {
            return (longitude + 180.0) / 360.0;
        }
    }

public double latitude2xyz(double latitude, int z)
    {
        double lat_rad = latitude / 180.0 * Math.PI;
        return (Math.Pow(2, map_zoom) * (1.0 - (Math.Log(Math.Tan(lat_rad) + 1.0 / Math.Cos(lat_rad)) / Math.PI)) / 2.0);
    }
if (e.Button == MouseButtons.Left)  // Add a noisesource to the map and to the list
{
    if (view_all)
        MessageBox.Show("Please click \"Toggle Display Mode\" before adding a noise source");
    else
    {
        string type = "T";
        if (radioButton2.Checked) type = "W";
        if (type == "T")
            g1.FillEllipse(new SolidBrush(Color.Red), e.X - 6, e.Y - 6, 12, 12);
        else
            g1.FillEllipse(new SolidBrush(Color.Blue), e.X - 6, e.Y - 6, 12, 12);
        double n = Math.Pow(2, map_zoom);
        double lon = xyz2longitude(((double)(map_center_x - tiles_x / 2) + ((double)e.X / 256), map_zoom);
        double lat = xyz2latitude(((double)(map_center_y - tiles_y / 2) + ((double)e.Y / 256), map_zoom);
        int radius = (int)numericUpDown3.Value;
        richTextBox1.AppendText(lon.ToString("F4", invC) + " " + lat.ToString("F4", invC) + " " + radius + " " + type + "\n");
        NoiseList.Add(new NoiseSource(lon, lat, radius, type));
        draw_map();
    }
}

private void pictureBox1_MouseMove(object sender, MouseEventArgs e)  // Show the mouse coordinates
{
    if (!view_all)
    {
        double lon = xyz2longitude(((double)(map_center_x - tiles_x / 2) + ((double)e.X / 256), map_zoom);
        double lat = xyz2latitude(((double)(map_center_y - tiles_y / 2) + ((double)e.Y / 256), map_zoom);
        textBox2.Text = lon.ToString("F4", invC) + "°";
        textBox3.Text = lat.ToString("F4", invC) + "°";
        textBox5.Text = elevation[e.X, e.Y].ToString();
        textBox6.Text = map_zoom.ToString() + " / " + (map_center_x - tiles_x / 2 + e.X / 256).ToString() + " / " + (map_center_y - tiles_y / 2 + e.Y / 256).ToString();
    }
    else
    {
        textBox2.Text = "";
        textBox3.Text = "";
        textBox5.Text = "";
        textBox6.Text = "";
    }
}

private void pictureBox1_MouseLeave(object sender, EventArgs e)  // Mouse did leave the map, clear the coordinates
{
    textBox2.Text = "";
    textBox3.Text = "";
    textBox6.Text = "";
}
private void button3_Click(object sender, EventArgs e)  // Save the noisesources file
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        richTextBox1.SaveFile(saveFileDialog1.FileName, RichTextBoxStreamType.PlainText);
}

private void button6_Click(object sender, EventArgs e)
{
    map_center_x--;  
    draw_map();
}

private void button7_Click(object sender, EventArgs e)
{
    map_center_x++;  
    draw_map();
}

private void button4_Click(object sender, EventArgs e)
{
    map_center_y--;  
    draw_map();
}

private void button5_Click(object sender, EventArgs e)
{
    map_center_y++;  
    draw_map();
}

private void button8_Click(object sender, EventArgs e)
{
    if (map_zoom < 17)
    {
        map_zoom++;  
        map_center_x *= 2;  
        map_center_y *= 2;  
        draw_map();
    }
    else
    {  
        Console.Beep(300, 500);
    }
}

private void button9_Click(object sender, EventArgs e)
{
    if (map_zoom > 8)
    {
        map_zoom--;  
        map_center_x /= 2;  
        map_center_y /= 2;
    }
}
```csharp
draw_map();
}
else
    Console.Beep(300, 500);
}

private void Form1_Shown(object sender, EventArgs e)
{
    draw_map();
}

private void numericUpDown4_ValueChanged(object sender, EventArgs e)
{
    tiles_x = (int)numericUpDown4.Value;
    pictureBox1.Size = new Size(256 * tiles_x, 256 * tiles_y);
    draw_map();
}

private void numericUpDown5_ValueChanged(object sender, EventArgs e)
{
    tiles_y = (int)numericUpDown5.Value;
    pictureBox1.Size = new Size(256 * tiles_x, 256 * tiles_y);
    draw_map();
}

private void button10_Click(object sender, EventArgs e)  // Load noise sources from file
{
    richTextBox1.Clear();
    NoiseList.Clear();
    using (TextReader noise = File.OpenText("noisesources.txt"))
    {
        string line;
        while ((line = noise.ReadLine()) != null)
        {
            if (line == "")
                break;
            string[] param = line.Split(new char[] { ' ' });
            double longitude = System.Convert.ToDouble(param[0], invC);
            double latitude = System.Convert.ToDouble(param[1], invC);
            int radius = System.Convert.ToInt32(param[2]);
            string type = param[3];
            NoiseList.Add(new NoiseSource(longitude, latitude, radius, type));
            richTextBox1.AppendText(line + "\n");
        }
    }
}

private void button11_Click(object sender, EventArgs e)  // Toggle display mode
{
}
```

Form1.designer.cs

namespace Topography
{
    partial class Form1
    {
        /// <summary>
        /// Erforderliche Designervariable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Verwendete Ressourcen bereinigen.
        /// </summary>
        /// <param name="disposing">True, wenn verwaltete Ressourcen gelöscht werden sollen; andernfalls False.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Vom Windows Form-Designer generierter Code

        /// <summary>
        /// Erforderliche Methode für die Designerunterstützung.
        /// Der Inhalt der Methode darf nicht mit dem Code-Editor geändert werden.
        /// </summary>
        private void InitializeComponent()
        {
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.button1 = new System.Windows.Forms.Button();
            this.panell1 = new System.Windows.Forms.Panel();
            this.splitContainer1 = new System.Windows.Forms.SplitContainer();
            this.button11 = new System.Windows.Forms.Button();
            this.richTextBox1 = new System.Windows.Forms.RichTextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.textBox6 = new System.Windows.Forms.TextBox();
            this.label10 = new System.Windows.Forms.Label();
            this.textBox5 = new System.Windows.Forms.TextBox();
        }

        #endregion
    }
}
this.splitContainer1.Panel2.Controls.Add(this.numericUpDown6);
this.splitContainer1.Panel2.Controls.Add(this.label12);
this.splitContainer1.Panel2.Controls.Add(this.button11);
this.splitContainer1.Panel2.Controls.Add(this.richTextBox1);
this.splitContainer1.Panel2.Controls.Add(this.label11);
this.splitContainer1.Panel2.Controls.Add(this.numericUpDown4);
this.splitContainer1.Panel2.Controls.Add(this.button9);
this.splitContainer1.Panel2.Controls.Add(this.button8);
this.splitContainer1.Panel2.Controls.Add(this.button7);
this.splitContainer1.Panel2.Controls.Add(this.button6);
this.splitContainer1.Panel2.Controls.Add(this.button5);
this.splitContainer1.Panel2.Controls.Add(this.button4);
this.splitContainer1.Panel2.Controls.Add(this.button3);
this.splitContainer1.Panel2.Controls.Add(this.button2);
this.splitContainer1.Panel2.Controls.Add(this.button1);
this.splitContainer1.Size = new System.Drawing.Size(1305, 865);
this.splitContainer1.SplitterDistance = 1131;
this.splitContainer1.TabIndex = 3;
// button11
//
this.button11.Location = new System.Drawing.Point(12, 187);
this.button11.Name = "button11";
this.button11.Size = new System.Drawing.Size(145, 24);
this.button11.TabIndex = 38;
this.button11.Text = "Toggle Display Mode";
this.button11.UseVisualStyleBackColor = true;
this.button11.Click += new System.EventHandler(this.button11_Click);
//
// richTextBox1
//
this.richTextBox1.Location = new System.Drawing.Point(7, 630);
this.richTextBox1.Name = "richTextBox1";
this.richTextBox1.Size = new System.Drawing.Size(157, 188);
this.richTextBox1.TabIndex = 37;
this.richTextBox1.Text = "";
//
// label11
//
this.label11.AutoSize = true;
this.label11.Location = new System.Drawing.Point(15, 351);
this.label11.Name = "label11";
this.label11.Size = new System.Drawing.Size(27, 13);
this.label11.TabIndex = 36;
this.label11.Text = "Tile:";
//
// textBox6
//
this.textBox6.Location = new System.Drawing.Point(48, 348);
this.textBox6.Name = "textBox6";
this.textBox6.ReadOnly = true;
this.textBox6.Size = new System.Drawing.Size(110, 20);
this.textBox6.TabIndex = 35;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(15, 302);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(54, 13);
this.label10.TabIndex = 34;
this.label10.Text = "Elevation:";
//
// textBox5
//
this.textBox5.Location = new System.Drawing.Point(81, 299);
this.textBox5.Name = "textBox5";
this.textBox5.ReadOnly = true;
this.textBox5.Size = new System.Drawing.Size(77, 20);
this.textBox5.TabIndex = 33;
//
// textBox4
//
this.textBox4.Location = new System.Drawing.Point(81, 325);
this.textBox4.Name = "textBox4";
this.textBox4.ReadOnly = true;
this.textBox4.Size = new System.Drawing.Size(77, 20);
this.textBox4.TabIndex = 32;
this.button10.Size = new System.Drawing.Size(73, 34);
this.button10.TabIndex = 26;
this.button10.Text = "Load Noise Sources";
this.button10.UseVisualStyleBackColor = true;
this.button10.Click += new System.EventHandler(this.button10_Click);

// textBox1
//
this.textBox1.Location = new System.Drawing.Point(69, 161);
this.textBox1.Name = "textBox1";
this.textBox1.ReadOnly = true;
this.textBox1.Size = new System.Drawing.Size(72, 20);
this.textBox1.TabIndex = 25;

// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(9, 164);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(54, 13);
this.label6.TabIndex = 24;
this.label6.Text = "Map Size:";

// label5
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(9, 136);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(70, 13);
this.label5.TabIndex = 23;
this.label5.Text = "Vertical Tiles:";

// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(9, 108);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(82, 13);
this.label4.TabIndex = 22;
this.label4.Text = "Horizontal Tiles:";

// numericUpDown5
//
this.numericUpDown5.Increment = new decimal(new int[] { 2, 0, 0, 0 });
this.numericUpDown5.Location = new System.Drawing.Point(101, 132);
this.numericUpDown5.Maximum = new decimal(new int[] { 16, 0, 0, 0 });
this.numericUpDown5.Minimum = new decimal(new int[] { 4, 0, 0, 0 });
this.numericUpDown5.Name = "numericUpDown5";
this.numericUpDown5.Size = new System.Drawing.Size(40, 22);
this.numericUpDown5.TabIndex = 21;
this.numericUpDown5.Value = new decimal(new int[] { 4, 0, 0, 0 });
this.numericUpDown5.ValueChanged += new System.EventHandler(this.numericUpDown5_ValueChanged);

// numericUpDown4
this.numericUpDown4.Increment = new decimal(new int[] { 2, 0, 0, 0 });
this.numericUpDown4.Location = new System.Drawing.Point(101, 104);
this.numericUpDown4.Maximum = new decimal(new int[] { 16, 0, 0, 0 });
this.numericUpDown4.Minimum = new decimal(new int[] { 4, 0, 0, 0 });
this.numericUpDown4.Name = "numericUpDown4";
this.numericUpDown4.Size = new System.Drawing.Size(40, 22);
this.numericUpDown4.TabIndex = 20;
this.numericUpDown4.Value = new decimal(new int[] { 6, 0, 0, 0 });
this.numericUpDown4.ValueChanged += new System.EventHandler(this.numericUpDown4_ValueChanged);
// button9
//
this.button9.Location = new System.Drawing.Point(79, 70);
this.button9.Name = "button9";
this.button9.Size = new System.Drawing.Size(66, 25);
this.button9.TabIndex = 19;
this.button9.Text = "zoom out";
this.button9.UseVisualStyleBackColor = true;
this.button9.Click += new System.EventHandler(this.button9_Click);

// button8
//
this.button8.Location = new System.Drawing.Point(79, 11);
this.button8.Name = "button8";
this.button8.Size = new System.Drawing.Size(66, 25);
this.button8.TabIndex = 18;
this.button8.Text = "zoom in";
this.button8.UseVisualStyleBackColor = true;
this.button8.Click += new System.EventHandler(this.button8_Click);

// button7
//
this.button7.Location = new System.Drawing.Point(56, 40);
this.button7.Name = "button7";
this.button7.Size = new System.Drawing.Size(25, 25);
this.button7.TabIndex = 17;
this.button7.Text = "E";
this.button7.UseVisualStyleBackColor = true;
this.button7.Click += new System.EventHandler(this.button7_Click);

// button6
//
this.button6.Location = new System.Drawing.Point(3, 40);
this.button6.Name = "button6";
this.button6.Size = new System.Drawing.Size(25, 25);
this.button6.TabIndex = 16;
this.button6.Text = "W";
this.button6.UseVisualStyleBackColor = true;
this.button6.Click += new System.EventHandler(this.button6_Click);

// button5
//
this.button5.Location = new System.Drawing.Point(31, 70);
this.button5.Name = "button5";
this.button5.Size = new System.Drawing.Size(23, 24);
this.button5.TabIndex = 15;
this.button5.Text = "S";
this.button5.UseVisualStyleBackColor = true;
this.button5.Click += new System.EventHandler(this.button5_Click);
// button4
//
// this.button4.Location = new System.Drawing.Point(31, 13);
this.button4.Name = "button4";
this.button4.Size = new System.Drawing.Size(23, 23);
this.button4.TabIndex = 14;
this.button4.Text = "N";
this.button4.UseVisualStyleBackColor = true;
this.button4.Click += new System.EventHandler(this.button4_Click);

// richTextBox3
//
// this.richTextBox3.Location = new System.Drawing.Point(5, 513);
this.richTextBox3.Name = "richTextBox3";
this.richTextBox3.Size = new System.Drawing.Size(160, 40);
this.richTextBox3.TabIndex = 13;
this.richTextBox3.Text = "";

// radioButton2
//
// this.radioButton2.AutoSize = true;
this.radioButton2.Location = new System.Drawing.Point(18, 607);
this.radioButton2.Name = "radioButton2";
this.radioButton2.Size = new System.Drawing.Size(127, 17);
this.radioButton2.TabIndex = 12;
this.radioButton2.Text = "Running Water Noise";
this.radioButton2.UseVisualStyleBackColor = true;

// radioButton1
//
// this.radioButton1.AutoSize = true;
this.radioButton1.Checked = true;
this.radioButton1.Location = new System.Drawing.Point(18, 587);
this.radioButton1.Name = "radioButton1";
this.radioButton1.Size = new System.Drawing.Size(85, 17);
this.radioButton1.TabIndex = 11;
this.radioButton1.TabStop = true;
this.radioButton1.Text = "Traffic Noise";
this.radioButton1.UseVisualStyleBackColor = true;

// label3
//
// this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(4, 563);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(87, 13);
this.label3.TabIndex = 10;
this.label3.Text = "Noise Radius [m]";
// numericUpDown3
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
this.numericUpDown3.Increment = new decimal(new int[] { 10, 0, 0, 0});
this.numericUpDown3.Location = new System.Drawing.Point(101, 559);
this.numericUpDown3.Maximum = new decimal(new int[] { 20000, 0, 0, 0 });
this.numericUpDown3.Minimum = new decimal(new int[] { 10, 0, 0, 0 });
this.numericUpDown3.Name = "numericUpDown3";
this.numericUpDown3.Size = new System.Drawing.Size(59, 22);
this.numericUpDown3.TabIndex = 9;
this.numericUpDown3.Value = new decimal(new int[] { 10000, 0, 0, 0 });
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(7, 433);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(114, 13);
this.label2.TabIndex = 8;
this.label2.Text = "Microphone Height [m]";
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(7, 405);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(122, 13);
this.label1.TabIndex = 7;
this.label1.Text = "Noise Source Height [m]";
//
// numericUpDown2
//
System.Drawing.GraphicsUnit.Point, (byte)(0));
    this.numericUpDown2.Location = new System.Drawing.Point(130, 427);
    this.numericUpDown2.Maximum = new decimal(new int[] { 20, 0, 0, 0});
    this.numericUpDown2.Name = "numericUpDown2";
    this.numericUpDown2.Size = new System.Drawing.Size(33, 22);
    this.numericUpDown2.TabIndex = 6;
    this.numericUpDown2.Value = new decimal(new int[] { 5, 0, 0, 0});
    //
    // numericUpDown1
    //
    this.numericUpDown1.Location = new System.Drawing.Point(130, 401);
    this.numericUpDown1.Maximum = new decimal(new int[] { 20, 0, 0, 0 });
    this.numericUpDown1.Name = "numericUpDown1";
    this.numericUpDown1.Size = new System.Drawing.Size(33, 22);
    this.numericUpDown1.TabIndex = 5;
    this.numericUpDown1.Value = new decimal(new int[] { 5, 0, 0, 0});
    // button3
    //
    this.button3.Location = new System.Drawing.Point(85, 828);
    this.button3.Name = "button3";
    this.button3.Size = new System.Drawing.Size(73, 34);
    this.button3.TabIndex = 4;
    this.button3.Text = "Save Noise Sources";
    this.button3.UseVisualStyleBackColor = true;
    this.button3.Click += new System.EventHandler(this.button3_Click);
    // button2
    //
    this.button2.Location = new System.Drawing.Point(12, 217);
    this.button2.Name = "button2";
    this.button2.Size = new System.Drawing.Size(146, 24);
this.button2.TabIndex = 2;
this.button2.Text = "Save Map as PNG";
this.button2.UseVisualStyleBackColor = true;
this.button2.Click += new System.EventHandler(this.button2_Click);

// saveFileDialog1
//
this.saveFileDialog1.DefaultExt = "txt";
this.saveFileDialog1.FileName = "noisesources.txt";

// richTextBox2
//
this.richTextBox2.Dock = System.Windows.Forms.DockStyle.Fill;
this.richTextBox2.Location = new System.Drawing.Point(0, 0);
this.richTextBox2.Name = "richTextBox2";
this.richTextBox2.Size = new System.Drawing.Size(1305, 39);
this.richTextBox2.TabIndex = 4;
this.richTextBox2.Text = "";

// splitContainer2
//
this.splitContainer2.Location = new System.Drawing.Point(0, 0);
this.splitContainer2.Name = "splitContainer2";

// splitContainer2.Panel1
//
this.splitContainer2.Panel1.Controls.Add(this.splitContainer1);

// splitContainer2.Panel2
//
this.splitContainer2.Panel2.Controls.Add(this.richTextBox2);
this.splitContainer2.Size = new System.Drawing.Size(1305, 908);
this.splitContainer2.SplitterDistance = 865;
this.splitContainer2.TabIndex = 5;

// saveFileDialog2
//
this.saveFileDialog2.DefaultExt = "png";
this.saveFileDialog2.FileName = "my_map.png";

// label12
//
this.label12.AutoSize = true;
this.label12.Location = new System.Drawing.Point(10, 456);
this.label12.Name = "label12";
this.label12.Size = new System.Drawing.Size(97, 13);
this.label12.TabIndex = 39;
this.label12.Text = "Raster Size [Pixels]";

// // numericUpDown6
this.numericUpDown6.Location = new System.Drawing.Point(130, 452);
this.numericUpDown6.Maximum = new System.Decimal(new int[] { 16, 0, 0, 0 });
this.numericUpDown6.Minimum = new System.Decimal(new int[] { 1, 0, 0, 0 });
this.numericUpDown6.Name = "numericUpDown6";
this.numericUpDown6.Size = new System.Drawing.Size(33, 22);
this.numericUpDown6.TabIndex = 40;
this.numericUpDown6.Value = new System.Decimal(new int[] { 8, 0, 0, 0 });

// // Form1
// this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.ClientSize = new System.Drawing.Size(1305, 908);
this.Controls.Add(this.splitContainer2);
this.Name = "Form1";
thisShown += new System.EventHandler(this.Form1_Shown);
((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
this.panell.ResumeLayout(false);
this.panell.PerformLayout();
this.splitContainer1.Panel1.ResumeLayout(false);
this.splitContainer1.Panel2.ResumeLayout(false);
this.splitContainer1.Panel2.PerformLayout();
((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).EndInit();
this.splitContainer1.Panel1.ResumeLayout(false);
this.splitContainer1.Panel2.ResumeLayout(false);
((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).EndInit();
this.splitContainer1.ResumeLayout(false);
((System.ComponentModel.ISupportInitialize)(this.numericUpDown5)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.numericUpDown4)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.numericUpDown3)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.numericUpDown2)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.numericUpDown1)).EndInit();
this.splitContainer2.Panel1.ResumeLayout(false);
(this.splitContainer2.Panel1.ResumeLayout(false);
this.splitContainer2.ResumeLayout(false);
private System.Windows.Forms.Label label11;
private System.Windows.Forms.TextBox textBox6;
private System.Windows.Forms.SaveFileDialog saveFileDialog2;
private System.Windows.Forms.Label label12;
35.4 Noise map (red = traffic noise, blue = running water noise)
36 Astronomy

36.1 What's the best time for moon observing?

That depends on the moon's phase as follows:

<table>
<thead>
<tr>
<th>Moon phase</th>
<th>Largest altitude:</th>
<th>When has the ecliptic the steepest angle to the horizon?</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Moon</td>
<td>(not during darkness)</td>
<td></td>
</tr>
<tr>
<td>Waxing crescent</td>
<td>(not during darkness)</td>
<td></td>
</tr>
<tr>
<td>First quarter</td>
<td>Spring, March 20, 18:00 o'clock</td>
<td></td>
</tr>
<tr>
<td>Waxing gibbous</td>
<td>Winter, February 5, 21:00 o'clock</td>
<td></td>
</tr>
<tr>
<td>Full moon</td>
<td>Winter, December 20, 0:00 o'clock</td>
<td></td>
</tr>
<tr>
<td>Waning gibbous</td>
<td>Autumn, November 5, 3:00 o'clock</td>
<td></td>
</tr>
<tr>
<td>Last quarter</td>
<td>Autumn, September 20, 6:00 o'clock</td>
<td></td>
</tr>
<tr>
<td>Waning crescent</td>
<td>(not during darkness)</td>
<td>Autumn, September 20, at sunset</td>
</tr>
</tbody>
</table>

The given dates are a rough estimate plus or minus one month, only valid for observers on the northern hemisphere.
36.2 Limiting magnitude for video astronomy

<table>
<thead>
<tr>
<th>Lens</th>
<th>Camera</th>
<th>Video mode</th>
<th>ISO</th>
<th>Limiting magnitude</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon EF 400mm f/2.8 + SpeedBooster 0.64x</td>
<td>Panasonic GH5S</td>
<td>FHD 25fps, [Ex. Tele Conv.] = 2.1x</td>
<td>25600</td>
<td>about 12.2 mag</td>
<td>Sky wasn't perfectly clear, with 4x contrast enhancement, no noise reduction</td>
</tr>
</tbody>
</table>

36.3 Limiting magnitude for stacked exposures

<table>
<thead>
<tr>
<th>Lens</th>
<th>Camera</th>
<th>Exposure</th>
<th>ISO</th>
<th>Limiting magnitude</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canon EF 400mm f/2.8</td>
<td>Canon 5D MK4</td>
<td>173 x 30s = 86.5 min</td>
<td>3200</td>
<td>about 18.5 mag</td>
<td>Stacked with DeepSkyStacker, sky wasn't perfectly clear</td>
</tr>
</tbody>
</table>

36.4 Useful calculations for Magnitudes

Convert magnitude \( m_V \) [mag] to illuminance \( E_V \) [Lux]:

\[
E_V = 10^{(-14.18 - M_V) / 2.5}
\]

Convert illuminance \( E_V \) [Lux] to magnitude \( m_V \) [mag]:

\[
M_V = -14.18 - 2.5 \times \log_{10} E_V
\]

Convert illuminance \( E_V \) [Lux] to irradiance \( E_\lambda \) [W/m^2], for wavelength 555nm:

512
\[ E_E = E_V / 683 \text{ lx/W} \]

Convert irradiance \( E_E \) [W/m\(^2\)] to illuminance \( E_V \) [Lux], for wavelength 555nm:
\[ E_V = E_E \times 683 \text{ lx/W} \]

### 36.5 Crab Pulsar

The crab pulsar is a supernova remnant and consists of a neutron star which is rapidly spinning with a frequency of about 30 Hz. It emits pulses in radio, visual, X-ray and gamma spectral range.

https://en.wikipedia.org/wiki/Crab_Nebula#Central_star

The frequency is slowly decreasing and the latest measurement results can be found here:
https://heasarc.gsfc.nasa.gov/W3Browse/all/crabtime.html

http://www.jb.man.ac.uk/~pulsar/crab/crab2.txt

In the book "Paul Horowitz, Winfield Hill: The Art of Electronics, Second Edition, Page 1030ff" is described how to measure the light curve with a 60" telescope, a photomultiplier and a signal averager. They used 5 million sweeps which is more than 41 hours of sampling time.

When averaging the signal, three effects must be considered:

1. The pulsar's frequency decreases by about 1.326µHz per hour.

2. The Doppler effect due to earth's rotation. The velocity of the observer is: \( V = 2 \times \pi \times 6370 \text{ km} / 86400 \text{ s} \times \cos(\text{latitude}) \)
   For 51.5° latitude the velocity is 0.288 km/s towards the east point of the local horizon.
   The Doppler frequency shift is \( f_0 = f \times V / c \) where \( c \) is the speed of light 300000 km/s.
   For \( f = 30 \text{ Hz} \) the Doppler frequency shift is 28.8µHz at the east horizon and -28.8µHz at the west horizon. The frequency shift is 0 at the meridian.
   Near the meridian the Doppler frequency shift decreases by 7.5µHz per hour.

3. The Doppler effect due to earth orbiting around the sun. The velocity of the observer is: \( V = 2 \times \pi \times 149.6 \times 10^6 \text{ km} / 365.25 \text{ d} / 86400 \text{ s} = 29.786 \text{ km/s} \)
   towards a point on the ecliptic which is about 90° west of the sun.
   The Doppler frequency shift is \( f_0 = f \times V / c \) where \( c \) is the speed of light 300000 km/s.
   For \( f = 30 \text{ Hz} \) the Doppler frequency shift is 99.3µHz on the ecliptic 90° west of the sun and -99.3µHz on the ecliptic 90° east of the sun. At
midnight in winter the pulsar is approximately 180° away from the sun, so that the Doppler frequency shift is small. Under these conditions the frequency shift decreases by 2.13µHz per hour.

Adding these three effects, the observed pulsar frequency decreases by about 11µHz per hour (valid only in winter at midnight, when the pulsar is in the south near the meridian).

An error $\Delta f$ in the reference frequency will produce after time $t$ a phase error $\Delta p$ with respect to the pulsar phase as follows: $\Delta p / 360° = t \cdot \Delta f$

Example: If the reference frequency is off by 10µHz, after 2 hours there will be a phase error of 25.92°.

The latest exact frequency of the pulsar is from December 15, 2019:

$f = 29.6122791665$ Hz - $0.000001326$ Hz / hour
DCF77 is a 77.5kHz longwave time signal transmitter in Germany. The time signal is coded with 59 short (100ms = "0") or long (200ms = "1") impulses as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start of minute, is always 0</td>
<td>29</td>
<td>Hours 1</td>
</tr>
<tr>
<td>1-16</td>
<td>Civil warning bits and other informations</td>
<td>30</td>
<td>Hours 2</td>
</tr>
<tr>
<td>17</td>
<td>CET: 0       CEST: 1</td>
<td>31</td>
<td>Hours 4</td>
</tr>
<tr>
<td>18</td>
<td>CET: 1     CEST: 0</td>
<td>32</td>
<td>Hours 8</td>
</tr>
<tr>
<td>19</td>
<td>Leap second announcement</td>
<td>33</td>
<td>Hours 10</td>
</tr>
<tr>
<td>20</td>
<td>Always 1</td>
<td>34</td>
<td>Hours 20</td>
</tr>
<tr>
<td>21</td>
<td>Minutes 1</td>
<td>35</td>
<td>Even parity over hours bits 29-35</td>
</tr>
<tr>
<td>22</td>
<td>Minutes 2</td>
<td>36-41</td>
<td>Day of month</td>
</tr>
<tr>
<td>23</td>
<td>Minutes 4</td>
<td>42-44</td>
<td>Day of week</td>
</tr>
<tr>
<td>24</td>
<td>Minutes 8</td>
<td>45-49</td>
<td>Month number</td>
</tr>
<tr>
<td>25</td>
<td>Minutes 10</td>
<td>50-57</td>
<td>Year within century</td>
</tr>
<tr>
<td>26</td>
<td>Minutes 20</td>
<td>58</td>
<td>Even parity over the date bits 36-58</td>
</tr>
<tr>
<td>27</td>
<td>Minutes 40</td>
<td>59</td>
<td>No impulse</td>
</tr>
<tr>
<td>28</td>
<td>Even parity over minute bits 21-2</td>
<td>0</td>
<td>Previous defined time is valid at the beginning of this impulse (which is always 0)</td>
</tr>
</tbody>
</table>
Schematic diagram of a DCF77 receiver which produces 2kHz beeps for recording with the GH5S camera:

Nr. 810054 from www.pollin.de

DCF77 Module

74LV132D

3.5mm stereo cable to GH5S

Left: Microphone
Right: DCF77

Electret condenser microphone

1.5V

VCC

1kΩ

Piezo

100Ω

10µF

100n

330kΩ

1nF

1µF

47kΩ

100Ω

VCC

2kΩ

2µF
This is an (almost) invisible lattice mast, made of 0.3mm carbon fiber rods. At the top of the 1400mm high mast is a small white ball with 8mm diameter. The purpose is to have a small motionless object fixed in space, as a target for inserting special effects into videos, for example for simulation of wormholes with a moving camera.

This is the C# program for calculating the lattice mast:

```csharp
using System;
using System.Globalization;
namespace Gittermast
{
    class Program
    {
        static void Main(string[] args)
        {
            CultureInfo invC = CultureInfo.InvariantCulture;
            Console.WriteLine("Lattice mast calculation:");
            Console.WriteLine("" Jimmy  ");
            double a = 200; // Base width 200mm
            double h = 0; // Height
            double maxh = 1200; // Maximum height 1200mm
            double alpha = (90.0 - 3.5) * Math.PI / 180; // Base angle
            double beta = 45.0 / 180 * Math.PI; // diagonal angle 45°
            double t = 0; // Total material length
            double dh, l;
            Console.WriteLine("Base width: " + String.Format(invC, "{0:F1}", a));
            Console.WriteLine("Base angle: " + String.Format(invC, "{0:F1}", alpha / Math.PI * 180) + "°");
            Console.WriteLine("Diagonal angle: " + String.Format(invC, "{0:F1}", beta / Math.PI * 180) + "°");
            Console.WriteLine("" Jimmy  ");
            while (h < maxh)
            {
                h = h + dh;
                l = dh * Math.Sqrt(1.0 + 1.0 / (Math.Tan(beta) * Math.Tan(beta)));
                a = a - 2.0 * dh / Math.Tan(alpha);
                t = t + 4.0 * (dh + l + a);
                Console.WriteLine("Diagonal: " + String.Format(invC, "{0:F1}", l));
                Console.WriteLine("Width: " + String.Format(invC, "{0:F1}", a));
                Console.WriteLine("Height: " + String.Format(invC, "{0:F1}", h));
                Console.WriteLine("" Jimmy  ");
            }
            Console.WriteLine("Total material length: " + String.Format(invC, "{0:F1}", t));
            Console.Read();
        }
    }
}
This is the result:

Lattice mast calculation:

<table>
<thead>
<tr>
<th>Diagonal</th>
<th>Width</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>266.5</td>
<td>176.9</td>
<td>188.5</td>
</tr>
<tr>
<td>235.8</td>
<td>156.5</td>
<td>355.2</td>
</tr>
<tr>
<td>208.6</td>
<td>138.5</td>
<td>502.7</td>
</tr>
<tr>
<td>184.6</td>
<td>122.5</td>
<td>633.3</td>
</tr>
<tr>
<td>163.3</td>
<td>108.4</td>
<td>748.7</td>
</tr>
<tr>
<td>144.5</td>
<td>95.9</td>
<td>850.9</td>
</tr>
<tr>
<td>127.8</td>
<td>84.9</td>
<td>941.3</td>
</tr>
<tr>
<td>113.1</td>
<td>75.1</td>
<td>1021.2</td>
</tr>
<tr>
<td>100.1</td>
<td>66.4</td>
<td>1092.0</td>
</tr>
<tr>
<td>88.5</td>
<td>58.8</td>
<td>1154.6</td>
</tr>
<tr>
<td>78.3</td>
<td>52.0</td>
<td>1210.0</td>
</tr>
</tbody>
</table>

Total material length: 16228.3

I don't show a picture of the invisible lattice mast here. It makes no sense because it's invisible. You couldn't see it :-(
Spherical Trigonometry and Rotation Matrices

See also [https://en.wikipedia.org/wiki/Rotation_matrix](https://en.wikipedia.org/wiki/Rotation_matrix)

This is the rotation matrix for yaw rotation:

\[
\begin{pmatrix}
\cos(\text{yaw}) & -\sin(\text{yaw}) & 0 \\
\sin(\text{yaw}) & \cos(\text{yaw}) & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

This is the rotation matrix for pitch rotation:

\[
\begin{pmatrix}
\cos(\text{pitch}) & 0 & \sin(\text{pitch}) \\
0 & 1 & 0 \\
-\sin(\text{pitch}) & 0 & \cos(\text{pitch})
\end{pmatrix}
\]

This is the rotation matrix for roll rotation:

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & \cos(\text{roll}) & -\sin(\text{roll}) \\
0 & \sin(\text{roll}) & \cos(\text{roll})
\end{pmatrix}
\]

This is the rotation matrix for yaw, pitch and roll rotations (in this order):

\[
\begin{pmatrix}
\cos(\text{yaw}) \times \cos(\text{pitch}) & \cos(\text{yaw}) \times \sin(\text{pitch}) \times \sin(\text{roll}) - \sin(\text{yaw}) \times \cos(\text{roll}) & \cos(\text{yaw}) \times \sin(\text{pitch}) \times \cos(\text{roll}) + \sin(\text{yaw}) \times \sin(\text{roll}) \\
\sin(\text{yaw}) \times \cos(\text{pitch}) & \sin(\text{yaw}) \times \sin(\text{pitch}) \times \sin(\text{roll}) + \cos(\text{yaw}) \times \cos(\text{roll}) & \sin(\text{yaw}) \times \sin(\text{pitch}) \times \cos(\text{roll}) - \cos(\text{yaw}) \times \sin(\text{roll}) \\
-\sin(\text{pitch}) & \cos(\text{pitch}) \times \sin(\text{roll}) & \cos(\text{pitch}) \times \cos(\text{roll})
\end{pmatrix}
\]

Angular distance between two stars:

\[
\cos(\text{distance}) = \cos(\alpha_1 - \alpha_2) \times \cos(\delta_1) \times \cos(\delta_2) + \sin(\delta_1) \times \sin(\delta_2)
\]

For small distances the above formula is not recommended, because the cos terms are near 1. In this case it's better to use this approximation:

\[
\text{Distance}^2 = (\alpha_1 - \alpha_2)^2 + (\delta_1 - \delta_2)^2
\]
My To Do List

-- Find out how to compile FFmpeg with options (difficult...)
-- Learn how Git works (difficult...)
-- Find out how to process videos with DaVinci Resolve
  -- Sync audio and video
  -- Edit
-- Deep zoom in
-- I know that the Windows scripts in this book need some modifications if they are to be used on MAC or Linux computers. But I have no experience with MAC and Linux. Who wants to write a short chapter about the differences of Windows/MAC/Linux scripts? I'd like to add it to this book, of course with proper credit to the author.
-- Stitching 360° videos from two or more cameras
-- How to show different videos synchronously on multiple beamers, for example in a dome where each beamer illuminates only a segment of the dome?
41 List of Abbreviations

ALL-I = Intraframe compression method, which saves all frames as intraframes (= I-frames = keyframes)
CRF = Constant Rate Factor
DAR = Display Aspect Ratio = the aspect ratio of the image
DR = Dynamic Range
ETC = Extra Tele Conversion
ETTL = Expose To The Left
ETTR = Expose To The Right
EV = Environment Variable
FPS = Frames Per Second
GOP = Group of Pictures
HDR = High Dynamic Range
HEVC = High Efficiency Video Coding
HLG = Hybrid Log Gamma
IPB = Interframe compression method, which saves three types of frames: Intraframes (I-frames), predicted frames (P-frames) and bi-directional predicted frames (B-frames)
IRE = a measurement of composite video signals, 100 IRE = the difference between black and white level
LHS = Left Hand Side
LUT = Look-up-Table
NR = Noise Reduction
PQ = Perceptual Quantization
RHS = Right Hand Side
SAR = Sample Aspect Ratio = the aspect ratio of the pixels
SDR = Standard Dynamic Range
SNR = Signal to Noise Ratio
VFR = Variable Frame Rate
VO = Voice-Over
Acknowledgements

I found many of the FFmpeg hints and examples somewhere in the internet. Thanks to all who posted them!

Thanks to all who contributed to the great FFmpeg software, and special thanks to Carl Eugen Hoyos, Paul B Mahol, Moritz Barsnick, Roger Pack and Gyan Doshi who answered many questions on the FFmpeg user mailing list.

Thanks to Dan Bridges for some sample scripts for FFmpeg.

Thanks to Bill Gray for writing the best astronomy software (Guide 9.1), and to Jost Jahn and Larry Wood for help with Guide 9.1.

Thanks to all who contributed to DeepSkyStacker software.

Thanks to all contributors to OpenStreetMap, OpenTopoMap, OpenDEM and Sergey Bobrovsky and Vitaliy Shibaev of BitMiracle for the LibTiff.net library.
Index

0-padded......................................................232
0x7..............................................................333
0xAABBGRR................................................258
0xRRGGBA................................................258
1.4x Teleconverter.................................393
1/4 wave antenna..............................359
1/90000s clock................................325
1> log.csv..........................................................242
1> measured.csv..............................166
10 bit video.......................................347
10-bit accuracy in filtergraphs.............240
10-bit compatibility.........................240
10-bit test video..............................240
10-bit video.......................................63, 278
10-bit VLog test video.......................382
16-Bit.........................................................243
16-bit GeoTiff......................................482
16-bit PNG...........................................244
18% gray...............................................380
1kHz tone..............................................239
2.1x Telescopic effect.........................392
2.5mm connector................................391
2k..............................................................246
2kdc1..........................................................246
2kflat...........................................................246
2x Teleconverter.................................393
2x2 mosaic.................................................53
3.5mm headphone output.......................310
3.5mm input jack................................310
3.5mm stereo connector.........................310, 421
360° Panoramic camera..........................405
3D Look-up table.................................236
3D LUT Creator........................................337
3x3 neighborhood...............................119
4:2:0.........................................................219
4:2:2.........................................................219
4:4:4.........................................................219
4K..............................................................246
4K..............................................................246
4kdc1.........................................................246
4kflat.........................................................246
5.1 channel layout.............................415
7artisans (Viltrox) 7.5mm..................366
Abbreviations, GH5S..........................521
abs..............................................................151, 276
Acknowledgements............................522
acos.............................................................276
Acoustic feedback..................................351
acrossfade............................................292, 294
acrossto.....................................................283
addition (blend filter)............................57
adelay......................................................218, 320
Adobe Digital Negative....................243
Adobe DNG Converter......................243, 331
adrawgraph.............................................262
aegisub.....................................................271
aeval.......................................................299, 407
aevalsrc..................................................32, 303, 314, 316, 415
affftilt.....................................................283, 311, 318
afreqshift...............................................304f.
ALL-I......................................................373, 521
alpha_mask.............................................138, 164, 170
alphaextract.............................................81
alphanumeric.................................81
Alternating left/right stereo sound.........321
altitude of the sun................................438
ambisonic sound...............................412
ambisonic test tone............................409
Ambisonics............................................407
amerge.....................................................300
amix......................................................283, 308
Amplifier.................................................300
amplify...................................................27, 283
amplitude.............................................303, 310
Anamorphic.........................................364
Anamorphic 10 bit Modes GH5S..........400
Anamorphic 8 bit Modes GH5S..........399
and (blend filter)....................................57
Animated GIF........................................54f.
animation..............................................229
anisodesrc............................................318
ANSI...........................................................41
anullsrc...............................................45, 297f.
apad......................................................303, 320
Aperture...............................................366
Aperture number..................................370
Append text.........................................334
Apply LUT (OBS)....................................355
Apprehension Engine.........................424
apt-get install ffmpeg.......................361
aresample...........................................247, 295f.
areverse.................................................32
ARIB STD-B67.........................................384
Artefacts...............................................119
hypot...58f., 121, 131ff., 137, 144, 150, 161, 163, 276
i-frame......................................................521
Identity *.cube LUT........................................68
IDS.........................................................355
if..............................................................235, 276
if (in batch file)............................................335
if %MODE%==1.............................................335
Illuminance................................................416f., 512
Image circle diameter..................................366
Image formats............................................243
Image processing devices.............................355
Image warping with displacement filter...........211
in_time.....................................................35
Insert text..................................................40
Integer arithmetic in Windows batch file...........20
Interframe compression...............................521
Internal Serial Number (ExifTool)....................330
Interpolation band.....................................145
Interval.....................................................434
Interviewing two people................................360
Intraframe..................................................521
Intraframe compression...............................521
Introduction to FFmpeg................................7, 14
Invert function..........................................122
Invert channels.........................................24
Inverting a video........................................24
Invisible Lattice Mast...................................517
IPB..........................................................373, 521
IRE..........................................................379, 381, 521
IrfanView..................................................244, 331
Irradiance..................................................512
Irregular quadrangle...................................209
it...............................................................35
iZugar MKX200-ASPH 3.8mm..........................367
iZugar MKX22 3.25mm..................................367
Jitter.........................................................117
join..................................................................300, 321
JPG............................................................243
JPG Test images..........................................238
Keybroad.....................................................428
Keyframe....................................................521
Kodak PIXPRO SP360 4K camera...75, 128, 404
Kodak SP360_4K.........................................248, 363
Korven, Mark..............................................424
L4500 video lights.......................................416
lagfun.........................................................73f., 79, 289
Lagfun filter..................................................71
LAN cable...................................................359
language=ger............................................271
Laowa 24mm f/14.........................................393
Laowa 4mm..................................................367
Latitude.....................................................481
Lattice mast...............................................517
ld............................................................117f., 121, 161, 211, 276, 315
Leica DG 12-60mm f/2.8-4.0.............330, 393, 435f.
Lens Serial Number (ExifTool)......................330
lenscorrection.............................................120
lerp..........................................................142, 161, 172, 277f., 287
Let system position window.........................115
LFE...........................................................412
LFO...........................................................430
LHS...........................................................521
libavutil.....................................................220
libswscale...............................................225
LibTiff......................................................483
libx264.......................................................228
libxvid.......................................................228
Licence......................................................312
Light curve...............................................257, 259
light value...............................................438
lighten (blend filter).................................57
lime..........................................................233f.
Limit the length..........................................54
Limiting magnitude......................................512
LINE GHSS...............................................423
Line of sight..............................................480
Linear chirp...............................................315
Linux.........................................................7
Linux (OBS)...............................................361
Linux script file.........................................361
List of all pixel formats.............................221
Little planet..............................................132, 162
Live broadcasting.......................................358
Live ultrasound conversion..........................310
log...........................................................277
long persistence time..................................71
Longer Exposure Time GH5S.........................389
Longitude...................................................481
loop..........................................................84
lossless.....................................................243
Low Frequency Oscillator............................430
Low-noise.................................................322
Iowpass.....................................................303, 310
lt.............................................................234, 277, 316
Ite...........................................................277
lum_expr....................................................142
Luminance Level GH5S..................................386, 388
LUT..........................................................338, 521
LUT (OBS)..................................................355
LUT_3D_SIZE............................................67, 236
lut3d.........................................................237, 339, 348
lut3d="%LUT%"............................................348
lux...........................................................416
Lux..........................................................512
M42..........................................................370
mag..........................................................512
Magnitude...............................................512
make........................................................361
Manual GHSS............................................378
map_metadata............................................228, 349
Vertical border ........................................... 142
Vertical flipping ........................................... 52
Vertical jitter effect ........................................... 118
veryfast ........................................... 228
veryslow ........................................... 228
vflip ........................................... 52, 472
VFR ........................................... 521
vga ........................................... 246
vibrato ........................................... 291
Video astronomy ........................................... 512
Video Capture Device ........................................... 353
Video Codescs ........................................... 228
Video format ........................................... 14
Video line jitter ........................................... 117
Video Modes GH5S ........................................... 394
Video resolution ........................................... 364
Video Size GH5S ........................................... 388
Video sizes ........................................... 246
Video-in-Video ........................................... 209
vidstabdetect ........................................... 181
vidstabtransform ........................................... 181
View through eyepiece ........................................... 58
Viewing direction ........................................... 136
vignetted ........................................... 62
Vignetting ........................................... 62
VLC player ........................................... 7, 355, 361
VLC Player ........................................... 336
VLOG ........................................... 67
VLog table ........................................... 381
VLOG-L ........................................... 379, 386
VLOG-L curve ........................................... 383
VLog-L to Rec709 LUT ........................................... 348
VO ........................................... 521
Voice-Over ........................................... 307, 351
Voltage Controlled Amplifier ........................................... 430
Voltage Controlled Filter ........................................... 430
Voltage Controlled Oscillator ........................................... 430
Volume ........................................... 254, 294, 301, 303, 321
Volume knob ........................................... 300
volume=0:enable= ........................................... 321
volume=enable ........................................... 297
volumedetect ........................................... 320
vstack ........................................... 52, 77, 237, 262
W/m^2 ........................................... 512
Waveform Monitor ........................................... 380
Webcam ........................................... 353
while ........................................... 278
Wiki ........................................... 7
Wildcards ........................................... 332
win_size ........................................... 318
Wind noise ........................................... 301, 480
Window Capture ........................................... 353
Windows ........................................... 7
Windows taskbar ........................................... 148
Wormhole ........................................... 165, 517
Wormhole simulation ........................................... 162
Write text on a transparent layer ........................................... 48
wuxga ........................................... 246
X-Rite ColorChecker ........................................... 337
xfade ........................................... 33, 291
xga ........................................... 246
xmap ........................................... 127
xor (blend filter) ........................................... 57
xstack ........................................... 52f, 65, 252
Y'CbCr ........................................... 219
yaw ........................................... 123, 163, 172
ymap ........................................... 127
YouTube Live ........................................... 280, 358
Youtube recommended settings ........................................... 279
Yumiki 2.5mm ........................................... 367
yuv420p ........................................... 220, 278
yuv420p10le ........................................... 220
yuv422p ........................................... 220
yuv422p10le ........................................... 220
-q:v ............................................................ 12
-r .......................................................... 15, 54, 279
-r:a .......................................................... 279
-re .............................................................. 307
-rtbufsize ................................................ 268
-s ............................................................. 302, 329
-sar .......................................................... 268
-shortest ........................................ 12, 52, 297f., 308
-show_entries ...................................... 166, 242
-show_region ........................................ 266
-ss ............................................................ 294, 302, 329
-sseof ...................................................... 21, 31
-start_number ...................................... 12, 18, 150, 170, 232, 238
-strftime .................................................. 232
-.. .......................................................... 294, 339
-fo ............................................................ 28
-top .......................................................... 179, 293
-tune .......................................................... 229

-v debug .................................................. 144
-v verbose ........................................ 144, 225
-vcocodec ........................................ 14, 83, 218
-vcodec (as an input option) .................. 269
-vf ........................................................... 22, 329
-video_size ...................................... 83, 218, 265f., 268
-vn ........................................................... 329
-window_x ............................................... 83f.
-window_y ............................................... 83f.
-y ............................................................. 21, 338
: .............................................................. 10
:: .......................................................... 10
> (Windows batch file) ........................... 334
>> (Windows batch file) ......................... 334
| character .............................................. 174

*.srt ..................................................... 270
#!/bin/bash ............................................. 361
% character ............................................ 10
%~ .......................................................... 332
%0nd ...................................................... 232
%1 .......................................................... 232
%d ........................................................... 327
%H ........................................................... 232
%m ........................................................... 232
%M ........................................................... 232
%nd ...................................................... 232
%S ........................................................... 232
%Y ........................................................... 232

*.ass ..................................................... 272
*.ass subtitles ....................................... 42
*.bat .......................................................... 10
*.cmd ...................................................... 177
*.cube ..................................................... 63, 67f., 70, 236

536