

An Introduction to
FFmpeg, DaVinci Resolve, 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, Image Processing and Astronomy Software

by

Michael Koch, astroelectronic@t-online.de

Version from July 21, 2022

Hint: The table of contents is quite long and it may be difficult to find what you are looking for. Try the index at the end of the book!

Contents

1	Introduction to FFmpeg	10
1.1	What can be done with FFmpeg?	12
1.2	If FFmpeg has no graphical user interface, how do we use it?	13
1.3	The first example	15
1.4	Using variables	16
2	FFmpeg in detail	17
2.1	Convert from one video format to another video format	17
2.2	Change the container format	17
2.3	Fit timelapse length to music length	18
2.4	Timelapse or slideshow from many images, with crossfading	19
2.5	Slideshow with different durations	21
2.6	Slideshow with scrolling images	22
2.7	Extract many images from a video	25
2.8	Extract the nth frame from a video	26
2.9	Extract the first and last frame from a video	26
2.10	Modify brightness, contrast, saturation, gamma and hue	27
2.11	Strong contrast enhancement	28
2.12	Inverting a video or image (make a negative)	30
2.13	Colorchannelmixer filter	32
2.14	Shuffleplanes filter	34
2.15	Day-for-Night	35
2.16	Colorcorrect filter	36
2.17	Colortemperature filter	38
2.18	Hue filter	40
2.19	Huesaturation filter	42
2.20	Selectivecolor filter	44
2.21	Colorbalance filter	46
2.22	Colorcontrast filter	48
2.23	Monochrome filter	50
2.24	Vibrance filter	54
2.25	Swapuv filter	57
2.26	Gradation curves	58
2.27	Color grading with color look-up tables, full workflow	59
2.28	Color grading with color look-up tables, simplified workflow	61
2.29	Size of color-look-up tables	62
2.30	Convert a *.cube LUT to a halclut file	66
2.31	Colorchart source	67
2.32	Colormap filter	69
2.33	Colorhold, chromahold and hsvhold filters	72
2.34	Atmospheric dispersion correction	74
2.35	Amplify filter	75
2.36	Sharpen or blur images	75
2.37	FFT Filtering	77
2.38	Extract a time segment from a video	84
2.39	Trim filter	85
2.40	Tpad filter, add a few seconds black at the beginning or end	86
2.41	Extract the last 30 seconds of a video	87
2.42	Fade-in and fade-out	88
2.43	Crossfading	89
2.44	Crop a video	90
2.45	Add borders to a video	91
2.46	Zoompan	91
2.47	Changing the speed: slow motion and timelapse	92
2.48	Slow motion or timelapse only for a segment of the video	93
2.49	Time Remapping	94
2.50	Insert a text which is visible for the whole duration	96
2.51	Slowly fade a text in and out	96
2.52	Vertical alignment of text	98
2.53	Show a running clock in the video	99
2.54	Generation of curved text for fulldome projection	101
2.55	Write text on a transparent layer	104
2.56	Combine multiple videos with concat demuxer	106
2.57	Combine multiple videos with concat filter	108
2.58	The "fps" filter	108
2.59	Split a video in multiple segments	109
2.60	Switch between two cameras, using audio from camera1	110
2.61	Stack videos side by side (or on top of each other)	111
2.62	Horizontal and vertical flipping	111

2.63	Stereo3d filter.....	111
2.64	Stack four videos to a 2x2 mosaic	112
2.65	Blink comparator.....	113
2.66	Creating animated GIF or PNG.....	115
2.67	Overlay an animated GIF over a video.....	116
2.68	Changing the size of an animated GIF.....	116
2.69	Replace one frame in a video by another.....	117
2.70	Blend filter.....	118
2.71	Circular mask (View through eyepiece).....	120
2.72	Radial attenuation (for overhead sky in a planetarium).....	122
2.73	Edge Attenuation.....	125
2.74	Binocular view simulation.....	126
2.75	Vignetting.....	127
2.76	Subtracting a darkframe.....	128
2.77	Histogram.....	128
2.78	Lagfun filter.....	129
2.79	Deflicker a video.....	131
2.80	Star trails.....	134
2.81	Bird trails.....	135
2.82	Rainbow-trail effect.....	138
2.83	Temporal slice-stacking effect.....	139
2.84	Extract and merge planes, split planes.....	140
2.85	Extract the alpha channel.....	141
2.86	Colorkey.....	142
2.87	Chromakey.....	143
2.88	HSVkey.....	144
2.89	Lumakey.....	145
2.90	Bluescreen / greenscreen.....	146
2.91	Real-time bluescreening	157
2.92	Extract moving objects from a video.....	159
2.93	Datascope.....	160
2.94	Video line jitter effect.....	161
2.95	Vertical jitter effect.....	162
2.96	CRT Screen Effect.....	163
2.97	Macroblock motion vectors.....	163
2.98	Deblock filter.....	163
2.99	Gradfun filter.....	164
2.100	Dilation filter.....	164
2.101	Morphological transforms.....	165
2.102	Correct the radial distortion of (fisheye-) lenses.....	168
2.103	Lensfun.....	172
2.104	Replace the lensfun filter by the remap filter.....	176
2.105	Deep-Zoom-In.....	179
2.106	V360 filter for rotation of equirectangular 360° videos.....	180
2.107	Equirectangular images of the night sky.....	184
2.108	Equirectangular images of the earth and planet surfaces.....	184
2.109	Undo spherical rotations.....	185
2.110	Remap a fisheye video to an equirectangular video.....	186
2.111	Remap an equirectangular video to a fisheye video.....	190
2.112	Realtime remapping from equirectangular to fisheye	192
2.113	How to replace the v360 filter by the remap filter.....	196
2.114	Off-center fisheye projection.....	209
2.115	DLP Beamer output.....	214
2.116	Remap from equirectangular to double-fisheye.....	215
2.117	Remap an equirectangular video to a "Little planet" video....	216
2.118	Remap an equirectangular video to a "Mirror sphere" video..	218
2.119	Shifting the viewing direction in a fisheye image or video.....	222
2.120	How the "drawbox" filter works.....	224
2.121	Stitching together double-fisheye videos.....	225
2.122	Remove stitching artefacts.....	227
2.123	Stitch double-fisheye images with alignment errors.....	231
2.124	Preprocessing a flat video for fulldome projection.....	235
2.125	Rotating the earth, moon or planet.....	238
2.126	Live rotation of the moon.....	244
2.127	Insert a crosshair in a live video.....	246
2.128	Enlarge the corners of a live video.....	247
2.129	Tunnel effect.....	249
2.130	Black hole simulation with remap filter.....	253
2.131	Wormhole simulation.....	259
2.132	Realtime Wormhole in Planetarium (Hackathon 2022).....	264
2.133	Spherical Transformations with two Inputs.....	276
2.134	Simulation of a moving wormhole.....	282
2.135	Dynamically change commands with "sendcmd" or "zmq"....	286
2.136	Sendcmd and commands.....	287
2.137	Sending commands with ZMQ.....	292
2.138	Changing an input image while FFmpeg is running.....	294

2.139	FFmpeg and C# Programming.....	295	2.177	Pseudocolor filter.....	361
2.140	Video stabilization.....	298	2.178	Test a filtergraph for 10-bit compatibility.....	362
2.141	Stabilization of 360° Videos.....	301	2.179	Make a 10-bit test video with audio.....	366
2.142	Stabilization of 360° Videos, improved.....	306	2.180	Find an object in a video and hide it.....	366
2.143	Remap Video-in-Video with perspective filter.....	308	2.181	Feedback filter.....	367
2.144	Image warping with displace filter.....	310	2.182	Find the coordinates of a moving object in a video	368
2.145	Noise reduction.....	315	2.183	Detect black frames and replace them by previous frame.....	369
2.146	-filter_complex script.....	316	2.184	Image formats.....	370
2.147	Time delay within a filter chain.....	316	2.185	Image size limits.....	371
2.148	Recording one-time events with pre-trigger	317	2.186	CR2 and DNG Images.....	372
2.149	Chroma subsampling, pixel formats of images or videos.....	318	2.187	Colorspace.....	374
2.150	Automatic format conversions.....	325	2.188	Video sizes.....	375
2.151	RGB Color Cube.....	327	2.189	Editing videos from PanoView XDV360 fulldome camera.....	378
2.152	Convert RGB to HSV or HSL.....	331	2.190	Editing videos from the Kodak SP360_4K camera.....	379
2.153	Convert HSV to RGB.....	332	2.191	Postprocessing of real time videos of the night sky.....	380
2.154	Video Codecs.....	333	2.192	Workflow for night sky videos with GH5S.....	383
2.155	Bitrate.....	334	2.193	Combine many options and filters.....	384
2.156	Keyframes.....	335	2.194	Timelapse example with masking, deflicker, rotation, fading.....	385
2.157	Open GOP and closed GOP.....	335	2.195	Slow motion with Panasonic GH5S at 240fps.....	386
2.158	Video codecs with alpha channel.....	336	2.196	Create a light curve of a star occultation.....	388
2.159	Metadata.....	337	2.197	Oscilloscope.....	394
2.160	Video filters "copy" and "null".....	338	2.198	Vectorscope.....	394
2.161	Re-numbering images.....	338	2.199	Capture a video from a webcam.....	395
2.162	Filenames for images.....	339	2.200	Capture video from the desktop or from a window.....	396
2.163	Create two-dimensional gradients.....	340	2.201	Capture video and audio from a HDMI to USB converter.....	398
2.164	Make spectrum images.....	341	2.202	Adding *.srt subtitles to a video.....	402
2.165	Radial Gradients.....	352	2.203	Adding *.ass subtitles or credit files to a video.....	404
2.166	Make an additive color mixing image.....	353	2.204	Removing subtitles.....	405
2.167	Make a test image with shifted RGB channels.....	354	2.205	frei0r.....	406
2.168	Make a 8-color test image.....	355	2.206	Mapping streams to the output (-map).....	406
2.169	Redshift.....	356	2.207	Creating multiple outputs.....	406
2.170	SMPTE Color Bars.....	358	2.208	Attaching a preview image to a video.....	407
2.171	Make many JPG test images.....	358	2.209	Attach Cover Art to a Video.....	408
2.172	Make a grid video.....	359	2.210	Timeline support.....	409
2.173	Make a chessboard video.....	359	2.211	Expression evaluation.....	420
2.174	Make a test video with audio.....	359	2.212	Evaluation of mathematical functions.....	423
2.175	Make a noise video.....	360	2.213	Error handling in expressions.....	424
2.176	Make a grayscale test image.....	360	2.214	Debugging of expressions.....	425

2.215	List the R,G,B values as ASCII text file.....	427
2.216	Downloading videos from Facebook.....	429
2.217	Uploading spherical 360 images to Facebook.....	429
2.218	Uploading videos to Facebook.....	430
2.219	Downloading videos from Youtube.....	430
2.220	Youtube recommended settings.....	431
2.221	Streaming from FFmpeg to YouTube Live.....	432
2.222	Limiting FFmpeg's encoding speed to realtime.....	432
2.223	Point-to-point streaming.....	433
2.224	Streaming a video to a multicast address.....	435
2.225	RTMP Streaming.....	437
2.226	Hardware acceleration.....	438
2.227	OpenCL Hardware Acceleration.....	439
2.228	Benchmarks.....	440
2.229	FFmpeg console output.....	440
2.230	FFmpeg source code.....	441
2.231	FFmpeg: Suggestions for improvement.....	442
3	Audio processing with FFmpeg.....	462
3.1	Concat two or more audio files.....	462
3.2	Combine multiple audio files with crossfading.....	462
3.3	Change audio volume.....	463
3.4	Change audio sample rate and number of audio channels.....	463
3.5	Change audio length, sample rate and/or pitch	465
3.6	Add one or more sound effects to an audio track.....	466
3.7	Replace a segment of the audio stream by silence.....	467
3.8	Add an audio stream to a video, if no audio stream exists.....	467
3.9	Stereo --> mix into one mono channel.....	468
3.10	Check if both stereo channels are equal.....	469
3.11	Check if two mono inputs are equal.....	469
3.12	Extract one mono channel from stereo.....	469
3.13	Stereo --> two mono channels.....	470
3.14	Use only one channel of a stereo signal.....	470
3.15	Mono --> stereo.....	471
3.16	Two mono channels --> stereo.....	471
3.17	Mix two stereo channels to one stereo channel.....	472
3.18	Create a file with multiple audio streams.....	472
3.19	How to choose the correct audio volume level.....	473
3.20	Remove low frequencies (wind noise) from an audio track.....	473
3.21	Remove silence from an audio stream	474
3.22	Make a video from an audio file.....	474
3.23	Convert ultrasound to the audible range, e.g. to hear bats.....	475
3.24	Record sound with the computer's built-in microphone.....	478
3.25	Record a "Voice-Over" audio track.....	479
3.26	Passing the FFmpeg output to FFplay.....	481
3.27	Record sound and pass the output to FFplay	481
3.28	Record, process and play sound with FFplay.....	481
3.29	Live ultrasound conversion.....	482
3.30	Extract the audio from a video.....	484
3.31	Split a video into audio-only and video-only.....	484
3.32	Synchronize audio with video.....	484
3.33	Sources for royalty-free music.....	485
3.34	Sound effects, from frequency domain to time domain.....	486
3.35	Create a Quadrature Signal.....	492
3.36	Gravitational waves from binary black hole mergers.....	493
3.37	Create band-limited noise.....	497
3.38	Show the frequency response of a filter.....	498
3.39	Make an audio file with a short test tone.....	499
3.40	Measuring the audio volume.....	499
3.41	Convert an audio waveform to a picture.....	500
3.42	Optical sound effect.....	506
3.43	Which equipment is useful for making sound records?.....	507
3.44	Create an alternating left/right stereo sound.....	508
3.45	The "avsynctest" source.....	508
3.46	Comparison of Rode NT1 and NTG2 microphones.....	509
3.47	Mathematical properties of sample rates 44100 and 48000.....	510
3.48	Speed of sound.....	512
4	FFprobe.....	513
4.1	Count the number of frames.....	514
4.2	Find the keyframe timestamps.....	514
5	FFplay.....	515
6	DaVinci Resolve 15 / 16 / 17.....	518
6.1	Tutorials on Youtube.....	519
6.2	Mouse buttons and keyboard shortcuts.....	521
6.3	GUI Controls.....	523
6.4	Preferences.....	524
6.5	Project settings.....	525

6.6	Timeline Proxy Mode and Timeline Resolution.....	526
6.7	The "Media" page in DaVinci Resolve.....	527
6.8	Archive projects.....	528
6.9	The "Cut" page in DaVinci Resolve.....	529
6.10	The "Edit" page in DaVinci Resolve.....	530
6.11	Speed Change.....	534
6.12	Change the framerate of a clip.....	534
6.13	GUI Controls in the "Edit" page.....	535
6.14	Markers.....	535
6.15	Edit the audio volume of a clip.....	536
6.16	Normalize Audio.....	536
6.17	The "Fusion" page in DaVinci Resolve.....	537
6.18	Add a new "MediaIn" node.....	538
6.19	Fusion Clips.....	539
6.20	The "Color" page in DaVinci Resolve.....	540
6.21	Corrector nodes in "Color" page.....	542
6.22	Secondary Adjustments.....	543
6.23	Save Color Gradings.....	543
6.24	Wipes.....	544
6.25	The "Fairlight" page in DaVinci Resolve.....	544
6.26	The "Deliver" page in DaVinci Resolve.....	545
6.27	Day-for-Night.....	546
6.28	Dynamic Zoom.....	546
6.29	Synchronize audio with video.....	547
6.30	Cutting on the beat of the music.....	547
6.31	Video stabilization.....	548
6.32	Processing of drone videos.....	549
6.33	Track an object.....	550
6.34	Track and blur (or pixelize) an object.....	553
6.35	Track an object and cover it by a colored rectangle.....	554
6.36	Track an object and cover it by an image.....	555
6.37	Track an object and corner pin an image.....	556
6.38	Isolate Humans with "Magic Mask".....	556
6.39	Overlay text.....	557
6.40	Color matching.....	558
6.41	Generate a LUT.....	559
6.42	Apply a look-up-table to a clip.....	560
6.43	Spline Editor.....	560
6.44	Special effects.....	561
6.45	Alpha masking.....	565
6.46	Qualifier.....	566
6.47	Exclude two regions from a mask.....	567
6.48	Snowfall.....	568
6.49	Polyline toolbar.....	569
6.50	Artificial lightning.....	570
6.51	Adjustment clips.....	572
6.52	Trails.....	573
6.53	Fog Effect.....	574
6.54	Explosion effect.....	575
6.55	Bluescreen / greenscreen.....	577
6.56	3D Effects.....	578
6.57	3D Water Effect.....	579
6.58	3D Camera Tracking.....	580
6.59	3D Plant Models.....	584
6.60	Resolve Live.....	584
6.61	Time remapping.....	585
6.62	Slow motion and timelapse.....	585
6.63	Reducing the framerate (Super 8 simulation).....	585
6.64	Noise reduction.....	586
6.65	Reverse a video.....	587
6.66	Recording a Voice-Over audio track	588
6.67	ADR (Automated Dialog Replacement).....	590
6.68	VR360.....	591
6.69	Reframing 360° videos with KartaVR.....	592
6.70	Little Planet.....	594
6.71	Spherical stabilizer.....	596
6.72	Mirror sphere.....	598
6.73	Black hole effect.....	600
6.74	Wormhole effect.....	602
6.75	Correcting Lens Distortion.....	602
6.76	Programming fuses.....	603
6.77	Multicam Editing.....	603
6.78	Subtracting a darkframe.....	603
6.79	Subtitles.....	604
6.80	Supported Codecs.....	605
6.81	Convert *.mkv videos for DaVinci Resolve.....	605

6.82	Convert 10-bit videos from GH5S for DaVinci Resolve	606
6.83	Metadata	609
6.84	Sound library	609
6.85	Blackmagic RAW speed test	610
6.86	DaVinci Resolve, Problems and Solutions	610
6.87	Miscellaneous unsorted things	612
7	Shotcut	612
8	Exiftool	613
9	IrfanView	615
9.1	Create an image with a transparent color	615
10	Gimp	616
11	Faststone Image Viewer	616
12	Adobe DNG converter	616
13	Batch files (DOS, Windows 7 and 10)	617
13.1	Wildcards in filenames	617
13.2	Create beeps in a batch file	618
13.3	Loop over all files in a directory	619
13.4	Create short text files or append text to a file	619
13.5	Calculate variables in a batch file	619
13.6	if conditions	620
13.7	Start a new process	620
14	Batch files (Unix, Linux)	620
15	VLC Player	621
15.1	Show fullscreen video on the extended desktop	622
16	MPV	622
17	CD Rippers	624
18	DVD Rippers	624
19	SpyderX Elite, Monitor calibration	624
20	Color grading with 3D LUT Creator	625
20.1	Known problems and solutions	628
20.2	Beginner tutorials for 3D LUT Creator	630
20.3	Advanced tutorials for 3D LUT Creator	631
20.4	Working with Color Targets in for 3D LUT Creator	632
20.5	Working with video in for 3D LUT Creator	632
21	OBS - Open Broadcaster Software	634
21.1	OBS - Problems and solutions	635
21.2	CPU Load	636
21.3	Facebook Live	637
21.4	Facebook Live - Mode "Camera"	637
21.5	Facebook Live - Mode "Connect"	638
21.6	YouTube Live	639
21.7	Desktop for live broadcasting	640
21.8	Live broadcasting, or recording in advance?	640
21.9	Using the same source multiple times	641
21.10	Studio Mode	641
21.11	Virtual Camera	641
22	Tips and tricks for video	642
22.1	Neutral Density Filters	643
22.2	Shutter Angle	644
22.3	Panning Speed	645
23	Screenwriting	646
24	Unix (Ubuntu), compiling FFmpeg	647
24.1	GIT	649
24.2	Compiling FFmpeg under Windows	651
25	Cameras and lenses for fulldome video production	652
25.1	Pixel Size in Fulldome Planetariums	653
25.2	Read-out chip size of cameras at different video modes	654
25.3	Overview of available fisheye lenses	655
25.4	Favorable camera / fisheye combinations	657
25.5	Fisheye projection lenses	659
25.6	Non-fisheye projection lenses	660
25.7	Beamers for fulldome projection	661
25.8	Flange distances	662
25.9	Aperture numbers, rounded and exact	662
25.10	Test patterns for fulldome projection	663
26	Canon 5D-Mark4	666
26.1	All Canon 5D-Mark4 video modes for PAL video system	666
26.2	All Canon 5D-Mark4 video modes for NTSC video system	667
26.3	Canon 5D-Mark4 Field of view	668
26.4	Video tutorials for Canon 5D-Mark4	670
27	Panasonic LUMIX GH5S	674
27.1	GH5S Autofocus	674
27.2	GH5S Record formats	675
27.3	GH5S Exposing for VLog-L	676
27.4	GH5S HLG (Hybrid Log Gamma)	681
27.5	GH5S Metering Modes	683

27.6	GH5S Recommended settings	684
27.7	GH5S Custom settings C1, C2, C3-1, C3-2, C3-2	685
27.8	GH5S Luminance level	686
27.9	GH5S Master pedestal level	686
27.10	GH5S Video size	686
27.11	GH5S Mechanical / electronic shutter	687
27.12	GH5S Longer exposure time than framerate allows	687
27.13	GH5S Variable frame rate	688
27.14	Recording duration on SD cards	689
27.15	GH5S Cable remote trigger	689
27.16	GH5S Cheap chinese battery adapters	690
27.17	GH5S Telescopic effect	690
27.18	GH5S External HDMI	690
27.19	GH5S Synchro Scan	690
27.20	GH5S Field of view with and without SpeedBooster 0.64x	691
27.21	GH5S, all 77 video modes	693
27.22	GH5S, all C4K 8 bit modes	697
27.23	GH5S, all C4K 10 bit modes	697
27.24	GH5S, all 4K 8 bit modes	698
27.25	GH5S, all 4K 10 bit modes	699
27.26	GH5S, all anamorphic 8 bit modes	699
27.27	GH5S, all anamorphic 10 bit modes	700
27.28	GH5S, all FHD 8 bit modes	701
27.29	GH5S, all FHD 10 bit modes	702
28	Immersive 360° videos	703
29	PanoView XDV360 camera	704
30	Kodak PIXPRO SP360 4K camera	705
30.1	Remote control	706
30.2	Using the PIXPRO SP360 4K Camera as a Webcam	707
30.3	Convert 235° fisheye image to equirectangular panorama	710
31	Chinese 360° Panoramic camera	711
32	Ricoh Theta V	712
32.1	Plugins	713
32.2	Bluetooth remote control	713
32.3	Change the viewing direction	714
32.4	How to hide something from a 360° image	714
32.5	Make an equirectangular 360° image or video	717
32.6	Make a "Little-Planet" picture or video	719
32.7	Live Streaming with Ricoh Theta V	723
32.8	Ambisonics	728
32.9	Making 360° test videos with ambisonic sound	730
32.10	Play ambisonic sound with 4 speakers	733
33	Insta360 GO 2	739
34	JJRC X17 Drone	742
35	Lightning	743
36	Color temperature test with video lights	744
37	TASCAM DR-70D	746
38	TASCAM DR-701D	748
38.1	Matching the DR-701D's output level to the GH5S' input level	750
39	The Apprehension Engine: Sound effects for horror films	752
40	Synthesizers and Midi	756
40.1	The keyboard	756
40.2	Frequencies and control voltages of the keys	757
40.3	Midi	758
40.4	USB/MIDI Adapters	758
40.5	Synthesizer Abbreviations	759
40.6	Presets / Patches	759
40.7	Potentiometer behaviour	759
40.8	MatrixBrute: Oscillator tuning	760
40.9	MatrixBrute power saving mode	760
40.10	MatrixBrute AUDIO MOD Section	760
40.11	MatrixBrute Control Voltages I/O	760
40.12	MatrixBrute: Which Modules are unused?	762
41	Timelapse duration table	763
42	Zhiyun Crane 3S	764
43	Timelapse+ View	766
44	Timelapse_Control_V2	767
45	Guide 9.1	771
45.1	Install Guide 9.1	771
45.2	Control a LX200-compatible telescope	771
45.3	Add new comets to Guide 9.1	772
45.4	Add ephemerides to Guide 9.1	773
45.5	Add an overlay with comments	774
45.6	Update the position of Jupiter's great red spot	775
45.7	Add a user-defined location	776
45.8	Add a user-defined horizon	776

45.9	Switch between several user-defined horizon files	778
45.10	Install the Gaia2 catalog for Guide 9.1	778
45.11	Set up Guide 9.1 to show only the Gaia2 catalog	778
45.12	Find objects from Tycho catalog	779
45.13	Moon libration	779
46	Stellarium	780
47	Space Engine	782
47.1	Keyboard shortcuts	783
47.2	Planet classification	794
47.3	Interesting objects	797
47.4	Export Textures	797
47.5	Export Skybox	798
47.6	Scripts	799
47.7	Useful links	800
47.8	Abbreviations	800
47.9	Suggestions for improvement	801
48	Fitswork	802
49	DeepSkyStacker 4.2.3	803
49.1	How to align images without stacking them	803
49.2	How to stack on comets with known motion	804
50	SharpCap	806
51	AutoHotKey	807
52	Autostakkert!	810
53	Tycho Tracker	810
54	C# Programming project / Digital maps and elevation data	811
54.1	Where is the best place for recording nature sounds?	811
54.2	Digital topographic maps	812
54.3	Digital elevation data	813
54.4	Noise map (red = traffic noise, blue = running water noise)	815
55	Astronomy	816
55.1	Moon observing	816
55.2	Moon videos	817
55.3	Limiting magnitude for video astronomy	818
55.4	Limiting magnitude for stacked exposures	819
55.5	Useful calculations for Magnitudes	819
55.6	Artificial Stars	819
55.7	Viewing below the horizon	820
55.8	Crab Pulsar	821
55.9	How to visualize the size of the universe	822
55.10	Solar System Model	822
55.11	Selfmade Planetariums	823
56	DCF77 Decoding	824
57	The invisible Lattice Mast	826
58	Fireflies, glowworms, lightning bugs	827
59	Spherical Trigonometry and Rotation Matrices	828
60	My To Do List	829
61	List of Abbreviations	830
62	Special characters	831
63	Acknowledgements	832
64	Index	833

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 file "ffmpeg-git-essentials.7z", 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 (for example lensfun), you might need to download the "ffmpeg-git-full" version instead. But you won't need it for most 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> This wiki contains many useful informations. All available pages are listed here:
<https://trac.ffmpeg.org/wiki/TitleIndex>
- https://en.wikibooks.org/wiki/FFMPEG_An_Intermediate_Guide This wiki is very incomplete and outdated.

Other websites with FFmpeg examples:

<https://hhsprings.bitbucket.io/docs/programming/examples/ffmpeg/index.html#>

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.
- In most cases, there are no restrictions for video size (width * height), except for extremely large sizes.
- 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.¹
- 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.
- FFmpeg and DaVinci Resolve complement each other. It's best if you know and use both of them.

¹ Why is FFmpeg's official documentation so incomplete? I think documentation has the lowest possible priority for the developers, and most of those users who could write better documentation (including me) are unable or unwilling to work with GIT, which is the only way to make any changes to the documentation.

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 x 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 black holes.
- Stabilizing of shaky videos
- Deflicker, for reducing brightness steps in timelapse.
- Change the video compression, to make the video smaller.
- and many, 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".
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. But the above ways aren't recommended, because in many cases the command lines are quite long and you don't want to type the same command line over and over again. 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>

The following was copied from the above link:

For Windows 7:

1. From the desktop, right-click the Computer icon and select Properties. If you don't have a Computer icon on your desktop, click Start, right-click the Computer option in the Start menu, and select Properties.
2. Click the Advanced System Settings link in the left column.
3. In the System Properties window, click the Advanced tab, then click the Environment Variables button near the bottom of that tab.
4. 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.

For Windows 10:

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 [...].

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 3 out.mp4  
  
pause :: wait for a keypress
```

What's the meaning of the parts?

rem A simple ...	This is a comment
c:\ffmpeg\ffmpeg	This is the path to ffmpeg.exe
-framerate 5	This defines how fast the pictures are read in, in this case 5 pictures per second.
-start_number 3551	This is the number of the first picture, in this case 3551
-i IMG_%%4d.jpg	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. The double %% characters are only required in batch files, because the % character must be escaped. If you type the command line directly in the console window, then you must use a single % character instead.
-i birds.mp3	This is the second input file, in this case an audio file.
^	If the command line becomes too long in the batch file, you can break it with the ^ character and continue in the next line. FFmpeg will get the whole line without the ^ character.
-shortest	This option means that the length of the output video is determined by the shortest of the two input files.
-codec:v mpeg4	This option means that a MPEG4 video will be produced.
-q:v 2	This is an option for the quality of the output video. 1 is best quality, 3 is normal, 31 is strongest compression.
out.mp4	Filename of the output video
pause	This command waits for a keypress, so that you have a chance to see any error messages before the console window closes.
:: wait for ...	Everything right of :: is a comment until the end of the line.

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 3" 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=3"                      :: MP4 Quality, 1 ist best quality, 3 is normal, 31 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.

2 FFmpeg in detail

2.1 Convert from one video format to another video format

Some examples for format conversion:

```
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 Change the container format

If want to change only the container format from mkv to mp4, it's not necessary to re-encode the video and audio streams. These commands are very fast:

```
ffmpeg -i in.mkv -vcodec copy -acodec copy out.mp4
or
ffmpeg -i in.mkv -c:v copy -c:a copy out.mp4
```

2.3 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 = Number_of_images / 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.

```
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=3"                           :: MP4 Quality, 1 is best Quality, 3 is normal, 31 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.

2.4 Timelapse or slideshow from many images, with crossfading

```
rem Make a timelapse 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=3"                           :: MP4 Quality, 1 is best Quality, 3 is normal, 31 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. It 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 explicitly 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 chosen 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.

How to repeat the frames, so that the length fits to the music length:

Add -loop 1 and -shortest

2.5 Slideshow with different durations

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

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.

It's also possible to do the same thing with the concat demuxer. Make a text file with this content:

```
file '/path/to/dog.png'  
duration 5  
file '/path/to/cat.png'  
duration 1  
file '/path/to/rat.png'  
duration 3  
file '/path/to/tapeworm.png'  
duration 2  
file '/path/to/tapeworm.png'
```

Note: The last image has to be specified twice, the second one without any duration.

Then use this command line:

```
ffmpeg -f concat -i input.txt -vsync vfr -pix_fmt yuv420p output.mp4  
pause
```

See also: <https://trac.ffmpeg.org/wiki/Slideshow>

2.6 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:

```
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%

pause
```

Images scrolling from top to bottom:

```
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%
pause
```

Images scrolling from bottom to top:

```
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%
pause
```

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:

```
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 1 -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 doesn't accept expressions.

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%  
  
pause
```

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%  
  
pause
```

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%  
  
pause
```

2.7 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 extracts 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
```

2.8 Extract the n_th frame from a video

```
rem Make a test video
ffmpeg -f lavfi -i testsrc2=size=vga -t 3 -y test.mov

rem Extract the N_th frame
set "N=10"      :: Frame number to be extracted (assuming the first frame is number 1)
ffmpeg -i test.mov -vf select='eq(n,%N%-1)' -frames 1 -y out.png
pause
```

Note: The frame numbers begin with 0, that's why you must compare n with 9, if you want the 10_th frame.

2.9 Extract the first and last frame from a video

```
rem Extract the first frame
ffmpeg -i in.mp4 -frames 1 -y first_frame.jpg

rem Extract the last frame
ffmpeg -sseof -0.2 -i in.mp4 -update 1 -y last_frame.jpg
pause
```

2.10 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
                           :: Typical values are in the -30...+30 range
set "QU=3"                  :: MP4 Quality, 1 is best Quality, 3 is normal, 31 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.
Warning: De-saturating a video with eq=saturation=0 produces a greenish tint in the output video. It's better to use hue=s=0.
- 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 write the parameters in the command line. They are always executed in the order contrast, brightness, gamma.

2.11 Strong contrast enhancement

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

Filter	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)	Notes
eq	<code>eq=brightness=0.3,eq=contrast=5</code>	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. Warning: The eq filter introduces dithering when used with 10-bit data.
	<code>eq=brightness=0.3:contrast=5</code>	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.
	<code>eq=brightness=1.5:contrast=5</code>	This doesn't work because the brightness value isn't in the allowed range [-1 ... +1]
geq	<code>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)'</code>	The drawback of the geq filter is that it's slow, has no built-in limiter and the function must be different for 8-bit and 10-bit videos.
	<code>For 10-bit video: geq=lum='5*(lum(X,Y)-102.4)':cr='cr(X,Y)':cb='cb(X,Y)' For 10-bit video with limiter: geq=lum='clip(5*(lum(X,Y)-102.4),0,1023)':cr='cr(X,Y)':cb='cb(X,Y)'</code>	For 10-bit data the geq filter produces the correct result without dithering. Best method for 10-bit data, if it's used together with the clip function as limiter.
lutyuv	<code>Version 1: lutyuv=y='5*(val-0.1*maxval)' Version 2: lutyuv=y='5*val-0.5*maxval' With limiter: lutyuv=y='clip(5*(val-0.1*maxval),minval,maxval)' For 10-bit data: lutyuv=y='32*val-16384' (This works, but I don't understand why)</code>	Both versions work fine with 8-bit data and the additional limiter seems to be unnecessary. Warning: This filter fails with 10-bit data!

colorlevels	<code>colorlevels=rimin=0.1:gimin=0.1:bimin=0.1:rimax=0.3:gimax=0.3:bimax=0.3</code>	Best method for 8-bit data 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. Warning: This filter fails with 10-bit data!
curves	<code>curves=all='0/0 0.1/0 0.3/1 1/1'</code>	This is a nonlinear transfer function because it uses a smooth curve through the specified points.
exposure	<code>exposure=exposure=2.3219:black=-0.1</code> Note: $\log(5)/\log(2)=2.3219$ With additional limiter: <code>exposure=exposure=2.3219:black=-0.1,limiter</code>	The order is first "black level", then "exposure". This filter works internally with floats. "exposure" is limited to the [-3..+3] range, which corresponds to a factor from 0.125 to 8.0. Because it has no built-in limiter, it's highly recommended to use an additional limiter.

2.12 Inverting a video or image (make a negative)

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

Filter	Example	Notes
eq	eq=contrast=-1	This changes only bright to dark and vice versa, but keeps the colors as they are.
negate	negate	This negates all channels and changes the colors to their complementary colors.
geq	geq=r='255-r(X,Y)':g='255-g(X,Y)':b='255-b(X,Y)'	Same result as negate. Can also be used if only one or two channels are to be inverted. For 10-bit videos you must replace 255 by 1023.

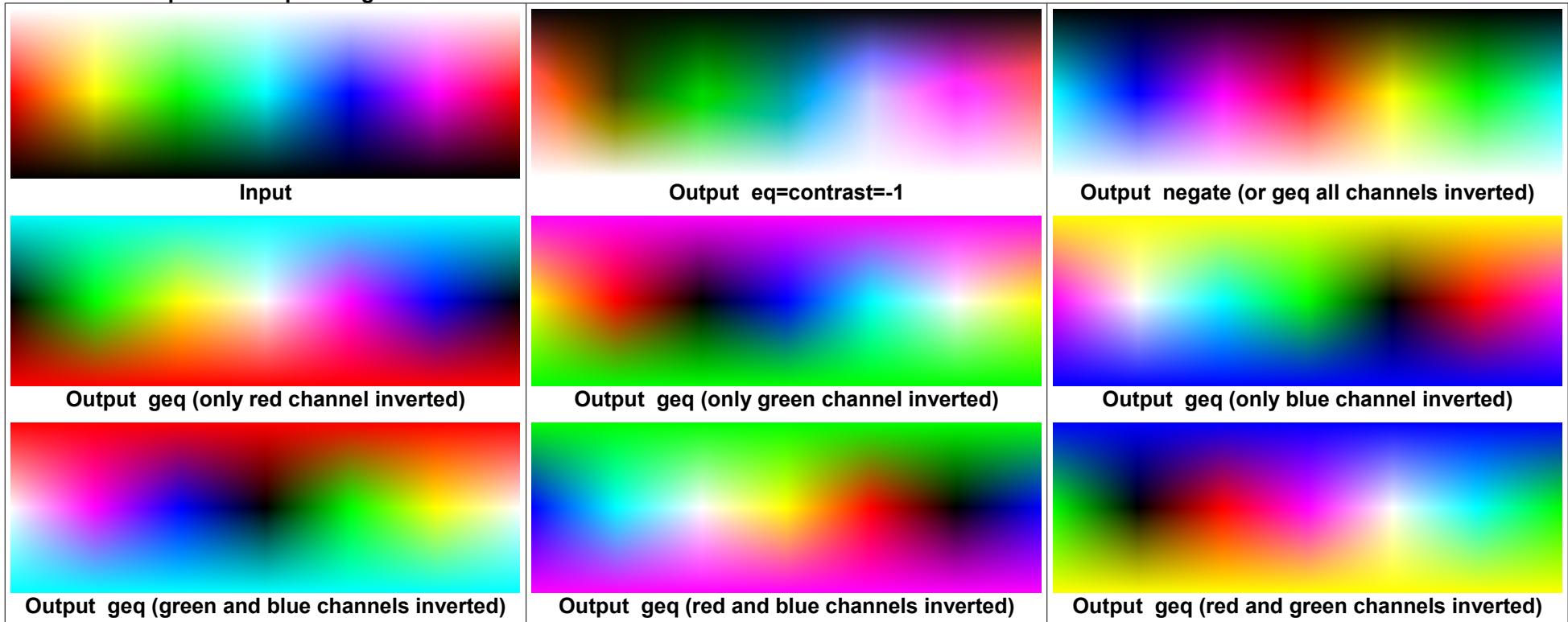
This is an example. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -vf eq=contrast=-1 -y eq.png
ffmpeg -i spectrum.png -vf negate -y negate.png
ffmpeg -i spectrum.png -vf geq=r='255-r(X,Y)':g='g(X,Y)':b='b(X,Y)' -y geq_r.png
ffmpeg -i spectrum.png -vf geq=r='r(X,Y)':g='255-g(X,Y)':b='b(X,Y)' -y geq_g.png
ffmpeg -i spectrum.png -vf geq=r='r(X,Y)':g='g(X,Y)':b='255-b(X,Y)' -y geq_b.png
ffmpeg -i spectrum.png -vf geq=r='255-r(X,Y)':g='255-g(X,Y)':b='b(X,Y)' -y geq_rg.png
ffmpeg -i spectrum.png -vf geq=r='255-r(X,Y)':g='g(X,Y)':b='255-b(X,Y)' -y geq_rb.png
ffmpeg -i spectrum.png -vf geq=r='r(X,Y)':g='255-g(X,Y)':b='255-b(X,Y)' -y geq_gb.png
ffmpeg -i spectrum.png -vf geq=r='255-r(X,Y)':g='255-g(X,Y)':b='255-b(X,Y)' -y geq_rgb.png

pause
```

These are the input and output images:



2.13 Colorchannelmixer filter

Color corrections can be made with the "colorchannelmixer" filter. In this example the red channel is enhanced by a factor 1.2 and the blue channel is attenuated by a factor 0.8. The values must be in the [-2 ... +2] range. The input image is a spectrum with white at the top and black at the bottom:

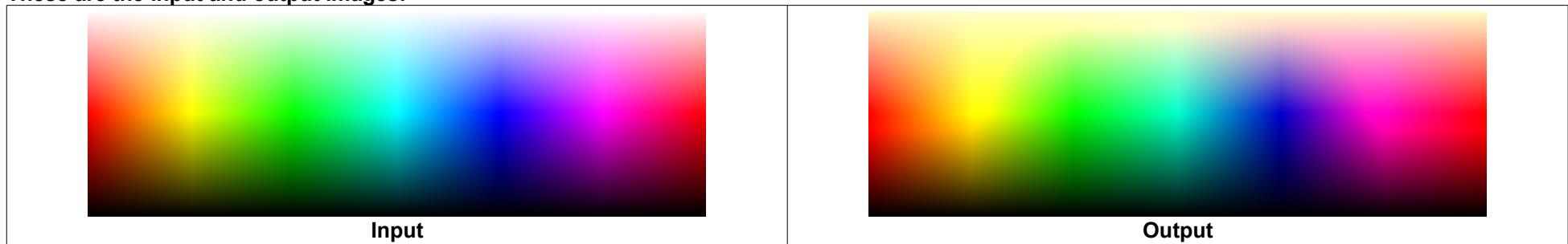
```
set "R=1.2"          :: Factor for red channel
set "G=1.0"          :: Factor for green channel
set "B=0.8"          :: Factor for blue channel

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)',scale=iw/4:ih/4 -frames
1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colorchannelmixer=rr=%R%:gg=%G%:bb=%B% out.png

pause
```

These are the input and output images:



Cyclic swapping of the RGB channels in one or the other direction:

```
colorchannelmixer=0:0:1:0:1:0:0:0:1:0:0          :: red becomes green, green becomes blue, blue becomes red
colorchannelmixer=0:1:0:0:0:0:1:0:1:0:0          :: red becomes blue, green becomes red, blue becomes green
```

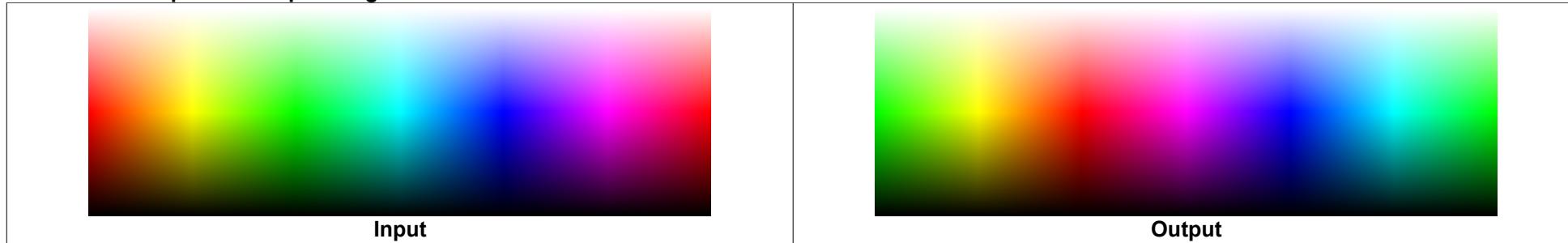
Example for exchanging the red and green components:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -vf colormixer=0:1:0:0:1:0:0:0:0:1:0 -y out.png

pause
```

These are the input and output images:



The options of the "colormixer" filter are specified in this order:

rr, rg, rb, ra,	red_out = red_in * rr + green_in * rg + blue_in * rb + alpha_in * ra
gr, gg, gb, ga,	green_out = red_in * gr + green_in * gg + blue_in * gb + alpha_in * ga
br, bg, bb, ba,	blue_out = red_in * br + green_in * bg + blue_in * bb + alpha_in * ba
ar, ag, ab, aa	alpha_out = red_in * ar + green_in * ag + blue_in * ab + alpha_in * aa

Copy the red channel to the green and blue channels

```
colormixer=1:0:0:0:1:0:0:0:1:0:0 :: copy the red channel to the green and blue channels
```

2.14 Shuffleplanes filter

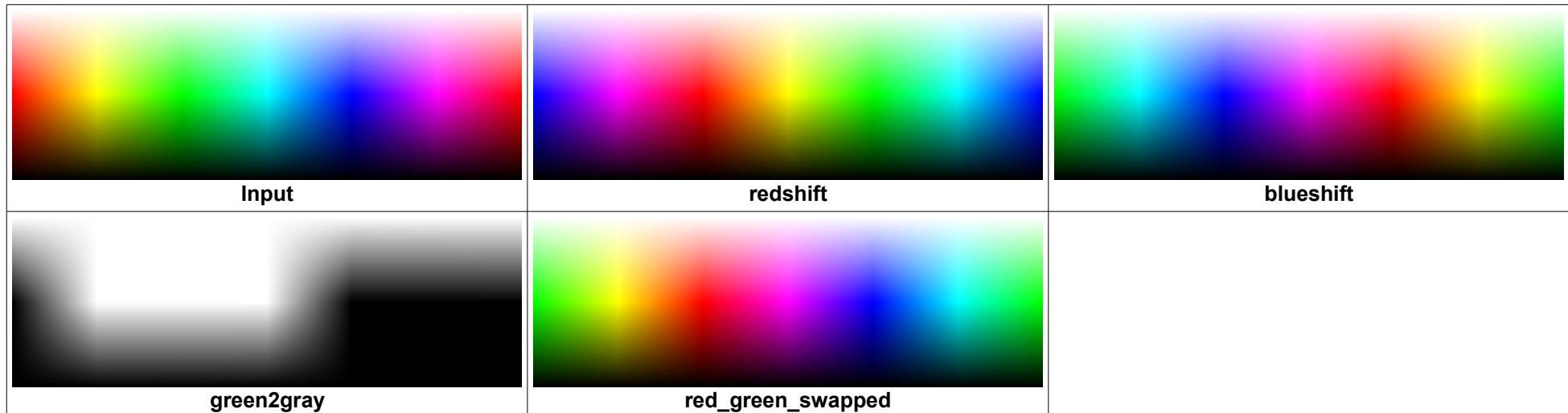
This filter can swap or copy planes. What's stored in a plane depends on the pixel format. It must be a planar pixel format like GBGP or YUV for example. If the input pixel format isn't planar (like RGB24), it will be automatically converted to a planar format and you might get unexpected results. In the following examples the pixel format is GBRP:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi format=gbrp,shuffleplanes=1:2:0 -y redshift.png
ffmpeg -i spectrum.png -lavfi format=gbrp,shuffleplanes=2:0:1 -y blueshift.png
ffmpeg -i spectrum.png -lavfi format=gbrp,shuffleplanes=0:0:0 -y green2gray.png
ffmpeg -i spectrum.png -lavfi format=gbrp,shuffleplanes=2:1:0 -y red_green_swapped.png

pause
```

These are the input and output images:



2.15 Day-for-Night

Day-for-Night is a technique for shooting night scenes in daytime. In times of analog filming, it was realized by using a red filter (which darkens the blue sky) and underexposing by about 2 stops.

It can be simulated as follows. The input image is a spectrum with white at the top and black at the bottom:

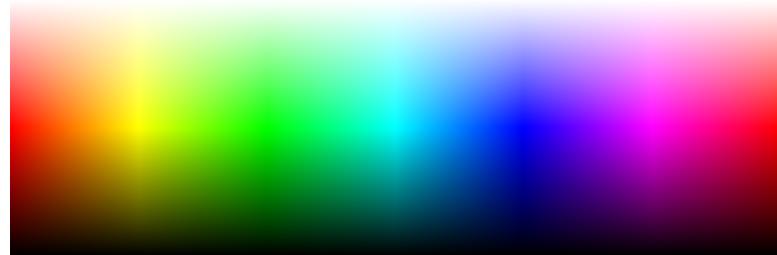
```
set "BW=0.3"    :: This parameter sets how much of the red channel is converted to black and white
set "RGB=0.1"   :: This parameter sets how much of the original RGB channels is kept

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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colormixer=rr=%BW%+%RGB%:gr=%BW%:br=%BW%:gg=%RGB%:bb=%RGB% -y out.png

pause
```

These are the input and output images:



See also https://en.wikipedia.org/wiki/Day_for_night

For example, day-for-night was used in Wim Wenders' film "Kings of the Road" (german: "Im Lauf der Zeit"):
https://en.wikipedia.org/wiki/Kings_of_the_Road

2.16 Colorcorrect filter

The "colorcorrect" filter has 4 parameters, which can be set to any value in the [-1 ... +1] range. The default values are 0:

rl	Correction value for red shadows
bl	Correction value for blue shadows
rh	Correction value for red highlights
bh	Correction value for blue highlights

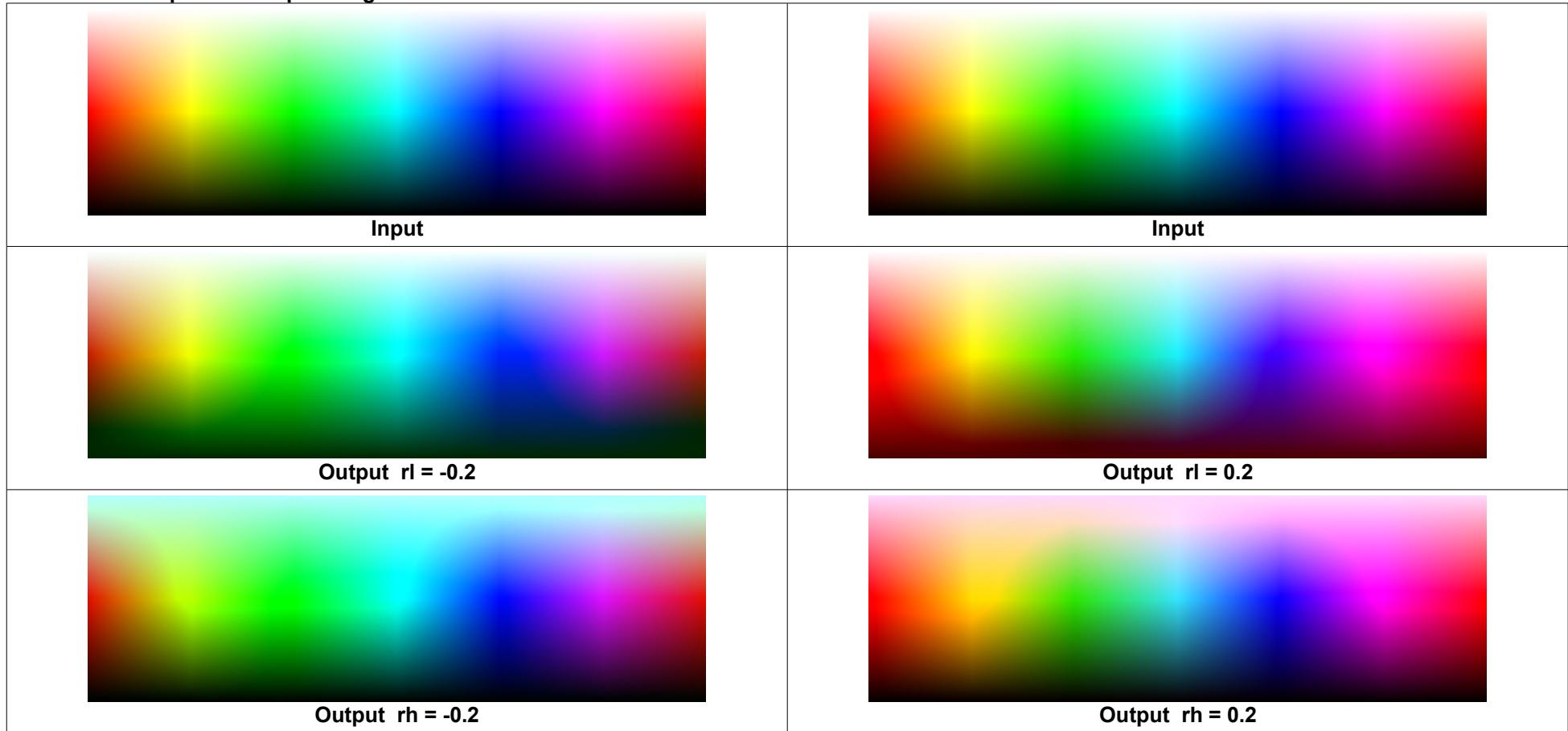
This is an example. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colorcorrect=rl=-0.2 -y rl_-02.png
ffmpeg -i spectrum.png -lavfi colorcorrect=rl=0.2 -y rl_+02.png
ffmpeg -i spectrum.png -lavfi colorcorrect=rh=-0.2 -y rh_-02.png
ffmpeg -i spectrum.png -lavfi colorcorrect=rh=0.2 -y rh_+02.png

pause
```

These are the input and output images:



The colorcorrect filter does also have a "saturation" option which is pretty clear.

There is also an "analyze" option, but I didn't fully understand it. If you know how it works, please let me know.

2.17 Colortemperature filter

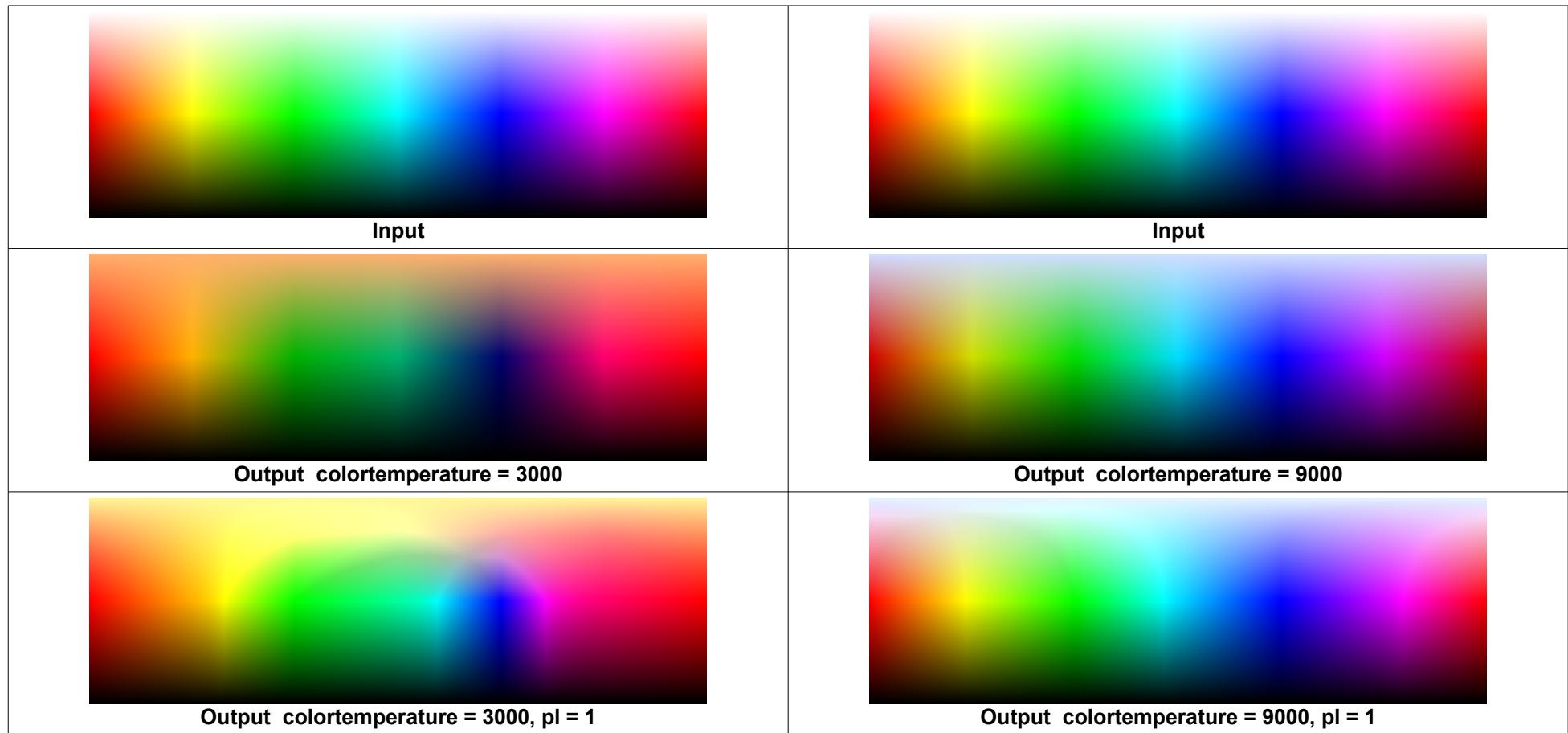
This is an example for the "colortemperature" filter. The default temperature is 6500K. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colortemperature=3000 -y 3000.png
ffmpeg -i spectrum.png -lavfi colortemperature=9000 -y 9000.png
ffmpeg -i spectrum.png -lavfi colortemperature=3000:pl=1 -y 3000pl.png
ffmpeg -i spectrum.png -lavfi colortemperature=9000:pl=1 -y 9000pl.png

pause
```

These are the input and output images:



2.18 Hue filter

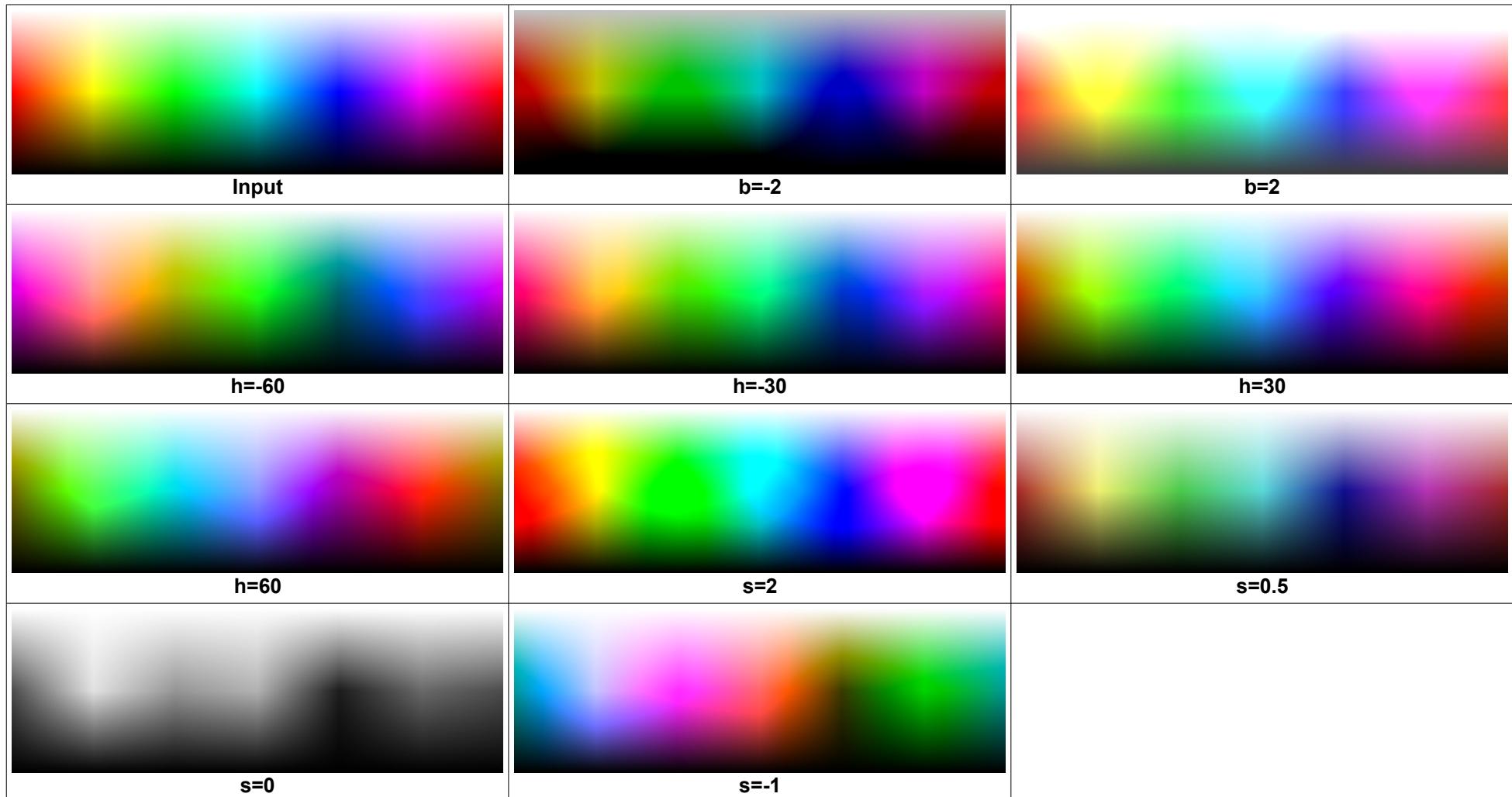
This filter changes hue, saturation and brightness:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi hue=h=60 -y h_60.png
ffmpeg -i spectrum.png -lavfi hue=h=30 -y h_30.png
ffmpeg -i spectrum.png -lavfi hue=h=-60 -y h_-60.png
ffmpeg -i spectrum.png -lavfi hue=h=-30 -y h_-30.png
ffmpeg -i spectrum.png -lavfi hue=s=2 -y s_2.png
ffmpeg -i spectrum.png -lavfi hue=s=0 -y s_0.png
ffmpeg -i spectrum.png -lavfi hue=s=0.5 -y s_05.png
ffmpeg -i spectrum.png -lavfi hue=s=-1 -y s_-1.png
ffmpeg -i spectrum.png -lavfi hue=b=2 -y b_2.png
ffmpeg -i spectrum.png -lavfi hue=b=-2 -y b_-2.png

pause
```

These are the input and output images:



2.19 Huesaturation filter

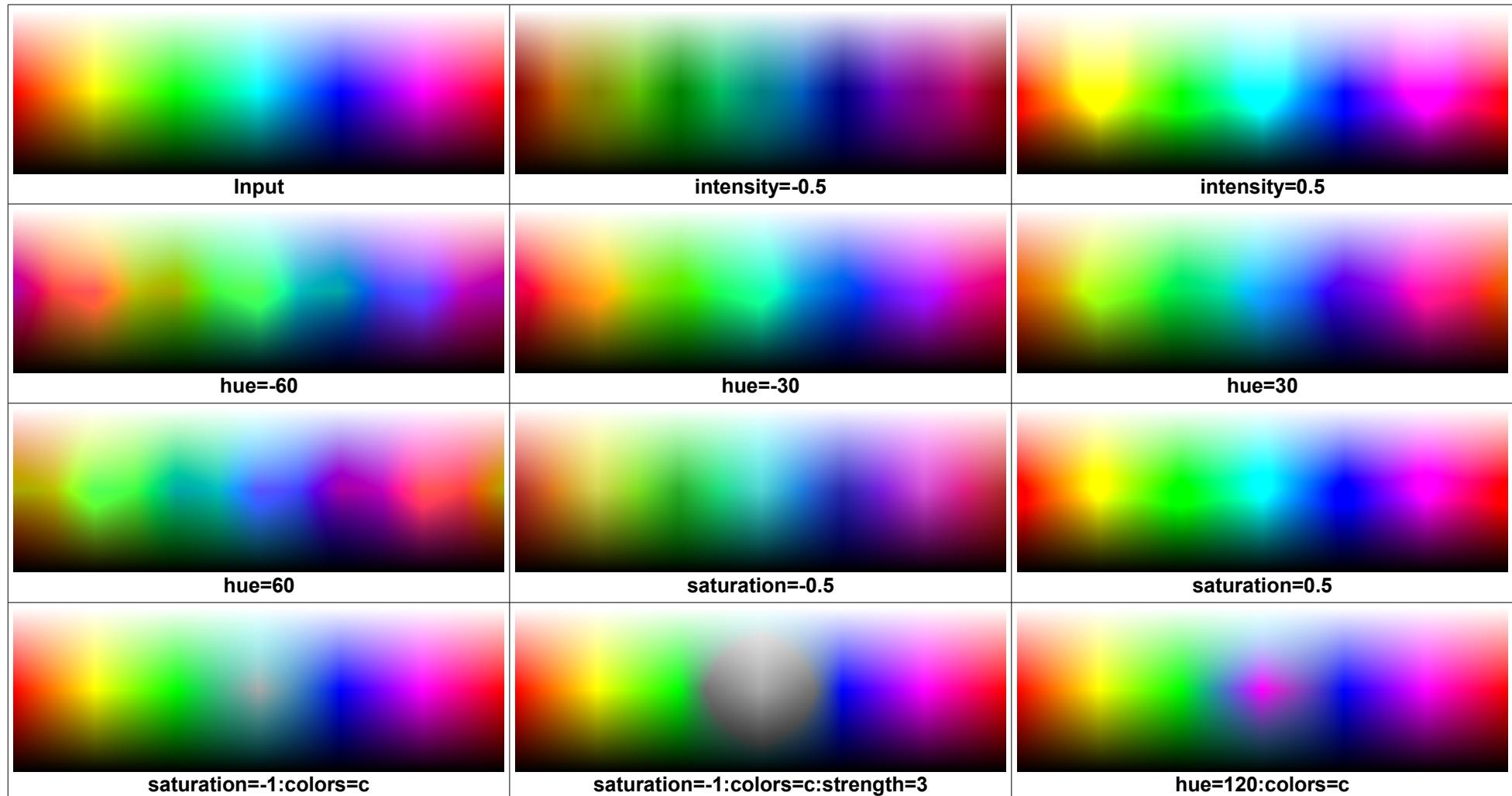
This filter works in RGB color space and changes hue, saturation and intensity:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi huesaturation=hue=60 -y h_60.png
ffmpeg -i spectrum.png -lavfi huesaturation=hue=30 -y h_30.png
ffmpeg -i spectrum.png -lavfi huesaturation=hue=-60 -y h_-60.png
ffmpeg -i spectrum.png -lavfi huesaturation=hue=-30 -y h_-30.png
ffmpeg -i spectrum.png -lavfi huesaturation=saturation=0.5 -y s_05.png
ffmpeg -i spectrum.png -lavfi huesaturation=saturation=-0.5 -y s_-05.png
ffmpeg -i spectrum.png -lavfi huesaturation=intensity=0.5 -y i_05.png
ffmpeg -i spectrum.png -lavfi huesaturation=intensity=-0.5 -y i_-05.png
ffmpeg -i spectrum.png -lavfi huesaturation=saturation=-1:colors=c -y cyan_s_-1.png
ffmpeg -i spectrum.png -lavfi huesaturation=saturation=-1:colors=c:strength=3 -y cyan_s_-1_strength_3.png
ffmpeg -i spectrum.png -lavfi huesaturation=hue=120:colors=c -y cyan_h_120.png

pause
```

These are the input and output images:



2.20 Selectivecolor filter

How does it work? There are 9 ranges of colors:

reds	Adjustments for red pixels (pixels where the red component is the maximum)
yellows	Adjustments for yellow pixels (pixels where the blue component is the minimum)
greens	Adjustments for green pixels (pixels where the green component is the maximum)
cyans	Adjustments for cyan pixels (pixels where the red component is the minimum)
blues	Adjustments for blue pixels (pixels where the blue component is the maximum)
magentas	Adjustments for magenta pixels (pixels where the green component is the minimum)
whites	Adjustments for white pixels (pixels where all components are greater than 128)
neutrals	Adjustments for all pixels except pure black and pure white
blacks	Adjustments for black pixels (pixels where all components are less than 128)

For each of these color ranges, four values in the [-1 .. 1] range can be specified for adjustment of cyan, magenta, yellow and black.

There are absolute and relative modes, however the difference between them isn't easy to understand.

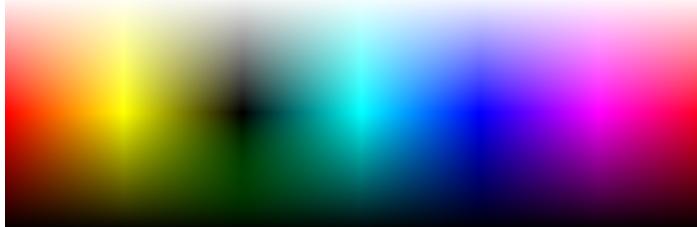
That's why I calculated some example images. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi selectivecolor=greens="0 1 0 0":correction_method=absolute -y abs_greens_0_1_0_0.png
ffmpeg -i spectrum.png -lavfi selectivecolor=greens="0 1 0 0":correction_method=relative -y rel_greens_0_1_0_0.png
ffmpeg -i spectrum.png -lavfi selectivecolor=greens="0 -1 0 0":correction_method=absolute -y abs_greens_0_-1_0_0.png
ffmpeg -i spectrum.png -lavfi selectivecolor=greens="0 -1 0 0":correction_method=relative -y rel_greens_0_-1_0_0.png
ffmpeg -i abs_greens_0_-1_0_0.png -i rel_greens_0_-1_0_0.png -lavfi blend=all_mode=grainextract -y diff.png

pause
```

These are the input and output images:

	Absolute mode (this is the default)	Relative mode
Input (same image for absolute and relative mode)		
Output for greens="0 1 0 0"		
Output for greens="0 -1 0 0"		

Obviously there is no difference between absolute and relative modes, if the correction value is negative (this was verified by calculating the difference image in the last FFmpeg command).

I find it quite surprising that pure green with +1 correction for magenta (in absolute mode) gives black output (and not white).

The same filter is also in Photoshop (including the absolute and relative modes) and you can google how it is assumed to work.

In my opinion the behaviour of this filter is difficult to understand.

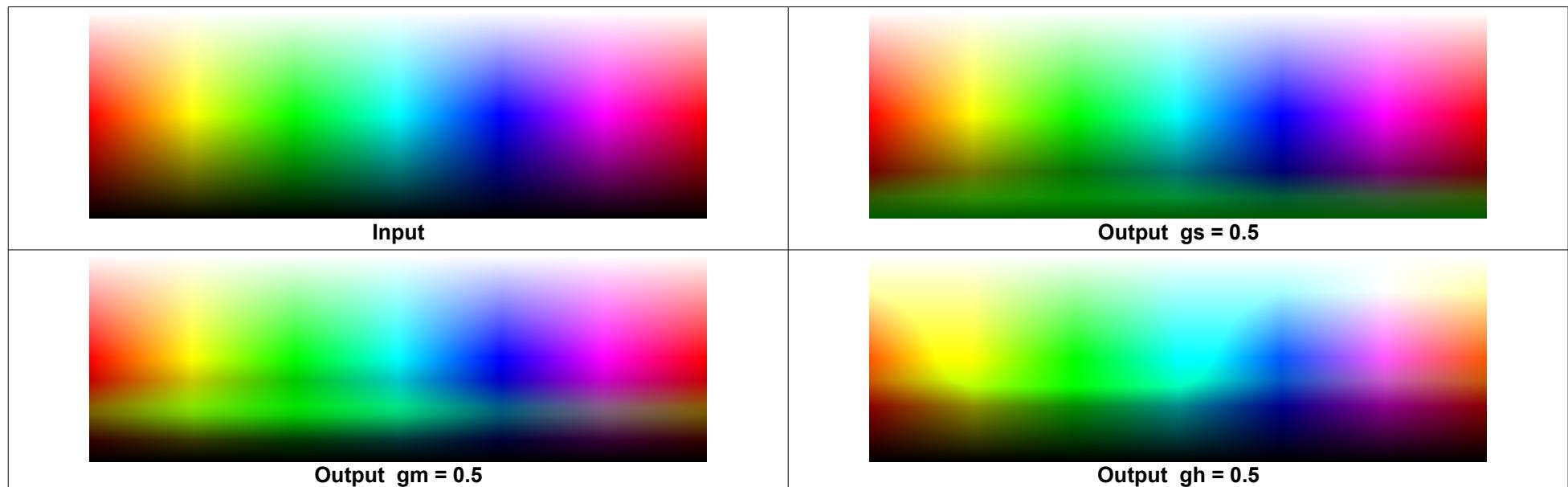
2.21 Colorbalance filter

This is an example for the colorbalance filter. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colorbalance=gs=0.5 -y gs_05.png
ffmpeg -i spectrum.png -lavfi colorbalance=gm=0.5 -y gm_05.png
ffmpeg -i spectrum.png -lavfi colorbalance=gh=0.5 -y gh_05.png

pause
```

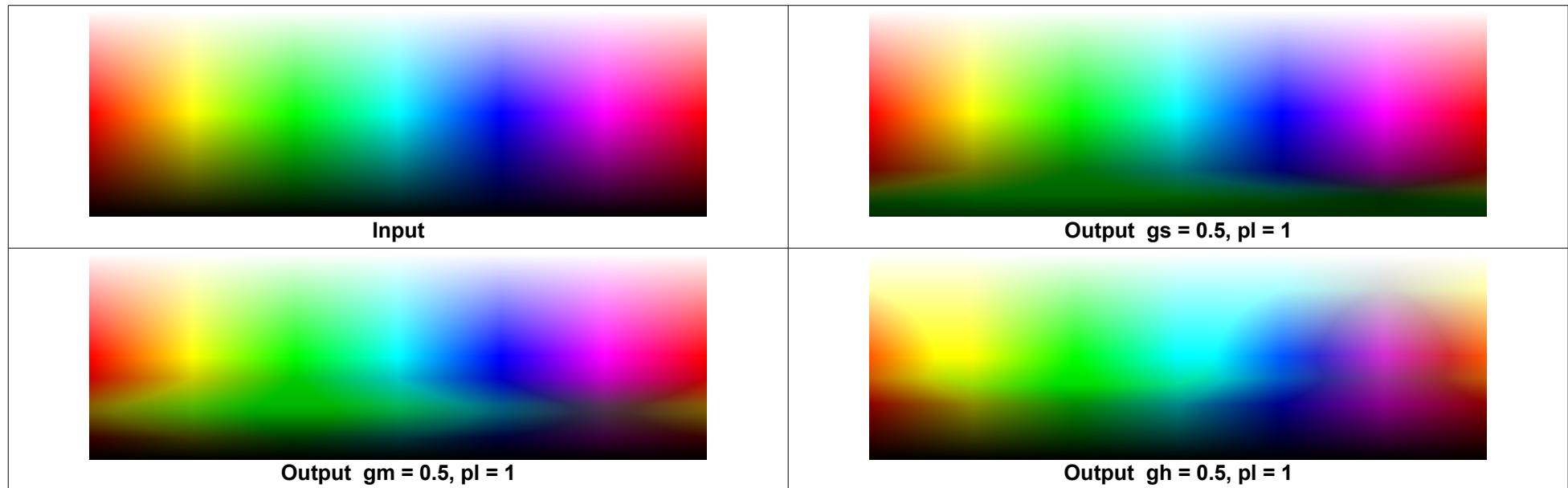


There is also a "preserve lightness" option which can be enabled:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colorbalance=gs=0.5:pl=1 -y gs_05.png
ffmpeg -i spectrum.png -lavfi colorbalance=gm=0.5:pl=1 -y gm_05.png
ffmpeg -i spectrum.png -lavfi colorbalance=gh=0.5:pl=1 -y gh_05.png

pause
```



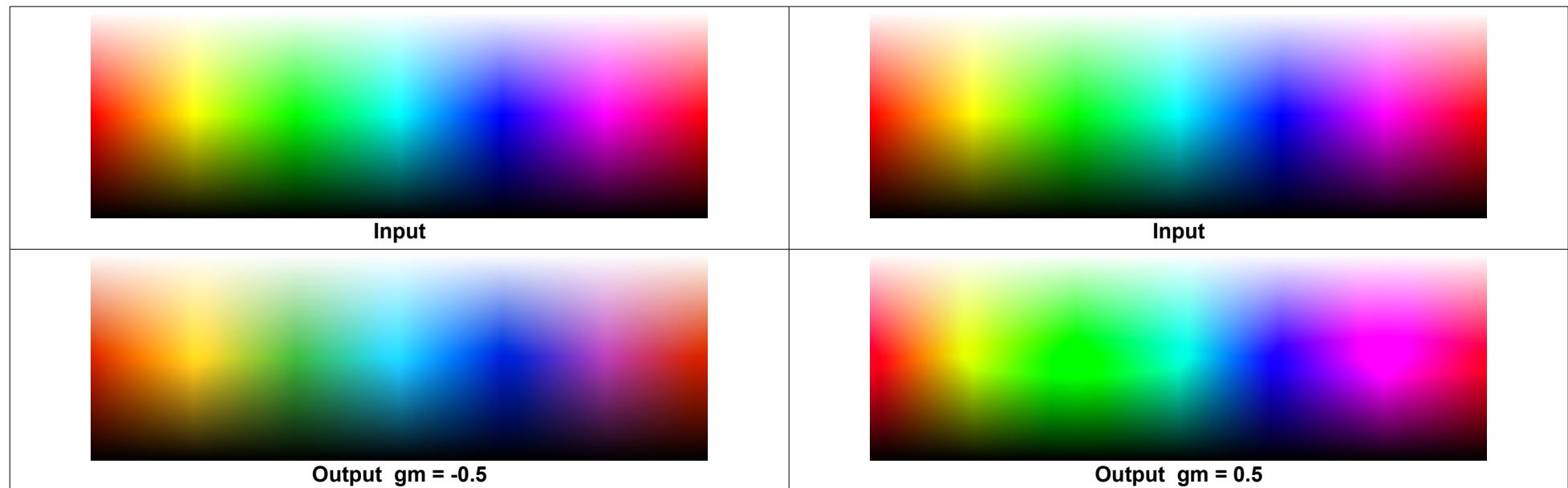
2.22 Colorcontrast filter

This is an example for the colorcontrast filter. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colorcontrast=gm=0.5:gmw=1 -y gm_05.png
ffmpeg -i spectrum.png -lavfi colorcontrast=gm=-0.5:gmw=1 -y gm_-05.png

pause
```

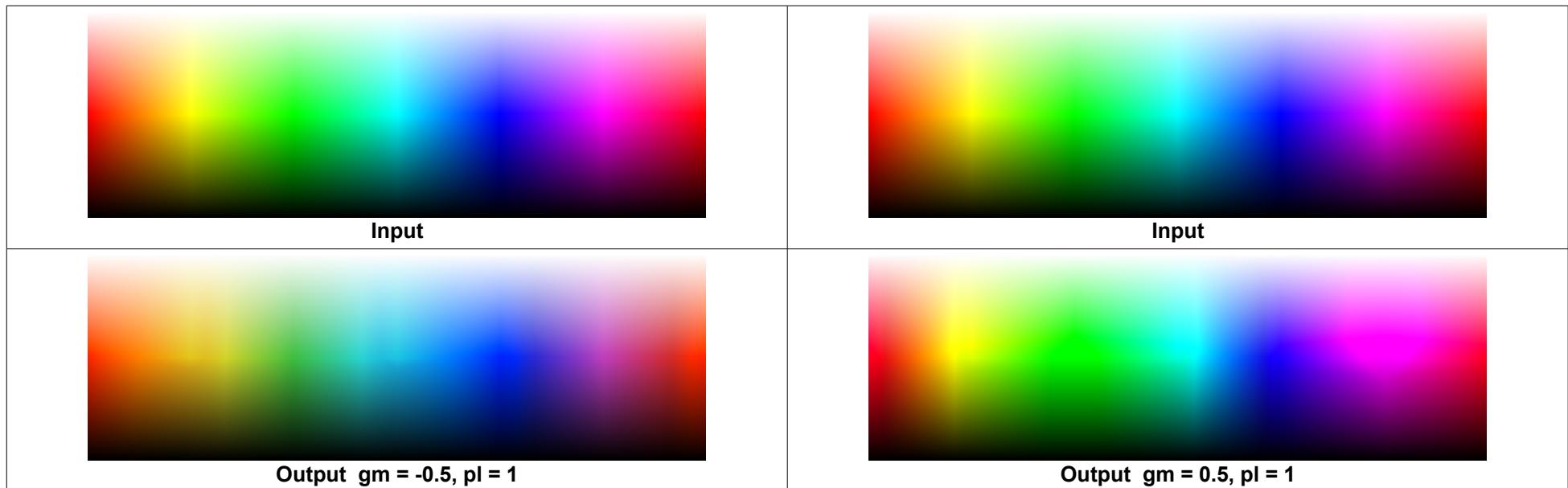


There is also a "preserve lightness" option which can be enabled:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi colorcontrast=gm=0.5:gmw=1:pl=1 -y gm_05.png
ffmpeg -i spectrum.png -lavfi colorcontrast=gm=-0.5:gmw=1:pl=1 -y gm_-05.png

pause
```



2.23 Monochrome filter

The "monochrome" filter can be used to convert a colored video to monochrome.

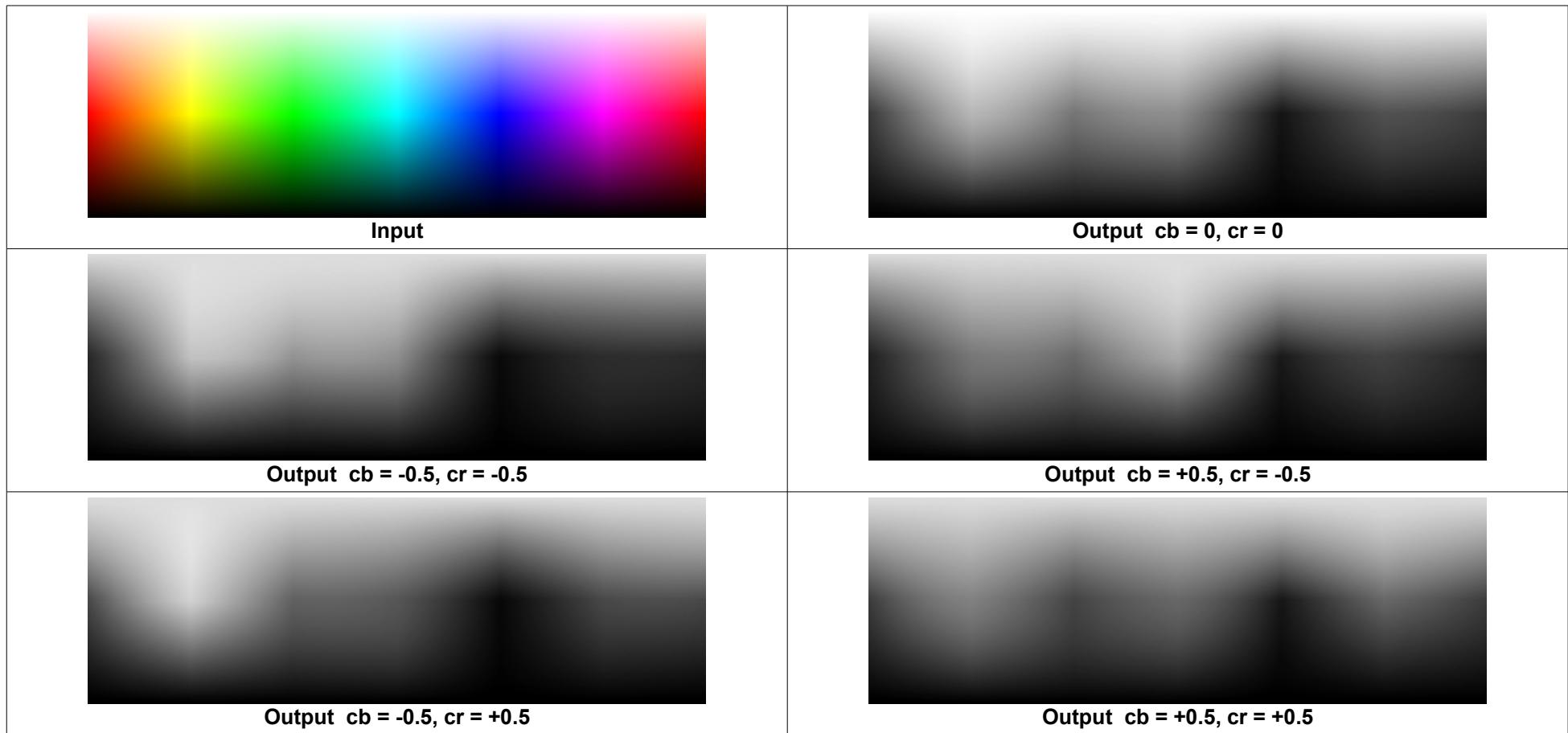
This is an example. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi monochrome -y cb_0_cr_0.png
ffmpeg -i spectrum.png -lavfi monochrome=cb=0.5:cr=0.5 -y cb_05_cr_05.png
ffmpeg -i spectrum.png -lavfi monochrome=cb=-0.5:cr=0.5 -y cb_-05_cr_05.png
ffmpeg -i spectrum.png -lavfi monochrome=cb=0.5:cr=-0.5 -y cb_05_cr_-05.png
ffmpeg -i spectrum.png -lavfi monochrome=cb=-0.5:cr=-0.5 -y cb_-05_cr_-05.png

pause
```

These are the input and output images:



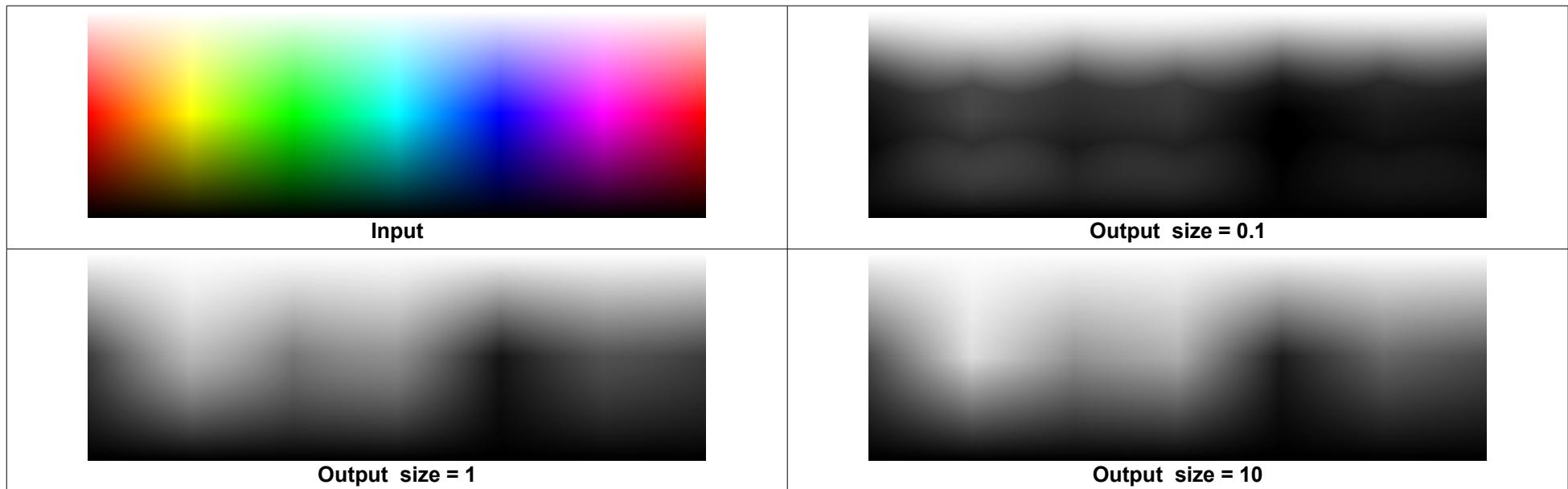
This is an example for the "size" option, which can be set in the [0.1 ... 10] range:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi monochrome=cr=0:cb=0:size=.1 -y size01.png
ffmpeg -i spectrum.png -lavfi monochrome=cr=0:cb=0:size=1 -y size1.png
ffmpeg -i spectrum.png -lavfi monochrome=cr=0:cb=0:size=10 -y size10.png

pause
```

These are the input and output images:



It's also possible to make a video monochrome by setting the saturation to 0, for example with the "eq" filter.

```
ffmpeg -i in.mp4 -lavfi eq=saturation=0 -y out.mp4  
pause
```

This is a very simple method for converting a YUV color video to a monochrome video. The Y plane is extracted:

```
ffmpeg -i in.mp4 -lavfi extractplanes=y -y out.mp4  
pause
```

2.24 Vibrance filter

Vibrance is difficult to understand. It's similar to saturation, but it protects skintones.

See also: <https://patkay.com/blogs/pk/the-difference-between-vibrance-and-saturation-in-lightroom>

This is an example. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi vibrance=intensity=1 -y int_1.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1 -y int_-1.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:alternate=1 -y int_-1_alt.png

ffmpeg -i spectrum.png -lavfi vibrance=intensity=1:gbal=-3 -y int_1_gbal_-3.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:gbal=-3 -y int_-1_gbal_-3.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:alternate=1:gbal=-3 -y int_-1_alt_gbal_-3.png

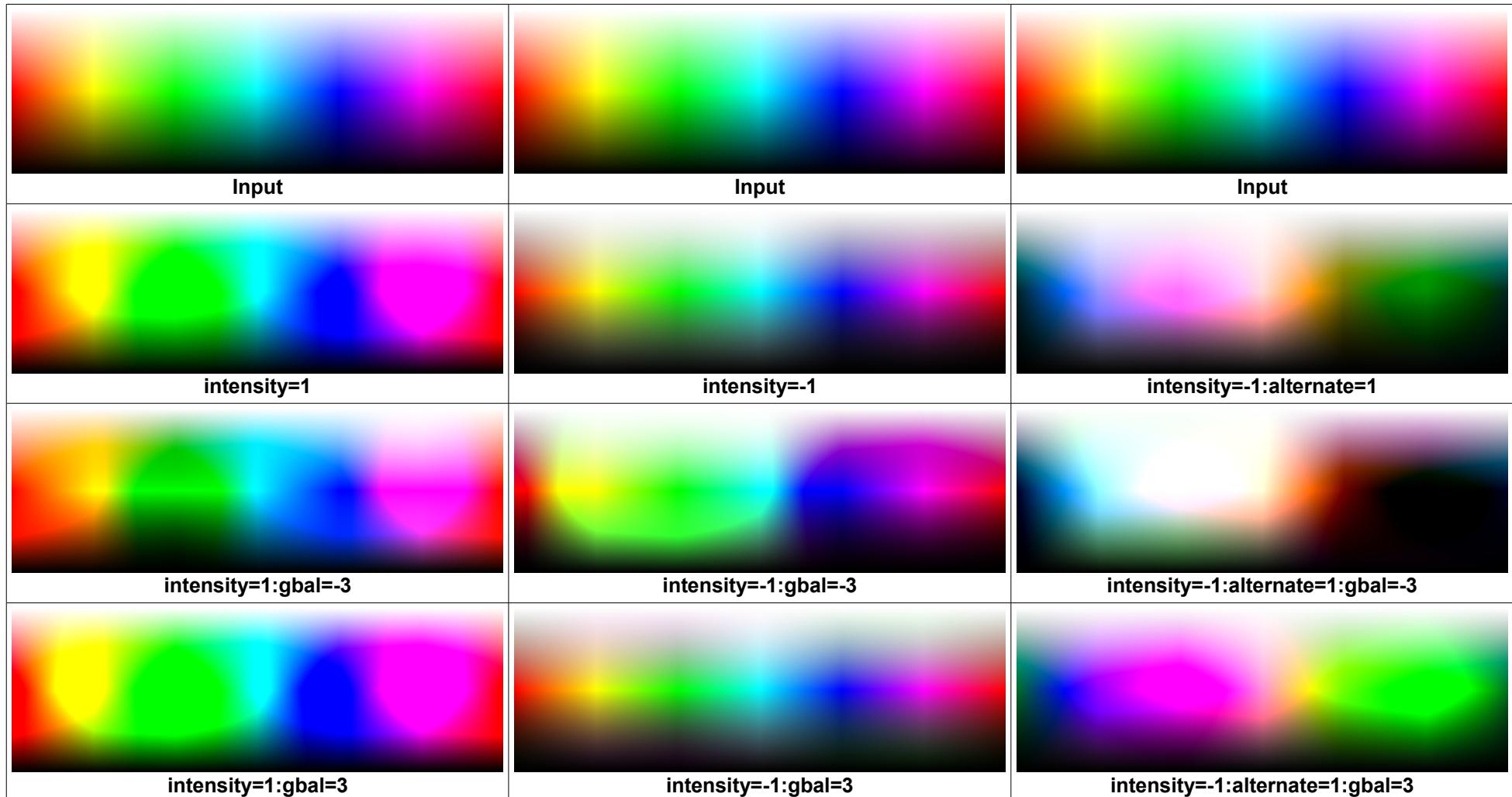
ffmpeg -i spectrum.png -lavfi vibrance=intensity=1:gbal=3 -y int_1_gbal_3.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:gbal=3 -y int_-1_gbal_3.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:alternate=1:gbal=3 -y int_-1_alt_gbal_3.png

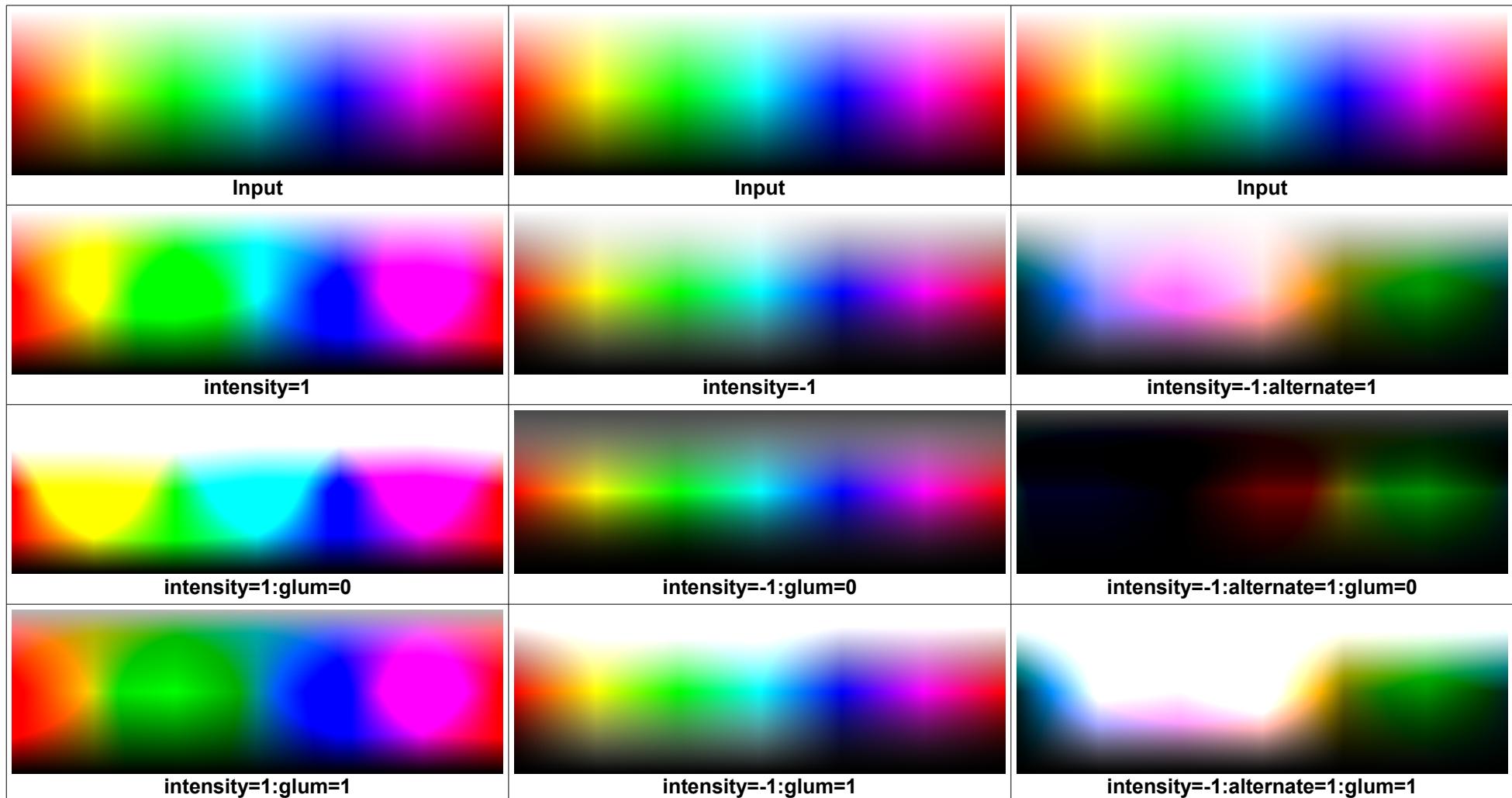
ffmpeg -i spectrum.png -lavfi vibrance=intensity=1:glum=0 -y int_1_glum_0.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:glum=0 -y int_-1_glum_0.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:alternate=1:glum=0 -y int_-1_alt_glum_0.png

ffmpeg -i spectrum.png -lavfi vibrance=intensity=1:glum=1 -y int_1_glum_1.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:glum=1 -y int_-1_glum_1.png
ffmpeg -i spectrum.png -lavfi vibrance=intensity=-1:alternate=1:glum=1 -y int_-1_alt_glum_1.png

pause
```

These are the input and output images:





Note: The default value of the "glum" value seems to be about 0.7.

2.25 Swapuv filter

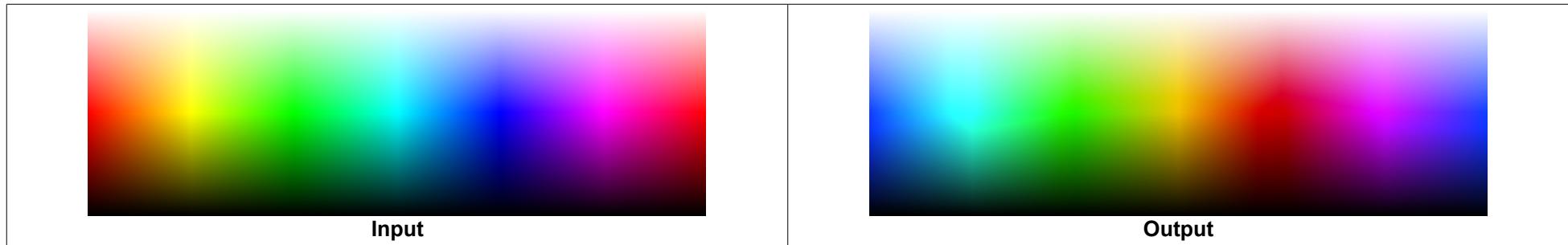
This filter swaps the U and V planes. This is an example. The input image is a spectrum with white at the top and black at the bottom:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -lavfi swapuv -y swapuv.png

pause
```

These are the input and output images:



2.26 Gradation curves

Note: This is obsolete. It's better to do this 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'

Example:

```
ffmpeg -i input.png -vf curves=psfile='curves.acv' -y out.png
pause
```

(1) <http://www.astro-electronic.de/GIMP2ACV.EXE>

2.27 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.

Specification of cube lut format: <https://wwwimages2.adobe.com/content/dam/acom/en/products/speedgrade/cc/pdfs/cube-lut-specification-1.0.pdf>

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:

```
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 512×512 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 $\text{LEVEL}^*\text{LEVEL}^*\text{LEVEL}$, at $\text{LEVEL}=8$ there are $64^*64^*64=262144$ colors and the image has $512^*512=262144$ 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 greater 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

```

2.28 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

pause
```

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. If the CLUT isn't required for correcting other videos, it can be deleted.

```
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
```

Note: The "haldclut" filter has an option "clut", which sets which frames are processed from second input stream, can be *first* or *all*. Default is *all*. If the second stream is an image, then this option doesn't care.

2.29 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:

Level n	Size of CLUT $n^3 \times n^3$	Typical file size of CLUT as 16-bit PNG	Number of points along the edge of the RGB cube n^2	Number of support points (= number of pixels) n^6	Distance between support points for 8-bit $255 / (n^2 - 1)$	Distance between support points for 16-bit $65535 / (n^2 - 1)$
2	8x8 Pixels	0.17 kB	4	64	85	21845
3	27x27 Pixels	1.2 kB	9	729	31.87	8191.87
4	64x64 Pixels	16.4 kB	16	4096	17	4369
5	125x125 Pixels	61.9 kB	25	15625	10.62	2730.62
6	216x216 Pixels	179 kB	36	46656	7.29	1872.43
7	343x343 Pixels	436 kB	49	117649	5.31	1365.31
8	512x512 Pixels	0.97 MB	64	262144	4.05	1040.24
9	729x729 Pixels		81	531441	3.19	819.19
10	1000x1000 Pixels		100	1000000	2.58	661.97
11	1331x1331 Pixels		121	1771561	2.12	546.12
12	1728x1728 Pixels		144	2985984	1.78	458.29
13	2197x2197 Pixels		169	4826809	1.52	390.09
14	2744x2744 Pixels		196	7529536	1.31	336.08
15	3375x3375 Pixels		225	11390625	1.14	292.57
16	4096x4096 Pixels		256	16777216	1	257

The size of *.cube files is as follows:

LUT_3D_SIZE n	Number of points n^3	Typical file size	Distance between support points for 8-bit 255 / (n-1)	Distance between support points for 16-bit 65535 / (n-1)
2	8		255	65535
3	27		127.5	32767.5
5	125		63.75	16383.75
9	729		31.875	8191.875
17	4913		15.937	4095.937
25	15625		10.625	2730.625
33	35937	1 MB	7.969	2047.969
64	262144	7 MB	4.047	1040.238
65	274625	7 MB	3.984	1023.984

*.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:

```
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:

```
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
0.0 0.5 0.5
# gray
0.5 0.5 0.5
# red-white
1.0 0.5 0.5
# green-cyan
0.0 1.0 0.5
# green-white
0.5 1.0 0.5
# yellow-white
1.0 1.0 0.5
# blue
0.0 0.0 1.0
# blue-magenta = violet
0.5 0.0 1.0
# magenta
1.0 0.0 1.0
# cyan-blue
0.0 0.5 1.0
# blue-white
0.5 0.5 1.0
# magenta-white
1.0 0.5 1.0
# cyan
0.0 1.0 1.0
# cyan-white
0.5 1.0 1.0
# white
1.0 1.0 1.0
```

My C# source code for creating *.cube LUT files can be downloaded here: http://www.astro-electronic.de/source/My_LUT_Creator.zip
(In this case it's a simple gamma function)

2.30 Convert a *.cube LUT to a halclut file

In this example it's shown how a halclut file can be generated from a *.cube LUT. Applying either of these two LUTs to an image gives almost the same results out1.png and out2.png. In the last command line the difference between the two results is calculated. If you try different values for the halclut level option, you see that in most cases level 5 or 6 is sufficient. This depends also on the size of the *.cube file.

```
rem Apply a *.cube lut to the image:  
  
ffmpeg -i test.jpg -vf lut3d="VLog_to_V709.cube" -y out1.png  
  
rem Convert the *.cube lut to a halclut image:  
  
ffmpeg -f lavfi -i halclutsrc=8 -vf lut3d="VLog_to_V709.cube" -frames 1 -y halclut.png  
  
rem Apply the halclut file to the image:  
  
ffmpeg -i test.jpg -i halclut.png -lavfi halclut -y out2.png  
  
rem Show the difference between the two results:  
  
ffmpeg -i out1.png -i out2.png -lavfi blend=all_mode=grainextract -y diff.png  
  
pause
```

Note: The "lut3d" filter accepts LUTs in the following file formats:

- *.3dl (AfterEffects)
- *.cube (Iridas)
- *.dat (DaVinci Resolve)
- *.m3d (Pandora)
- *.csp (cineSpace)

2.31 Colorchart source

The "colorchart" video source generates an image with 24 colors, similar to the X-Rite ColorChecker.

```
ffmpeg -f lavfi -i colorchart=preset=reference -frames 1 -y reference.png
pause
```

These are the colors and RGB values in the output image.

First row: RGB values from the "colorchart=preset=reference" source, which are identical to the values listed for the X-Rite ColorChecker.

Second row: RGB values printed on the back side of the cheap chinese ColorChecker, which has slightly different colors (Source:

<https://www.aliexpress.com/item/32738129067.html>).

Third row: RGB values of the cheap chinese ColorChecker, as measured by comparing with a new "Calibrite ColorChecker Classic Mini".

dark skin 115, 82, 68 115, 82, 69 120, 95, 82	light skin 194, 150, 130 204, 161, 141 202, 159, 136	blue sky 98, 122, 157 101, 134, 179 102, 130, 163	foliage 87, 108, 67 89, 109, 61 102, 117, 85	blue flower 133, 128, 177 141, 137, 194 159, 168, 190	bluish green 103, 189, 170 132, 228, 208 127, 206, 186
orange 214, 126, 44 249, 118, 35 247, 146, 83	purple red 80, 91, 166 80, 91, 182 86, 97, 174	moderate red 193, 90, 99 222, 91, 125 232, 119, 131	purple 94, 60, 108 91, 63, 123 108, 76, 130	yellow green 157, 188, 64 173, 232, 91 181, 202, 106	orange yellow 224, 163, 46 255, 164, 26 252, 179, 60
blue 56, 61, 150 44, 56, 142 67, 70, 94	green 70, 148, 73 74, 148, 81 69, 146, 81	red 175, 54, 60 179, 42, 50 187, 78, 84	yellow 231, 199, 31 250, 226, 21 244, 213, 35	magenta 187, 86, 149 191, 81, 160 194, 93, 148	cyan 8, 133, 161 6, 142, 172 7, 138, 168
white 243, 243, 242 252, 252, 252 237, 241, 243	neutral 80 200, 200, 200 230, 230, 230 225, 229, 232	neutral 65 160, 160, 160 200, 200, 200 169, 171, 175	neutral 50 122, 122, 121 143, 143, 142 130, 132, 136	neutral 35 85, 85, 85 100, 100, 100 97, 98, 102	black 52, 52, 52 50, 50, 50 63, 64, 64

See also: https://xritephoto.com/documents/literature/en/ColorData-1p_EN.pdf

See also: <https://en.wikipedia.org/wiki/ColorChecker>

This batch file generates a test image with the RGB values from the back side of the cheap chinese ColorChecker:

```
rem Make a test image with RGB values for chinese colorchecker:  
  
ffmpeg ^  
-f lavfi -i color=0x735245:s=64x64 -f lavfi -i color=0xccca18d:s=64x64 -f lavfi -i color=0x6586b3:s=64x64 ^  
-f lavfi -i color=0x596d3d:s=64x64 -f lavfi -i color=0x8d89c2:s=64x64 -f lavfi -i color=0x84e4d0:s=64x64 ^  
-f lavfi -i color=0xf97623:s=64x64 -f lavfi -i color=0x505bb6:s=64x64 -f lavfi -i color=0xde5b7d:s=64x64 ^  
-f lavfi -i color=0x5b3f7b:s=64x64 -f lavfi -i color=0xade85b:s=64x64 -f lavfi -i color=0xffa41a:s=64x64 ^  
-f lavfi -i color=0x2c388e:s=64x64 -f lavfi -i color=0x4a9451:s=64x64 -f lavfi -i color=0xb32a32:s=64x64 ^  
-f lavfi -i color=0xfae215:s=64x64 -f lavfi -i color=0xbf51a0:s=64x64 -f lavfi -i color=0x068eac:s=64x64 ^  
-f lavfi -i color=0xfcfcfc:s=64x64 -f lavfi -i color=0xe6e6e6:s=64x64 -f lavfi -i color=0xc8c8c8:s=64x64 ^  
-f lavfi -i color=0x8f8f8e:s=64x64 -f lavfi -i color=0x646464:s=64x64 -f lavfi -i color=0x323232:s=64x64 ^  
-lavfi [0][1][2][3][4][5]hstack=6[a];[6][7][8][9][10][11]hstack=6[b];^  
[12][13][14][15][16][17]hstack=6[c];[18][19][20][21][22][23]hstack=6[d];[a][b][c][d]vstack=4 -frames 1 -y china.png  
  
pause
```

This batch file generates a test image with the RGB values for the cheap chinese ColorChecker, as per my own calibration:

```
rem Make a test image with RGB values for chinese colorchecker:  
  
ffmpeg ^  
-f lavfi -i color=0x785f52:s=64x64 -f lavfi -i color=0xca9f88:s=64x64 -f lavfi -i color=0x6682a3:s=64x64 ^  
-f lavfi -i color=0x667555:s=64x64 -f lavfi -i color=0x9fa8be:s=64x64 -f lavfi -i color=0x7fceba:s=64x64 ^  
-f lavfi -i color=0xf79253:s=64x64 -f lavfi -i color=0x5661ae:s=64x64 -f lavfi -i color=0xe87783:s=64x64 ^  
-f lavfi -i color=0x6c4c82:s=64x64 -f lavfi -i color=0xb5ca6a:s=64x64 -f lavfi -i color=0xfcdb33c:s=64x64 ^  
-f lavfi -i color=0x43465e:s=64x64 -f lavfi -i color=0x459251:s=64x64 -f lavfi -i color=0xbb4e54:s=64x64 ^  
-f lavfi -i color=0xf4d523:s=64x64 -f lavfi -i color=0xc25d94:s=64x64 -f lavfi -i color=0x078aa8:s=64x64 ^  
-f lavfi -i color=0xedf1f3:s=64x64 -f lavfi -i color=0xe1e5e8:s=64x64 -f lavfi -i color=0xa9abaf:s=64x64 ^  
-f lavfi -i color=0x828488:s=64x64 -f lavfi -i color=0x616266:s=64x64 -f lavfi -i color=0x3f4040:s=64x64 ^  
-lavfi [0][1][2][3][4][5]hstack=6[a];[6][7][8][9][10][11]hstack=6[b];^  
[12][13][14][15][16][17]hstack=6[c];[18][19][20][21][22][23]hstack=6[d];[a][b][c][d]vstack=4 -frames 1 -y china.png  
  
pause
```

2.32 Colormap filter

The colormap filter can be used for correcting the colors, so that they fit to the colors of the ColorChecker:

```
rem Make the reference image:  
ffmpeg -f lavfi -i colorchart=preset=reference -frames 1 -y reference.png  
  
rem Now edit the reference image in image processing software. Change brightness,  
rem contrast, gamma, hue and saturation.  
rem Draw some additional lines in the image, but keep the centers of the patches unaffected.  
rem Save the image as "modified.png"  
  
rem Now try to reconstruct the original colors from the modified image:  
ffmpeg -i modified.png -i modified.png -i reference.png -lavfi colormap=nb_patches=24 -frames 1 -y out.png  
  
rem Show the difference between the two images. It should be a uniform gray surface,  
rem if all colors have been reconstructed successfully. The additional lines must be visible.  
ffmpeg -i out.png -i reference.png -lavfi blend=all_mode=grainextract -y diff.png  
pause
```

Note: It's possible that "colormap" fails without error message, for example if two colors in the "source" image are identical, while the corresponding colors in the target image are different. It's impossible to map one source color to two different target colors. In this case the output of the "colormap" filter is a copy of the first input.

Note: The second (source) and third (target) input of the colormap filter must have the same size. The first input can have any size.

Note: The "colormap" filter picks only the central pixels from the patches. There is no averaging done in this filter. That means some preprocessing is required for images of real-world colorcheckers, especially geometric transformation (for example with "perspective" filter) and averaging with a blurring filter. See next example.

Note: The default value of the "patch_size" option is 64x64, which is also the default patch size in the colorchart source. So it's not necessary to specify

them.

Note: The option "nb_patches" is the number of patches that are used. By default it equals the number of available patches. If "nb_patches" is smaller than the number of available patches, then the patches are used line-wise from left to right, beginning in the top left corner.

This is a real-world example for color mapping. Geometric transformation is applied to the image of the ColorChecker, and then the image is scaled to the same size as the reference image. It is also blurred to improve color accuracy. Finally the colors of the input image are mapped to the correct output colors.

```
rem Take a picture of the ColorChecker and measure the corner coordinates

set "X0=1656"      :: Top left corner (dark skin)
set "Y0=691"
set "X1=4116"      :: Top right corner (bluish green)
set "Y1=1226"
set "X2=1269"      :: Bottom left corner (white)
set "Y2=2316"
set "X3=3796"      :: Bottom right corner (black)
set "Y3=2870"

rem Apply geometric transformation, scale to 384x256 and use avgblur for averaging:

ffmpeg -i image.jpg -lavfi format=argb,perspective=x0=%X0%:y0=%Y0%:x1=%X1%:y1=%Y1%:x2=%X2%:y2=%Y2%:x3=%X3%:y3=%Y3%,scale=384x256,avgblur=10 -y source_colors.png

rem Make the reference image (this is the perfect ColorChecker image):

ffmpeg -f lavfi -i colorchart=preset=reference -frames 1 -y reference.png

rem Now reconstruct the true colors of the input image:

ffmpeg -i image.jpg -i source_colors.png -i reference.png -lavfi colormap=nb_patches=24 -frames 1 -y out.png

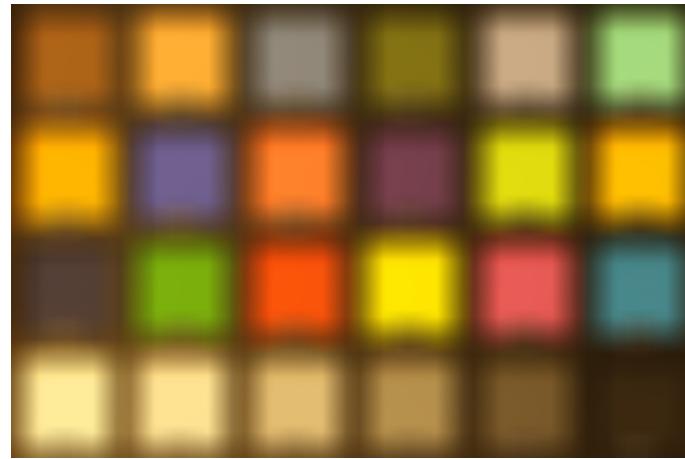
rem Calculate the difference between output and input:

ffmpeg -i out.png -i image.jpg -lavfi blend=all_mode=grainextract -y diff.png

pause
```



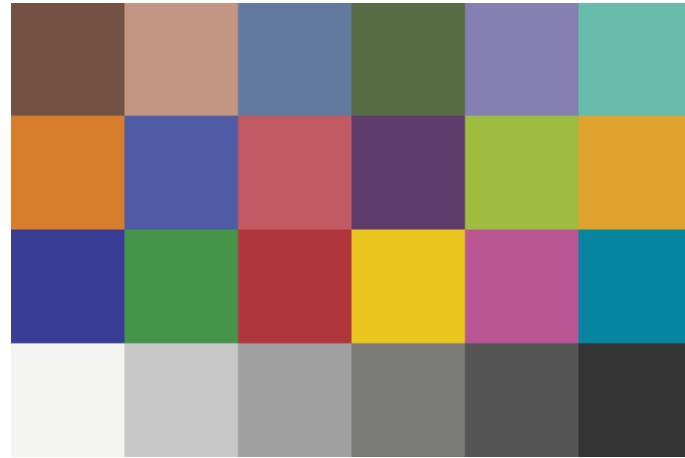
This is the input image with wrong colors because of fluorescent lamp.



This is the extracted "source_colors.png" image.



This is the corrected output image.



This is the "reference.png" image which contains the target colors.

Note: It's possible to speed up filtering by using the "haldclutsrc" source and "haldclut" filter. First apply the "colormap" filter to a "haldclutsrc" image, and then filter the image or video with the "haldclut" filter.

2.33 Colorhold, chromahold and hsvhold filters

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  
  
pause
```

Output of this example:



There is also a "chromahold" filter which is similar to the "colorhold" filter but works in YUV range.

There is also a "hsvhold" filter, which is similar to the "colorhold" filter but works in the HSV (Hue-Saturation-Value) range:

```
ffmpeg -i 7Z7A2027.jpg -filter_complex split[1][2];[1]hsvhold=hue=45:sat=0.7:val=0.5:similarity=0.30:blend=0[3];[2][3]hstack -y hsv.jpg  
pause
```

Output of this example:

The problem with these filters is that it's difficult to adjust the parameters properly, without seeing the result in real time.

Hint: Use the "FastStone Image Viewer" for showing the result image. This viewer does automatically detect when the image is overwritten by a new image, and shows the new image automatically.

2.34 Atmospheric dispersion correction

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

```
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%

pause
```

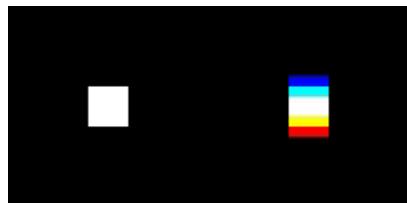
This example shows a white square before and after applying the correction:

```
set "RV=5"                  :: Vertical shift of red channel
set "BV=-5"                 :: Vertical shift of blue channel

ffmpeg -f lavfi -i color=black:s=100x100 -lavfi drawbox=color:white:x=40:y=40:w=20:h=20:t=fill,split[a][b];
[b]rgbashift=rv=%RV%:bv=%BV%[c];[a][c]hstack -frames 1 -y out.png

pause
```

This is the output image:



The chromashift filter seems to be similar, but I haven't yet tested it.

2.35 Amplify filter

The "amplify" filter amplifies differences between adjacent frames. Good for motion detection, but it's also sensitive to noise.

2.36 Sharpen or blur images

Images or videos can be blurred or sharpened with the "unsharp" filter:

```
ffmpeg -i 7z7a1256.jpg -vf unsharp=la=2 -y out.jpg  
pause
```

These are the parameters of the "unsharp" filter:

Parameter	Default value	Description
lx	5	Luma matrix horizontal size, it must be an odd integer between 3 and 23.
ly	5	Luma matrix vertical size, it must be an odd integer between 3 and 23.
la	1	Luma effect strength, it must be a floating point number, reasonable values are between -1.5 and 1.5. Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.
cx	5	Chroma matrix horizontal size, it must be an odd integer between 3 and 23.
cy	5	Chroma matrix vertical size, it must be an odd integer between 3 and 23.
ca	0	Chroma effect strength, it must be a floating point number, reasonable values are between -1.5 and 1.5. Negative values will blur the input video, while positive values will sharpen it, a value of zero will disable the effect.

For blurring you can also use the "avgblur" filter.

For variable blur you can use the "varblur" filter:

```
rem Create a circular mask:  
ffmpeg -f lavfi -i nullsrc=size=vga -lavfi format=gray8,geq='25*lt(hypot(X-W/2,Y-H/2),200)' -frames 1 -y mask.png  
  
rem Apply variable blurring:  
ffmpeg -f lavfi -i testsrc2=size=vga -i mask.png -lavfi [0]format=gbrp[a];[a][1]varblur=max_r=25 -t 10 -y out.mp4  
pause
```

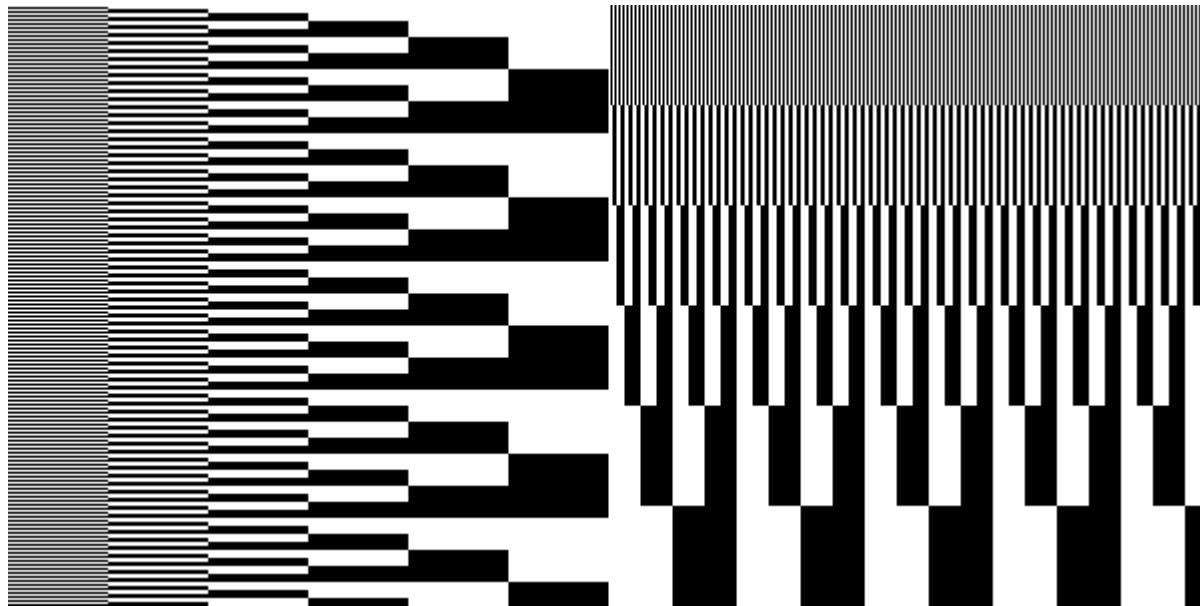
For directional blur, see the "dblur" filter.

2.37 FFT Filtering

First let's make a test image which contains different wavelengths, both horizontal and vertical:

```
ffmpeg -f lavfi -i color=black:s=300x50 -lavfi drawgrid=c=white:y=-1:w=2:h=51,split[a][b];^  
[b]crop=iw/2:x=0,scale=2*iw:ih:flags=neighbor,split[b][c];^  
[c]crop=iw/2:x=0,scale=2*iw:ih:flags=neighbor,split[c][d];^  
[d]crop=iw/2:x=0,scale=2*iw:ih:flags=neighbor,split[d][e];^  
[e]crop=iw/2:x=0,scale=2*iw:ih:flags=neighbor,split[e][f];^  
[f]crop=iw/2:x=0,scale=2*iw:ih:flags=neighbor[f];^  
[a][b][c][d][e][f]vstack=6,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png  
  
pause
```

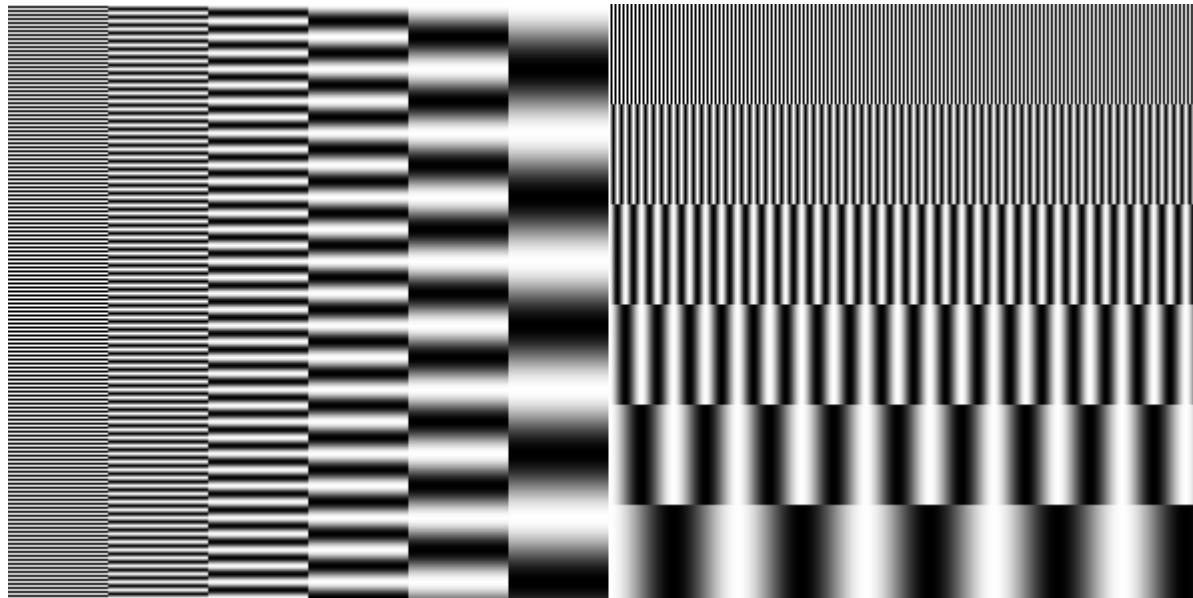
This is the test image. The wavelengths are 2, 4, 8, 16, 32 and 64 pixels per linepair:



After some experimenting I found out that it's better to use sine waves:

```
ffmpeg -f lavfi -i color=black:s=300x300 -lavfi geq='r=127.5+127.5*cos(X*PI/(pow(2,trunc(Y/50))))',colorchannelmixer=1:0:0:0:1:0:0:1:0:0:0,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png
```

This is the test image. The wavelengths are 2, 4, 8, 16, 32 and 64 pixels per linepair:



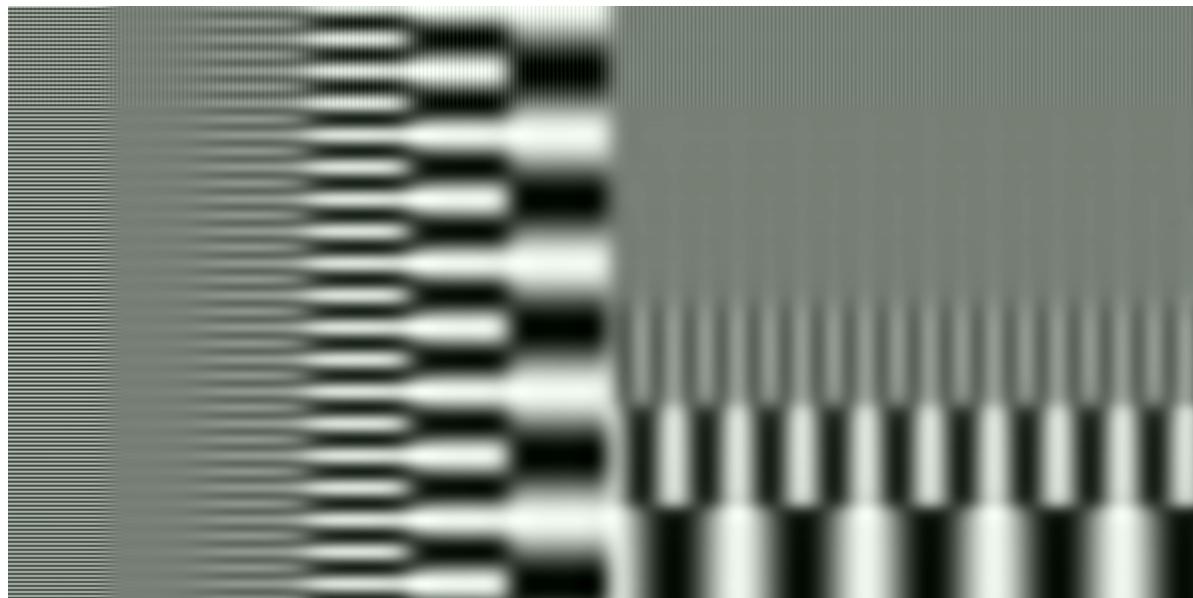
Now let's test the lowpass example from the official documentation:

```
ffmpeg -f lavfi -i color=black:s=300x300 -lavfi geq='r=127.5+127.5*cos(X*PI/(pow(2,trunc(Y/50))))',colorchannelmixer=1:0:0:0:1:0:0:1:0:0:0,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png

ffmpeg -i test.png -vf fftfilt=dc_Y=0:weight_Y='squish((Y+X)/100-1)' -y out1.png

pause
```

This is the output image:



It's clearly visible that the horizontal filtering frequency is different from the vertical filtering frequency. That's because the image has a 2:1 aspect ratio, but in the expression the X and Y values are simply added with the same weight.

It's also visible that the highest frequency isn't fully filtered out. This problem can be solved by scaling the image to double size before filtering and scaling down to the original size after filtering.

If the `weight_U` and `weight_V` expressions aren't set, they are by default copied from `weight_Y`. That's why the output image has a greenish tint in this example. To solve this problem, you should always set `weight_U` and `weight_V` to 1.

This is an infinitely sharp lowpass filter which has the same cutoff frequencies in both directions:

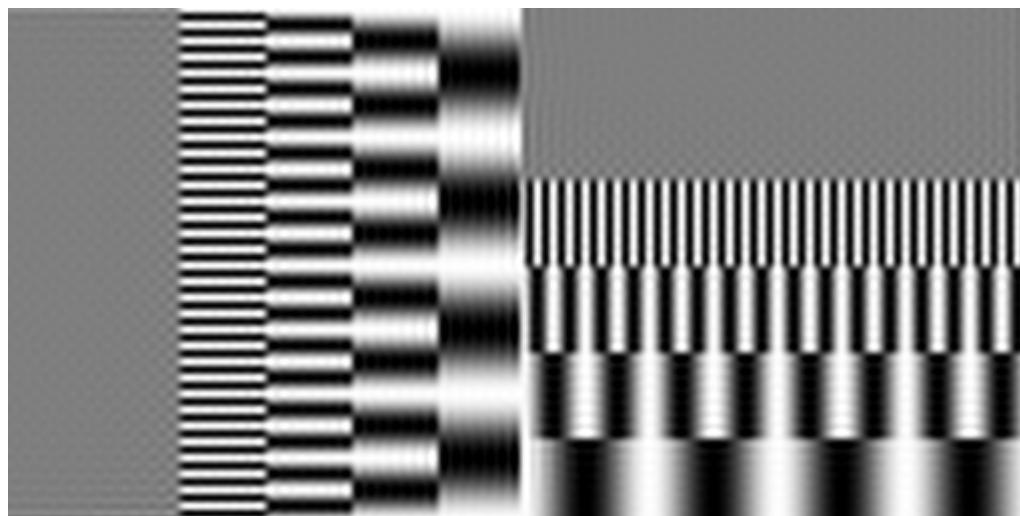
```
ffmpeg -f lavfi -i color=black:s=300x300 -lavfi geq='r=127.5+127.5*cos(X*PI/(pow(2,trunc(Y/50))))',colorchannelmixer=1:0:0:0:1:0:0:1:0:0:0,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png

ffmpeg -i test.png -vf
scale=2*iw:2*ih,fftfilt=dc_y=0:dc_U=0:dc_V=0:weight_Y='lt(hypot(Y/H,X/W),0.333)':weight_U=1:weight_V=1,scale=iw/2:ih/2
-y out2.png

pause
```

Note: In the above example, the "colorchannelmixer" copies the red channel to the green and blue channels.

This is the output image:

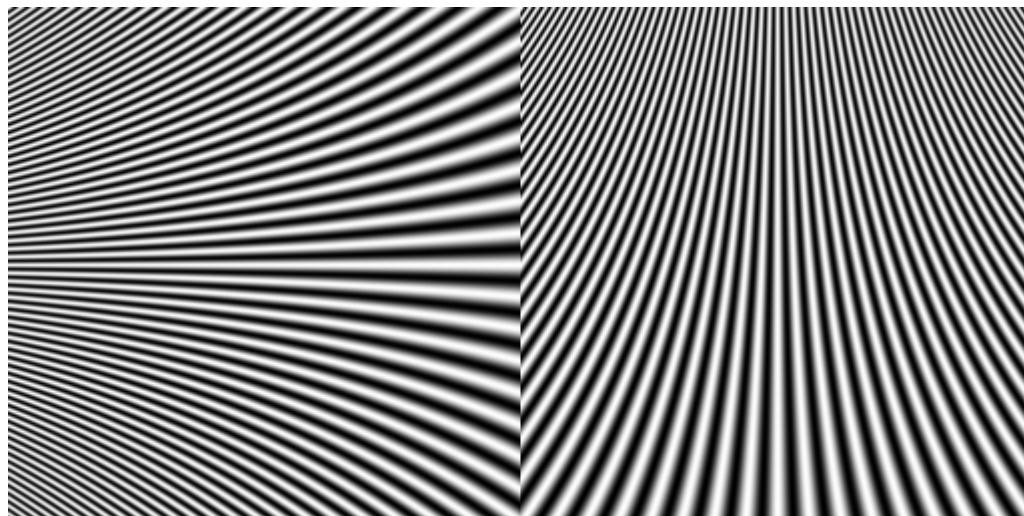


Note: The "fftfilt" filter works only in YUV pixel format.

It's better to use sine intensity profiles for testing, because square waves contain many harmonics. This command creates a test image where the wavelength changes continuously from 4 to 8 (in the center) to 16 pixels per linepair. The size can be changed, but width and height of the "color" source must be equal, because otherwise the hstack at the end filter would fail:

```
ffmpeg -f lavfi -i color=black:s=256x256 -lavfi geq='r=127.5+127.5*cos((X-W/2)*PI/(pow(2,(1+2*Y/H))))',colorchannelmixer=1:0:0:0:1:0:0:0:1:0:0:0,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png  
pause
```

This is the test image:



Note: The internal FFT array size changes between image size=230 and image size=232, as can be shown by the following example. This is probably because the FFT size is always a power of 2 and it's chosen at least about 10% bigger than the input size because a window function is used.

See also vf_fftfilt.c line 285: for (rdft_hbits = 1; 1 << rdft_hbits < w*10/9; rdft_hbits++);

```
set "P=8"

ffmpeg -f lavfi -i color=black:s=230x230 -lavfi geq='r=127.5+127.5*cos((X-W/2)*PI/(pow(2,(1+2*Y/H))))',colorchannelmixer=1:0:0:0:1:0:0:1:0:0:0,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png

ffmpeg -i test.png -vf scale=2*iw:2*ih,fftfilt=dc_Y=128:dc_U=1:dc_V=1:weight_Y='between(hypot(Y/H,X/W),1.9/%P%,2.1/%P%):weight_U=1:weight_V=1,scale=iw/2:ih/2 -y out230.png

ffmpeg -f lavfi -i color=black:s=232x232 -lavfi geq='r=127.5+127.5*cos((X-W/2)*PI/(pow(2,(1+2*Y/H))))',colorchannelmixer=1:0:0:0:1:0:0:1:0:0:0,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png

ffmpeg -i test.png -vf scale=2*iw:2*ih,fftfilt=dc_Y=128:dc_U=1:dc_V=1:weight_Y='between(hypot(Y/H,X/W),1.9/%P%,2.1/%P%):weight_U=1:weight_V=1,scale=iw/2:ih/2 -y out232.png

pause
```

The size of the FFT arrays in the "fftfilt" filter is accessible via the "WS" and "HS" variables, or could be calculated as follows:

```
set "ARRAY_H=pow(2,ceil(log(ceil(W*10/9))/log(2)))"      :: horizontal fft array size
set "ARRAY_V=pow(2,ceil(log(ceil(H*10/9))/log(2)))"      :: vertical fft array size
```

Example for lowpass, highpass, bandpass and notch filters, where the filter wavelength is independent of the image size:

```
set "P=8"          :: filter wavelength = pixels per linepair
set "SIZE=512"     :: vertical size of test image

:: create test image, wavelength varies continuously from 4 to 8 (in the center) to 16 pixels per linepair:
ffmpeg -f lavfi -i color=black:ss=%SIZE%*x%SIZE% -lavfi geq='r=127.5+127.5*cos((X-W/2)*PI/(pow(2,(1+2*Y/H))))',colorchannelmixer=1:0:0:0:1:0:0:0:1:0:0:0,split[h][v];[v]transpose[v];[v][h]hstack -frames 1 -y test.png

:: lowpass, highpass, bandpass and notch filtering:

ffmpeg -i test.png -vf scale=2*iw:2*ih,fftfilt=dc_Y=0:dc_U=0:dc_V=0:weight_Y='lte(hypot(X/WS,Y/HS),1.0/%P%)':weight_U=1:weight_V=1,scale=iw/2:ih/2 -y lowpass.png

ffmpeg -i test.png -vf scale=2*iw:2*ih,fftfilt=dc_Y=128:dc_U=1:dc_V=1:weight_Y='gte(hypot(X/WS,Y/HS),1.0/%P%)':weight_U=1:weight_V=1,scale=iw/2:ih/2 -y highpass.png

ffmpeg -i test.png -vf scale=2*iw:2*ih,fftfilt=dc_Y=128:dc_U=1:dc_V=1:weight_Y='between(hypot(X/WS,Y/HS),0.8/%P%,1.2/%P%)':weight_U=1:weight_V=1,scale=iw/2:ih/2 -y bandpass.png

ffmpeg -i test.png -vf scale=2*iw:2*ih,fftfilt=dc_Y=0:dc_U=0:dc_V=0:weight_Y='1-between(hypot(X/WS,Y/HS),0.8/%P%,1.2/%P%)':weight_U=1:weight_V=1,scale=iw/2:ih/2 -y notch.png

pause
```

Why is the image scaled up before filtering and scaled down after filtering? If the image contains the highest possible frequency (2 pixels per linepair), this frequency wouldn't be filtered out with a lowpass filter. I think that's because of the YUV subsampling. As a workaround the image is scaled up before filtering and scaled down after filtering.

2.38 Extract a time segment from a video

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%

pause
```

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

pause
```

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

For more informations about seeking, see also <https://trac.ffmpeg.org/wiki/Seeking>

Note: If -ss is written before the input file, the cut will be at the nearest keyframe but not at the accurate time. However if -ss is written after the input file, the cut will be at the accurate time but there may be an empty part at the beginning of the file, until the next keyframe.

2.39 Trim filter

Drop everything except the second minute of input:

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

Keep only the first second:

```
ffmpeg -i in.mp4 -vf trim=duration=1 out.mp4  
pause
```

See also <https://transang.me/practical-ffmpeg-commands-to-manipulate-a-video/>

2.40 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
```

2.41 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
```

2.42 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:

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

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

```
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  
pause
```

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

```
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,atrim=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  
pause
```

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:

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

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

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

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

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

Afade curves are shown on this wiki page: <https://trac.ffmpeg.org/wiki/AfadeCurves>

2.43 Crossfading

The different types of xfade crossfadings are shown on this wiki page:

<https://trac.ffmpeg.org/wiki/Xfade>

Both inputs must be constant frame-rate and have the same resolution, pixel format, framerate and timebase.

2.44 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:

```
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 greater 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=3"                  :: MP4 Quality, 1 is best Quality, 3 is normal, 31 is strongest compression

ffmpeg -i %INPUT% -vf crop=%CROP% -s %SIZE% -q:v %QU% -codec:v mpeg4 %OUTPUT%

pause
```

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. See next chapter.

2.45 Add borders to a video

Borders can be added with the "pad" filter. This example adds a black 40 pixel border at the bottom of the video, for example for writing text into it:

```
ffmpeg -i input.mp4 -vf pad=iw:ih+32 -y output.mp4  
pause
```

2.46 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.

See also "Use of 'geq' as 'zoompan' alternative":

https://hhsprings.bitbucket.io/docs/programming/examples/ffmpeg/manipulating_video_colors/use_of_geq_as_zoompan_alternative.html

(I'm still working on this chapter...)

2.47 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, greater than 1 = slow motion
set "QU=3"                   :: MP4 Quality, 1 is best Quality, 3 is normal, 31 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 of 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.

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:

setpts=0.5*PTS	Double speed
setpts=2.0*PTS	Half speed
setpts=PTS+x/(FR*TB) or tpad=x	Delay by x frames (assuming the framerate is constant)
setpts=PTS+x/TB or tpad=x/framerate	Delay by x seconds
setpts=PTS-STARTPTS	Start counting PTS from zero

See also these Wiki pages:

<https://trac.ffmpeg.org/wiki/How%20to%20speed%20up%20/%20slow%20down%20a%20video>

<https://trac.ffmpeg.org/wiki/ChangingFrameRate>

2.48 Slow motion or timelapse only for a segment of the video

See the comments for explanation.

```
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, greater 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

pause
```

2.49 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 "^\n[0]trim=0.0:3.2,setpts=(PTS-STARTPTS) [1];^\n[0]trim=3.2:3.6,setpts=(PTS-STARTPTS)/0.80 [2];^\n[0]trim=3.6:4.0,setpts=(PTS-STARTPTS)/0.60 [3];^\n[0]trim=4.0:6.0,setpts=(PTS-STARTPTS)/0.40 [4];^\n[0]trim=6.0:6.4,setpts=(PTS-STARTPTS)/0.60 [5];^\n[0]trim=6.4:6.8,setpts=(PTS-STARTPTS)/0.80 [6];^\n[0]trim=6.8:10.0,setpts=(PTS-STARTPTS) [7];^\n[1][2][3][4][5][6][7]concat=n=7:v=1" -y out.mp4\n\npause
```

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 "^\n[0]trim=0:1,setpts=(PTS-STARTPTS)/0.975 [1];^\n[0]trim=1:2,setpts=(PTS-STARTPTS)/0.925 [2];^\n[0]trim=2:3,setpts=(PTS-STARTPTS)/0.875 [3];^\n[0]trim=3:4,setpts=(PTS-STARTPTS)/0.825 [4];^\n[0]trim=4:5,setpts=(PTS-STARTPTS)/0.775 [5];^\n[0]trim=5:6,setpts=(PTS-STARTPTS)/0.725 [6];^\n[0]trim=6:7,setpts=(PTS-STARTPTS)/0.675 [7];^\n[0]trim=7:8,setpts=(PTS-STARTPTS)/0.625 [8];^\n[0]trim=8:9,setpts=(PTS-STARTPTS)/0.575 [9];^\n[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\n\npause
```

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.

```
rem > So, to compute the timestamp of a frame with variable speed:  
rem >  
rem > * Express your frame rate as a complete formula:  $t \rightarrow v$   
rem >  
rem > * Integrate it:  $t \rightarrow f$ .  
rem >  
rem > * Find the reciprocal:  $f \rightarrow t$ .  
rem  
rem Let's assume we have a 10s video and the framerate changes linearly from 20 at the beginning to 10 at the end:  
rem  $v = 20 - t$        $v(0) = 20$      $v(10) = 10$   
rem  
rem After integrating we get:  $f = 20 * t - 0.5 * t^2$   
rem  
rem The inverse function is:  $t = 20 - \sqrt{400 - 2 * f}$   
  
rem Create a test video with framerate=20 and length=10s:  
ffmpeg -f lavfi -i testsrc2=size=vga:duration=10:rate=20 -y test.mp4  
  
rem Apply the time remapping:  
ffmpeg -i test.mp4 -lavfi setpts='(20-sqrt(400-2*N))/TB' -y out.mp4  
  
pause
```

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.

Note: It's much easier to do time remapping in DaVinci Resolve.

Keywords for searching: "Time remapping", "Time ramp", "Slow motion ramp", "Speed ramp"

2.50 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
```

2.51 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=3"                   :: MP4 Quality, 1 is best Quality, 3 is normal, 31 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
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
```

```

set "DS=0.5"           :: Start time
set "DE=4.5"           :: End time
set "FID=1.0"          :: Fade-in duration (may be small, but not zero)
set "FOD=1.0"          :: Fade-out duration (may be small, but not zero)

set %NAME%="drawtext='fontfile=%FONT%:textfile=%TEXT%:fontcolor_expr=%COLOR%@%%{e\clip(((t-%DS%)/%FID%)*^
between(t,%DS%,(%DS%+%DE%)/2)+(%DE%-t)/%FOD%*between(t,(%DS%+%DE%)/2,%DE%),0,1)}:fontsize=%SIZE%:x=%POS_X%:y=%POS_Y%'

set "NAME=TEXT2"        :: 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=red"         :: Text color
set "SIZE=250"          :: Font size
set "POS_X=(w-tw)/2"    :: X position of text, use (w-tw)/2 for centering
set "POS_Y=(h-th)/3"    :: Y position of text, use (h-th)/2 for centering
set "DS=0.0"             :: Start time
set "DE=3.0"             :: End time
set "FID=0.5"            :: Fade-in duration (may be small, but not zero)
set "FOD=0.5"            :: Fade-out duration (may be small, but not zero)

set %NAME%="drawtext='fontfile=%FONT%:textfile=%TEXT%:fontcolor_expr=%COLOR%@%%{e\clip(((t-%DS%)/%FID%)*^
between(t,%DS%,(%DS%+%DE%)/2)+(%DE%-t)/%FOD%*between(t,(%DS%+%DE%)/2,%DE%),0,1)}:fontsize=%SIZE%:x=%POS_X%:y=%POS_Y%'

ffmpeg -i %INPUT% -vf "%TEXT1%,%TEXT2%" -q:v %QU% -codec:v mpeg4 -an -y %OUTPUT%

pause

```

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 EF_{hex} BB_{hex} BF_{hex}.

See also here: 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/>
Notepad++ has the unexpected behaviour that it always restores the last session when you start it. To disable this behaviour, go to Settings --> Preferences --> Backup and untick "Remember current session for next launch". Then it just opens a new empty file when you start it by double-clicking on the Notepad++ icon.
- 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.

2.52 Vertical alignment of text

Problem: In the "drawtext" filter, the content of the variable "text_h" depends on which characters are printed. For example, the characters "a", "A", "g", "_" and "^" do all have different heights.

Vertical position	What does it do?
<code>y=100</code>	The highest point of the string (the ascent) is used for vertical alignment. Warning: The vertical alignment depends on the content of the string.
<code>y=100-text_h</code>	The lowest point of the string (the descent) is used for vertical alignment. Warning: The vertical alignment depends on the content of the string.
<code>y=100-ascent</code>	The baseline of the string is used for alignment. The vertical alignment doesn't depend on the content of the string. This is the recommended method when you want to print several strings in the same line.

2.53 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,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%

pause
```

This batch file does the same thing and is simpler:

```
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:

```
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
```

2.54 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=3"                :: MP4 quality level, 1 is best quality, 3 is normal, 31 is strong compression
set "FPS=30"               :: Output Framerate
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
%*between(t,%S%,%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%

pause
```

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:

```

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.WriteLine("P2\n");
xmap.WriteLine("# Xmap file for fulldome remap \n");
xmap.WriteLine(a.ToString() + " " + a.ToString() + " \n");
xmap.WriteLine("65535\n");

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

```

This is a simpler example for generating curved text for fulldome projection, using the v360 filter:

```
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
```

2.55 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 text

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%

pause
```

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.

In this example the white square has a dynamically changing size, and "scale2ref" is used to scale the red input to the same size:

```
ffmpeg -f lavfi -i color=black:size=400x400 ^
-f lavfi -i color=white:size=100x100 ^
-f lavfi -i color=red ^
-lavfi [1]scale=width='100+n':height='100+n':eval=frame[w];[2][w]scale2ref=eval=frame[r][w];[r]null[r];[w][r]overlay[w];
[0][w]overlay -t 4 -y gif.mp4

pause
```

It's unclear why the "null" filter is required in this example. If the "null" filter is omitted, then it seems there a one frame delay so that the red square is one pixel too small, which is visible as a white border. See also <https://trac.ffmpeg.org/ticket/9680>

2.56 Combine multiple videos with concat demuxer

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

```
rem Final cut with concat demuxer

ffmpeg -f concat -i concat_list.txt -c copy -y MyVideo.mp4

pause
```

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

```
file text1.mp4          :: 10  Title: A year in the woods
file text2.mp4          :: 10  When and where
file Videos/scene20.mp4 :: 12  Live video in the wood
# This is a comment
file text22.mp4         :: 10  In 15 months...
file Videos/scene22.mp4 :: 52  Live video, camera
file text98.mp4         :: 10  the end
```

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 pixel format
- 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.

How to create a concat_list file which contains all *.mp4 files from a folder:

```
if exist concat_list.txt del concat_list.txt  
(for %%G in (*.mp4) do @echo file '%G') >> concat_list.txt  
pause
```

See also here: <https://trac.ffmpeg.org/wiki/Concatenate>

2.57 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%
pause
```

See also here: <https://trac.ffmpeg.org/wiki/Concatenate>

Note: Cutting the input videos to the required section can also be done with the "trim" filter.

The opposite of the "concat" filter is the "segment" filter, which splits a video into several streams.

2.58 The "fps" filter

This filter is described in detail on Jim DeLaHunt's website: <http://blog.jdlh.com/en/2020/04/30/ffmpeg-fps-documented/>

2.59 Split a video in multiple segments

A video can be split in multiple segments with the segment muxer. All segments will have the same length, except the last one.

```
set "IN=my_video.mov"      :: Input video
set "L=10"                  :: Segment length in seconds
set "OUT=out%%2d.mov"       :: Output filename

ffmpeg -i %IN% -f segment -segment_time %L% -c copy %OUT%

pause
```

This batch file extracts a segment with known start and end frame numbers:

```
set "start=100"            :: First frame number
set "end=200"               :: Last frame number

set /a startms=%start%*1001/30   :: This calculation is for framerate 30000/1001 = 29.97
set /a endms=(%end%+1)*1001/30    :: Note that in the batch file only integer arithmetic is possible!
                                    :: It's important to do first the multiplication and then the division

ffmpeg -i in.mp4 -ss %startms%ms -to %endms%ms -c copy -y out.mp4

pause
```

Note: The above command line with "-c copy" works only for intraframe codecs, meaning that all frames are I-frames. For interframe codecs you must remove "-c copy", but then the video will be re-encoded and the process is much slower.

2.60 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  
  
pause
```

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:

	$0 < t < 2$	$2 < t < 4$	$4 < t < 6$
Video	from video1 (0...2)	from video2 (2...4)	from video1 (4...6)
Audio	from video1 (0...2)	from video1 (2...4)	from video1 (4...6)

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  
  
pause
```

In this case you get this output video:

	$0 < t < 2$	$2 < t < 4$	$4 < t < 6$
Video	from video1 (0...2)	from video2 (0...2)	from video1 (4...6)
Audio	from video1 (0...2)	from video1 (2...4)	from video1 (4...6)

2.61 Stack videos side by side (or on top of each other)

```
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%
pause
```

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

2.62 Horizontal and vertical flipping

This can be done with the "hflip" and "vflip" filters.

2.63 Stereo3d filter

The stereo3d filter can be used for splitting a square (fisheye) image in two halves (top and bottom) and stacking the halves side by side (left and right):
In this example a 4096x4096 fisheye image is splitted in two halves and the output size 8192x2048:

```
rem Test pattern from http://www.paulbourke.net/dome/testpattern/4096.png
ffmpeg -i 4096.png -vf stereo3d=abl:sbsl -y out1.png
pause
```

Note: If left/right images are swapped, use "sbsr" instead of "sbsl".

2.64 Stack four videos to a 2x2 mosaic

```
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%

pause
```

Other method using xstack:

```
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%

pause
```

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

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

pause
```

2.65 Blink comparator

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

```
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 1 -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:

```
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"                     :: Length 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 1 -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".

This is an example for toggling between two images or two video streams:

```
ffmpeg -f lavfi -i color=yellow -vf drawtext='text=1:fontcolor=red:fontsize=100:x=140:y=80' -frames 1 -y 1.png  
ffmpeg -f lavfi -i color=yellow -vf drawtext='text=2:fontcolor=red:fontsize=100:x=140:y=80' -frames 1 -y 2.png  
  
ffmpeg -loop 1 -i 1.png -loop 1 -i 2.png -lavfi blend=all_expr='if(lt(mod(T,2),1),A,B)' -t 10 -y out.mp4  
  
pause
```

Note: "blend" is slow.

It's also possible to toggle between two images or video streams with "sendcmd" and "streamselect", but it's quite complicated to escape the commas:

```
ffmpeg -f lavfi -i color=yellow -vf drawtext='text=1:fontcolor=red:fontsize=100:x=140:y=80' -frames 1 -y 1.png  
ffmpeg -f lavfi -i color=yellow -vf drawtext='text=2:fontcolor=red:fontsize=100:x=140:y=80' -frames 1 -y 2.png  
  
ffmpeg -loop 1 -i 1.png -loop 1 -i 2.png  
-lavfi "sendcmd=c='0 [expr] streamselect map '\''gte(mod(T\,2)\,1)'\''',streamselect=map=0" -t 10 -y out.mp4  
  
pause
```

Note: I have inserted a line feed in the command line only for clarity. Of course it must all be written in one line.

2.66 Creating animated GIF or PNG

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.

```
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%[p];[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>

For animated PNG (APNG) see: <https://stackoverflow.com/questions/43795518/using-ffmpeg-to-create-looping-apng>

2.67 Overlay an animated GIF over a video

```
set "BACK=background.MOV"      :: Background video
set "OVL=thumbsUp.gif"        :: Overlay video
set "OUT=out.mp4"            :: Output video
set "X=200"                   :: X position of overlay
set "Y=400"                   :: Y position of overlay
set "S=0.5"                   :: Size factor for overlay
set "T=10"                    :: Maximum length in seconds

ffmpeg -i %BACK% -ignore_loop 0 -i %OVL% -lavfi [1]scale=iw*%S%:ih*%S%[a];[0][a]overlay=x=%X%:y=%Y% -t %T% -y %OUT%

pause
```

Note: Use "-ignore_loop 0" if you want to loop the GIF, or remove this option if you want to play the GIF only one time.

2.68 Changing the size of an animated GIF

```
set "IN=thumbsUp.gif"      :: Input file
set "S=0.5"                  :: Size factor
set "OUT=out.gif"            :: Output file

ffmpeg -i %IN% -lavfi scale=iw*%S%:-1,split[a][b];[a]paletten=reserve_transparent=on:transparency_color=fffff[p];[b]
[p]paletteuse -y %OUT%

pause
```

The trick with paletten and paletteuse is required to keep the transparency color.

2.69 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 inserted
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
```

2.70 Blend filter

Unfortunately the FFmpeg documentation doesn't explain what all the modes do. So you have to look it up in the source code (the filename is "blend_modes.c") or in the wiki page. The default mode is "normal".

Mode	Function	Notes
normal	$A * \text{opacity} + B * (1 - \text{opacity})$	Output is a mix of A and B, the default opacity is 1
addition	$A + B$	Output is $(A+B)$ with an upper limit at white level
average	$(A + B) / 2$	Output is the arithmetic mean of A and B
subtract	$A - B$	Output is $(A-B)$ with a lower limit at black level
multiply	$A * B$	Output is the product of A by B. Both inputs are normalized to the [0...1] range before multiplication.
difference	$\text{abs}(A - B)$	Output is the absolute difference of A and B
grainextract	$50\%_{\text{gray_level}} + A - B$	Output is $(A-B)$, shifted to 50% gray level, with limits at black and white levels
darken	$\min(A, B)$	Output is the minimum of A and B
lighten	$\max(A, B)$	Output is the maximum of A and B
and	$A \& B$	Output is bitwise AND of A and B
or	$A B$	Output is bitwise OR of A and B
xor	$A ^ B$	Output is bitwise XOR of A and B
screen	$1 - (1 - A) * (1 - B)$	Output is inverse of product of inverted A by inverted B. Might be usable for clips which have no alpha channel, black becomes transparent?

This table contains only a subset of the modes. There are more of them.

For a comparison of all blend modes, see also: <https://trac.ffmpeg.org/wiki/Blend>

Note: When using one of the preset modes of the blend filter, don't forget to write "all_mode", as it's not the default!

Note: The "opacity" options aren't used if a user-defined expression is used. They are only used if one of the "mode" options is used. The default is "all_opacity=1", which means the full blend effect is applied. A smaller opacity value means the output is mixed with the first input stream. "all_opacity=0" means the effect is disabled and the first input stream is returned. But there is one exception from this rule: If the mode is "normal", then "all_opacity=1" returns the first stream and "all_opacity=0" returns the second stream.

This batch file can be used for showing the different modes of the blend filter:

```
ffmpeg -f lavfi -i color=s=256x256,geq=r='H-1-Y':g='H-1-Y':b='H-1-Y' -frames 1 -y test.png  
ffmpeg -i test.png -vf "split[a][b];[b]transpose[b];[a][b]blend=all_mode=harmonic,pseudocolor=preset=turbo" -y  
harmonic.png  
pause
```

See also: <http://oioiiooxiii.blogspot.com/2017/01/ffmpeg-generate-image-of-tiled-results.html>

Calculate the difference between two images. If the images are identical, the output is 50% gray:

```
ffmpeg -i image1.png -i image2.png -lavfi blend=all_mode=grainextract -y diff.png  
pause
```

Apply a 5 second sine-shaped crossfade to two videos:

```
ffmpeg -f lavfi -i color=red -f lavfi -i color=yellow -lavfi blend=all_expr='A*(0.5+0.5*sin(T*2*PI/5))+B*(0.5-  
0.5*sin(T*2*PI/5))' -t 30 -y out.mp4  
pause
```

2.71 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%

pause
```

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%

pause
```

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.

<code>gt(hypot(X-x0,Y-y0),radius)</code>	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 <code>gt()</code> function.
<code>hypot(X-x0,Y-y0)-radius</code>	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.
<code>256/width*(hypot(X-x0,Y-y0)-radius)</code>	Multiply by a factor 256 and divide by the width of the transition band (in pixels).
<code>128+256/width*(hypot(X-x0,Y-y0)-radius)</code>	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.
<code>clip(128+256/width*(hypot(X-x0,Y-y0)-radius),0,255)</code>	Finally you must clip to the 0...255 range (black to white). The result is a circular mask with a smooth edge.

2.72 Radial attenuation (for overhead sky in a planetarium)

In a fulldome planetarium it's a good idea to avoid large white areas in videos, because the bright light is reflected from the dome and illuminates the opposite side of the dome, where it reduces contrast. In this example is shown how the overhead sky can be attenuated with the "multiply" filter:

```
set "IN=gras.png"      :: Input image
set "SIZE=2880"        :: Image size
set "R1=500"           :: Inner radius in pixels
set "R2=1000"          :: Outer radius in pixels
set "DARK=90"          :: Brightness level inside of R1, 0 is black, 255 is original brightness
set "BRIGHT=255"        :: Brightness level outside of R2, 0 is black, 255 is original brightness

rem Create the mask file

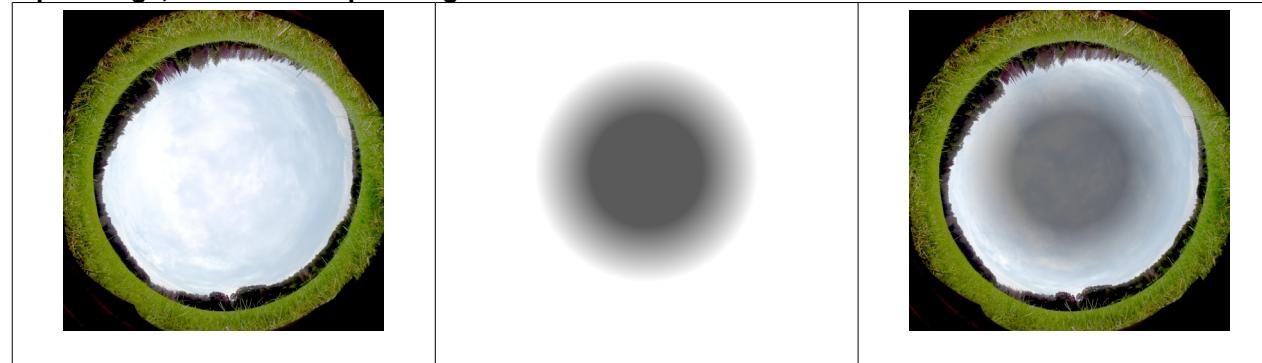
ffmpeg -f lavfi -i nullsrc=size=%SIZE%x%SIZE% -lavfi format=gray8,geq='clip(lerp(%DARK%,%BRIGHT%,(hypot(X-%SIZE%/2,Y-%SIZE%/2)-%R1%)/(%R2%-%R1%)),%DARK%,%BRIGHT%)',format=rgb24 -frames 1 -y mask.png

rem Apply the mask

ffmpeg -i %IN% -i mask.png -lavfi multiply=offset=0 -frames 1 -y out.png

pause
```

Input image, mask and output image:



With the same technique it's also possible to attenuate the corners of a fisheye image:

```
set "IN=gras.png"          :: Input image
set "SIZE=2880"            :: Image size
set "R3=1300"              :: Inner radius in pixels, full brightness inside of this radius
set "RAMP=200"              :: Width of the ramp in pixels

rem Create the mask file

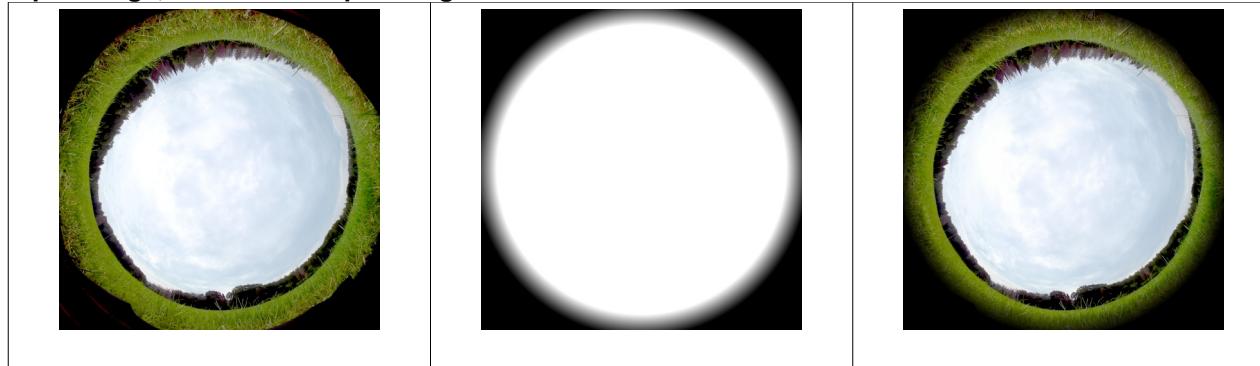
ffmpeg -f lavfi -i nullsrc=size=%SIZE%x%SIZE% -lavfi format=gray8,geq='clip(lerp(255,0,(hypot(X-%SIZE%/2,Y-%SIZE%/2)-%R3%)/(%RAMP%)),0,255)',format=rgb24 -frames 1 -y mask.png

rem Apply the mask

ffmpeg -i %IN% -i mask.png -lavfi multiply=offset=0 -frames 1 -y out.png

pause
```

Input image, mask and output image:



Both effects can be combined together in the same mask file:

```
set "IN=gras.png"      :: Input image
set "SIZE=2880"        :: Image size
set "R1=500"           :: Inside of radius R1 the value is DARK
set "R2=1000"          :: Between R1 and R2 the value is a ramp from DARK to 255
set "R3=1300"          :: Between R2 and R3 the value is 255 (original brightness)
set "R4=1500"          :: Between R3 and R4 the value is a ramp from 255 to 0
                      :: Outside of radius R4 the value is 0
set "DARK=90"          :: Brightness level in the center, 0 is black, 255 is original brightness

rem Create the mask file

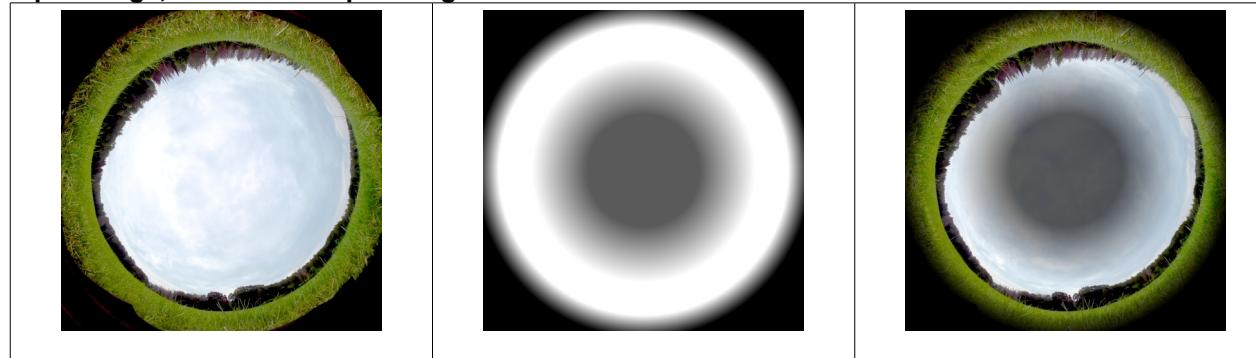
ffmpeg -f lavfi -i nullsrc=size=%SIZE%x%SIZE% -lavfi format=gray8,geq='st(0,hypot(X-%SIZE%/2,Y-%SIZE%/2));lte(ld(0),%R1%)*%DARK%+bitand(gte(ld(0),%R1%),lt(ld(0),%R2%))*lerp(%DARK%,255,(ld(0)-%R1%)/(%R2%-%R1%))+bitand(gte(ld(0),%R2%),lt(ld(0),%R3%))*255+bitand(gte(ld(0),%R3%),lt(ld(0),%R4%))*lerp(255,0,(ld(0)-%R3%)/(%R4%-%R3%))',format=rgb24 -frames 1 -y mask.png

rem Apply the mask

ffmpeg -i %IN% -i mask.png -lavfi multiply=offset=0 -frames 1 -y out.png

pause
```

Input image, mask and output image:



2.73 Edge Attenuation

```
rem Create a mask file with one 50% gray line at the edges
ffmpeg -f lavfi -i color=color=white:size=1920x1080 -lavfi drawbox=color=gray:t=1 -frames 1 -y mask.png

rem Or alternatively create a mask file with two 33% and 67% gray lines at the edges
ffmpeg -f lavfi -i color=color=white:size=1920x1080 -lavfi
drawbox=color=0x555555:t=1,drawbox=color=0xAAAAAA:x=1:y=1:w=iw-2:h=ih-2:t=1 -frames 1 -y mask.png

rem Apply the mask
ffmpeg -i in.png -i mask.png -lavfi multiply=offset=0 -frames 1 -y out.png
pause
```

2.74 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=0.05"                :: 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`.

This is an output image:



2.75 Vignetting

Vignetting at the edge of the image can be compensated with the "vignette" filter. "mode=backward" makes the corners brighter and "mode=forward" makes them darker. The value must be set so that the corners are neither too bright nor too dark.

Example:

```
ffmpeg -i input.png -vf vignette=a=0.5:mode=backward:x0=(w-1)/2:y0=(h-1)/2 -y out.png
```

```
pause
```

Note: The "a" value is clipped to the [0...pi/2] range.

Note: The default values for x0, y0 are w/2 and h/2. With these values, the vignetting effect isn't exactly centered in the frame. It's offset by 0.5 pixels. That's why you should always use x0=(w-1)/2 and y0=(h-1)/2.

This is the formula for forward mode: $\text{output} = \text{input} / \cos^4(a * \text{dist} / \text{dist_max})$

This is the formula for backward mode: $\text{output} = \text{input} * \cos^4(a * \text{dist} / \text{dist_max})$

with a = "angle" option dist = distance from the pixel to x0,y0 dist_max = half of the image diagonal

angle	Corner pixel in forward mode (input = 1)	Corner pixel in backward mode (input = 1)
PI/16	1.08	0.925
PI/8	1.37	0.723
PI/5 (default)	2.33	0.428
PI/4	4	0.25
PI/2	infinite	0

2.76 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 1024) frames from the dark video and save the darkframe lossless as 16-bit PNG:

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

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

```
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=all_mode=subtract" -y %OUT%
pause
```

2.77 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
```

2.78 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 persistance time.

Time_constant_in_seconds = $1 / ((1 - \text{decay}) * \text{framerate})$

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=3"               :: MP4 quality level, 1 is best quality, 3 is normal, 31 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=gbrp,lagfun=decay=%DECAY%:planes=1" -y %OUT%

pause
```

planes=	color
1	green
2	blue
3	cyan
4	red
5	yellow
6	magenta
7	white

2.79 Deflicker a video

This is an example for the deflicker filter:

```
rem Make a flickering video  
ffmpeg -f lavfi -i color=gray -vf geq=lum='128+mod(13*N,20):cb=128:cr=128' -t 10 -y flicker.mp4  
  
rem Deflicker the video  
ffmpeg -i flicker.mp4 -vf deflicker -y out.mp4  
  
pause
```

Note: A much better deflickered result can be obtained with DaVinci Resolve (ResolveFX Revival --> Deflicker). Surprisingly, the default "Timelapse" mode is better than the "Fluoro Light" mode, although in my example it clearly was flicker from fluorescent lights.

If the video was made with fluorescent light and rolling shutter, the deflickered video won't be acceptable because flicker isn't uniform over the whole frame. In this case it helps to split the video into many horizontal stripes, deflicker all of them separately, and then stitch the deflickered stripes together. This is an example for input size 1280x720:

```

set "S=10"      :: number of frames for deflicker filter
set "T=6"       :: duration

rem  This is the simple version without stripes:
ffmpeg -i in.mp4 -lavfi deflicker=%S% -t %T% -y deflicker1.mp4

rem  2 stripes:
ffmpeg -i in.mp4 -lavfi split=2[a][b];[a]crop=1280:360:0:0,deflicker=%S%[aa];[b]crop=1280:360:0:360,deflicker=%S%[bb];
[aa][bb]vstack=2 -t %T% -y deflicker2.mp4

rem  4 stripes:
ffmpeg -i in.mp4 -lavfi split=4[a][b][c][d];[a]crop=1280:180:0:0,deflicker=%S%[aa];[b]crop=1280:180:0:180,deflicker=%S%[bb];
[c]crop=1280:180:0:360,deflicker=%S%[cc];[d]crop=1280:180:0:540,deflicker=%S%[dd];[aa][bb][cc][dd]vstack=4 -t %T%
-y deflicker4.mp4

rem  8 stripes:
ffmpeg -i in.mp4 -lavfi split=8[a][b][c][d][e][f][g][h];[a]crop=1280:90:0:0,deflicker=%S%[aa];
[b]crop=1280:90:0:90,deflicker=%S%[bb];[c]crop=1280:90:0:180,deflicker=%S%[cc];[d]crop=1280:90:0:270,deflicker=%S%[dd];
[e]crop=1280:90:0:360,deflicker=%S%[ee];[f]crop=1280:90:0:450,deflicker=%S%[ff];[g]crop=1280:90:0:540,deflicker=%S%[gg];
[h]crop=1280:90:0:630,deflicker=%S%[hh];[aa][bb][cc][dd][ee][ff][gg][hh]vstack=8 -t %T% -y deflicker8.mp4

rem  15 stripes:
ffmpeg -i in.mp4 -lavfi split=15[a][b][c][d][e][f][g][h][i][j][k][l][m][n][o];[a]crop=1280:48:0:0,deflicker=%S%[aa];
[b]crop=1280:48:0:48,deflicker=%S%[bb];[c]crop=1280:48:0:96,deflicker=%S%[cc];[d]crop=1280:48:0:144,deflicker=%S%[dd];
[e]crop=1280:48:0:192,deflicker=%S%[ee];[f]crop=1280:48:0:240,deflicker=%S%[ff];[g]crop=1280:48:0:288,deflicker=%S%[gg];
[h]crop=1280:48:0:336,deflicker=%S%[hh];[i]crop=1280:48:0:384,deflicker=%S%[ii];[j]crop=1280:48:0:432,deflicker=%S%[jj];
[k]crop=1280:48:0:480,deflicker=%S%[kk];[l]crop=1280:48:0:528,deflicker=%S%[ll];[m]crop=1280:48:0:576,deflicker=%S%[mm];
[n]crop=1280:48:0:624,deflicker=%S%[nn];[o]crop=1280:48:0:672,deflicker=%S%[oo];[aa][bb][cc][dd][ee][ff][gg][hh][ii][jj]
[kk][ll][mm][nn][oo]vstack=15 -t %T% -y deflicker15.mp4

rem  Compare the simple deflickered video with the 15-stripes version:
ffmpeg -i deflicker1.mp4 -i deflicker15.mp4 -lavfi hstack -y out.mp4

pause

```

Note: Further improvement might be possible if the RGB channels are deflickered separately, because fluorescent light does also have color flicker.
Note: Why didn't I use 16 stripes instead of 15? Because $720 / 16 = 45$, that's an invalid odd number for the height of a video stream.

This is an example for splitting the video into 8 horizontal stripes and RBG colors and deflickering all 24 stripes separately. Unfortunately the result isn't better than the previous example.

```
set "S=10"      :: number of frames for deflicker filter
set "T=6"       :: duration

rem This is the simple version without stripes:
ffmpeg -i in.mp4 -lavfi deflicker=%S% -t %T% -y deflicker1.mp4

rem 8 stripes rgb:
ffmpeg -i in.mp4 -lavfi format=rgb24,extractplanes=r+g+b[r][g][b];
[r]split=8[r1][r2][r3][r4][r5][r6][r7][r8];
[g]split=8[g1][g2][g3][g4][g5][g6][g7][g8];
[b]split=8[b1][b2][b3][b4][b5][b6][b7][b8];
[r1]crop=1280:90:0:0,deflicker=%S%[r1d];[r2]crop=1280:90:0:90,deflicker=%S%[r2d];
[r3]crop=1280:90:0:180,deflicker=%S%[r3d];[r4]crop=1280:90:0:270,deflicker=%S%[r4d];
[r5]crop=1280:90:0:360,deflicker=%S%[r5d];[r6]crop=1280:90:0:450,deflicker=%S%[r6d];
[r7]crop=1280:90:0:540,deflicker=%S%[r7d];[r8]crop=1280:90:0:630,deflicker=%S%[r8d];
[r1d][r2d][r3d][r4d][r5d][r6d][r7d][r8d]vstack=8[rr];
[g1]crop=1280:90:0:0,deflicker=%S%[g1d];[g2]crop=1280:90:0:90,deflicker=%S%[g2d];
[g3]crop=1280:90:0:180,deflicker=%S%[g3d];[g4]crop=1280:90:0:270,deflicker=%S%[g4d];
[g5]crop=1280:90:0:360,deflicker=%S%[g5d];[g6]crop=1280:90:0:450,deflicker=%S%[g6d];
[g7]crop=1280:90:0:540,deflicker=%S%[g7d];[g8]crop=1280:90:0:630,deflicker=%S%[g8d];
[g1d][g2d][g3d][g4d][g5d][g6d][g7d][g8d]vstack=8[gg];
[b1]crop=1280:90:0:0,deflicker=%S%[b1d];[b2]crop=1280:90:0:90,deflicker=%S%[b2d];
[b3]crop=1280:90:0:180,deflicker=%S%[b3d];[b4]crop=1280:90:0:270,deflicker=%S%[b4d];
[b5]crop=1280:90:0:360,deflicker=%S%[b5d];[b6]crop=1280:90:0:450,deflicker=%S%[b6d];
[b7]crop=1280:90:0:540,deflicker=%S%[b7d];[b8]crop=1280:90:0:630,deflicker=%S%[b8d];
[b1d][b2d][b3d][b4d][b5d][b6d][b7d][b8d]vstack=8[bb];
[gg][bb][rr]mergeplanes=0x001020:gbrp -t %T% -y deflicker8rgb.mp4

rem Compare the simple deflickered video with the 8-stripes-rgb version:
ffmpeg -i deflicker1.mp4 -i deflicker8rgb.mp4 -lavfi hstack -y out.mp4

pause
```

Note: Line feeds were inserted only for clarity. Of course it must all be written in one command line.

2.80 Star trails

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

```
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=decay=1 -y startrail.mp4

pause
```

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:

```
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

pause
```

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

2.81 Bird trails

Paul Bourke did make a nice image of bird trails here: <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

pause
```

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

pause
```

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

```
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

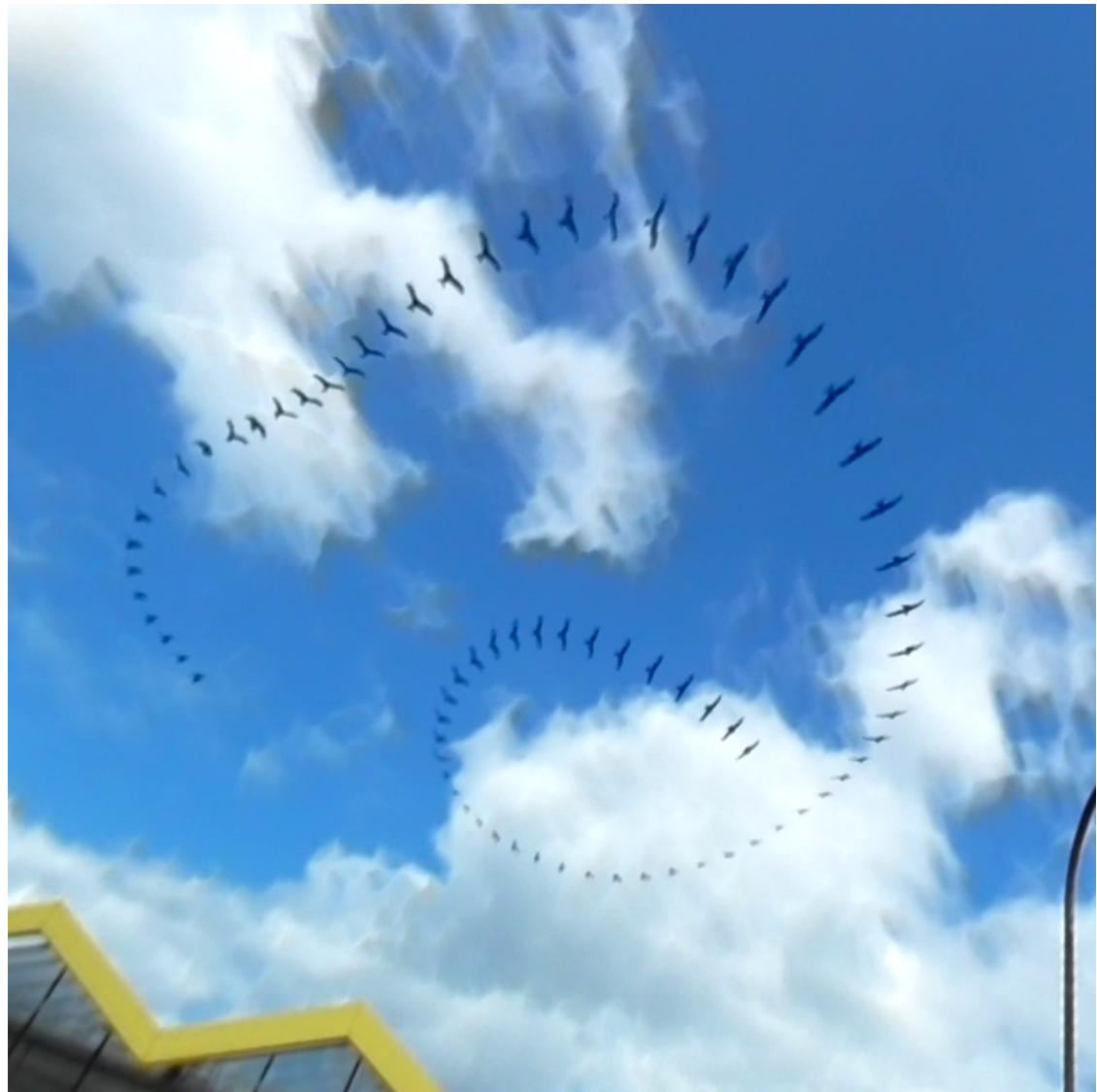
rem Make a bird trail video

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.

This is an image from the output video:



2.82 Rainbow-trail effect

I found the original version of this effect here: <http://oioiiooxiii.blogspot.com/2020/07/ffmpeg-improved-rainbow-trail-effect.html>

This is my version as a Windows batch file:

```
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=2:0:0:0:2:0:0:0:0:0:0:0"
set "ORANGE=colorchannelmixer=2:0:0:0:.5:0:0:0:0:0:0:0"
set "RED=colorchannelmixer=2:0:0:0:0:0:0:0:0:0:0:0"

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
%,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
```

2.83 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.

```
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][a1][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://oioiiooixiii.blogspot.com/2017/11/ffmpeg-temporal-slice-stacking-effect.html>

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

```
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:

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

pause
```

Or with tpad filter:

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

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  
pause
```

2.84 Extract and merge planes, split planes

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

```
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%  
  
pause
```

Use different delay for RGB planes:

```
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%  
  
pause
```

2.85 Extract the alpha channel

```
rem Make a short test video with alpha channel
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=3 -lavfi format=rgba,geq=a='255*gt(X,Y)':r='r(X,Y)':g='g(X,Y)':b='b(X,Y)' -c:v prores_ks -y test.mov

rem Extract the RGB channels
ffmpeg -i test.mov -pix_fmt rgb48le -y rgb.mov

rem Extract the alpha channel
ffmpeg -i test.mov -lavfi extractplanes=a -y alpha.mov
pause
```

Note: alpha=0 means transparent, alpha=255 means opaque

2.86 Colorkey

This filter works in RGB colorspace. The color is defined as a RGB color and a similarity value.

Example for "colorkey" filter:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -f lavfi -i color=lightblue:size=384x128 -lavfi [0]colorkey=color=orange:similarity=0.1[a];[1][a]overlay -frames 1 -y out.png

pause
```

These are the input and output images:



Input



Output

2.87 Chromakey

This filter works in YUV colorspace. The color is defined alternatively as a RGB or YUV color and a similarity value.

Example for "chromakey" filter:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -f lavfi -i color=lightblue:size=384x128 -lavfi [0]chromakey=color=orange:similarity=0.1[a];[1][a]overlay -frames 1 -y out.png

pause
```

These are the input and output images:



Input



Output

2.88 HSVkey

In this filter the color is defined as a HSV color (Hue-Saturation-Value) and a similarity value.

Example for "hsvkey" filter:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

ffmpeg -i spectrum.png -f lavfi -i color=lightblue:size=384x128 -lavfi [0]hsvkey=hue=45:sat=0.7:val=0.5:similarity=0.2[a];[1][a]overlay -frames 1 -y out.png

pause
```

These are the input and output images:



Input



Output

2.89 Lumakey

In this filter the color is defined by a luminance value (threshold) and a tolerance value. The edges can be softened by a "softness" parameter.

Example for "lumakey" filter:

```
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)',scale=iw/4:ih/4 -frames 1 -y spectrum.png

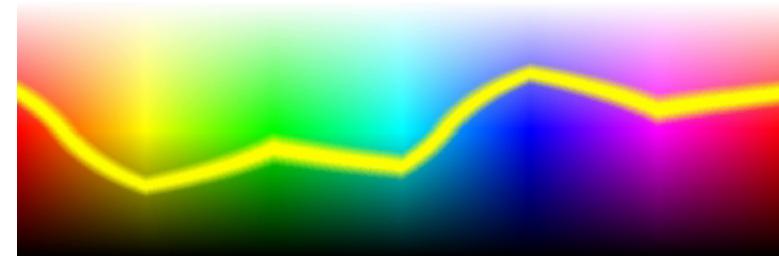
ffmpeg -i spectrum.png -f lavfi -i color=yellow:size=384x128 -lavfi
[0]lumakey=threshold=0.5:tolerance=0.02:softness=0.05[a];[1][a]overlay -frames 1 -y out.png

pause
```

These are the input and output images:



Input



Output

2.90 Bluescreen / greenscreen

```
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"                   :: Corp height
set "CX=500"                    :: Crop X
set "CY=0"                      :: Crop Y
set "COLOR=0x2223395"           :: Measured average blue screen color in format 0xRRGGBB
set "SIM=0.11"                  :: Similarity for blue color
set "SC=0.35"                   :: Scale factor for foreground video
set "X=800"                     :: X Position of foreground
set "Y=310"                     :: Y Position of foreground
set "T=28"                      :: Duration
set "OUT=out.mp4"               :: Output video

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

ffmpeg -ss 5 -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%,scale=iw*%SC%:ih*%SC%[FG];[BG][FG]overlay=%X%:%Y%" -t %T% -y %OUT%

pause
```

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.

Note: It's much easier to make the greenscreen and despill process in DaVinci Resolve.

Note: There is also a "colorkey" filter which is similar to the "chromakey" filter, but works in RGB (Red-Green-Blue) range.

Note: There is also a "hsvkey" filter which is similar to the "chromakey" filter, but works in HSV (Hue-Saturation-Value) range.

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. This workaround is no longer required because the "eq" filter was changed in October 2021 and does now support streams with alpha channel (the alpha channel remains unchanged):

```
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%  
pause
```

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.

Note about "alphamerge" filter: If the input video has no alpha channel, then an alpha channel is added, using the greyscale data from the second input. If the input video has an alpha channel, then the alpha channel is replaced by the greyscale data from the second input.

See also: despill filter, colorkey filter, hsvkey filter

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

```
This filter accepts the following options:  
'type'      Set what type of despill to use.  
'mix'       Set how spillmap will be generated.  
'expand'    Set how much to get rid of still remaining spill.  
'red'        Controls amount of red in spill area.  
'green'     Controls amount of green in spill area. Should be -1 for greenscreen.  
'blue'       Controls amount of blue in spill area. Should be -1 for bluescreen.  
'brightness' Controls brightness of spill area, preserving colors.  
'alpha'      Modify alpha from generated spillmap.
```

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

<code>type</code>	<code><int></code>	<code>..FV..... set the screen type (from 0 to 1) (default green)</code>
<code>green</code>	<code>0</code>	<code>..FV..... greenscreen</code>
<code>blue</code>	<code>1</code>	<code>..FV..... bluescreen</code>
<code>mix</code>	<code><float></code>	<code>..FV..... set the spillmap mix (from 0 to 1) (default 0.5)</code>
<code>expand</code>	<code><float></code>	<code>..FV..... set the spillmap expand (from 0 to 1) (default 0)</code>
<code>red</code>	<code><float></code>	<code>..FV..... set red scale (from -100 to 100) (default 0)</code>
<code>green</code>	<code><float></code>	<code>..FV..... set green scale (from -100 to 100) (default -1)</code>
<code>blue</code>	<code><float></code>	<code>..FV..... set blue scale (from -100 to 100) (default 0)</code>

```

brightness      <float>      . . . FV..... set brightness (from -10 to 10) (default 0)
alpha          <boolean>     . . . FV..... change alpha component (default false)
This filter has support for timeline through the 'enable' option.

```

This is the same bluescreen example as before, with additional despill filter:

```

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"                    :: Corp 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 bluescreen 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:

```
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%

pause
```

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:

```
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 -
```

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

```
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?

```
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 * (redscale   + 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):

<http://av-wiki.htwk-leipzig.de/index.php/Despill>

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: $\text{spillmap} = \text{blue} - (\text{red} * \text{spillmix} + \text{green} * \text{factor})$

"spillmap2" is for a modified formula: $\text{spillmap2} = \text{blue} - (\text{red} * \text{spillmix} + \text{blue} * \text{factor})$

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=0.00	spillmap2=0.00	

Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=0.00	expand=0.00	spillmap=0.00	spillmap2=0.00	
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=0.00	expand=0.00	spillmap=0.00	spillmap2=0.00	
Input: R=0.40 G=0.30 B=0.40 magenta-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=0.00	spillmap=0.00	spillmap2=0.00	
Input: R=0.30 G=0.30 B=0.30 gray	mix=0.00	expand=0.50	spillmap=0.15	spillmap2=0.15	
Input: R=0.30 G=0.30 B=0.40 blue-gray	mix=0.00	expand=0.50	spillmap=0.25	spillmap2=0.20	<--
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=0.00	expand=0.50	spillmap=0.10	spillmap2=0.15	<--
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=0.00	expand=0.50	spillmap=0.20	spillmap2=0.20	
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=0.00	expand=0.50	spillmap=0.15	spillmap2=0.15	
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=0.00	expand=0.50	spillmap=0.25	spillmap2=0.20	<--
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=0.00	expand=0.50	spillmap=0.10	spillmap2=0.15	<--
Input: R=0.30 G=0.30 B=0.30 gray	mix=0.00	expand=1.00	spillmap=0.30	spillmap2=0.30	
Input: R=0.30 G=0.30 B=0.40 blue-gray	mix=0.00	expand=1.00	spillmap=0.40	spillmap2=0.40	
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=0.00	expand=1.00	spillmap=0.30	spillmap2=0.30	
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=0.00	expand=1.00	spillmap=0.40	spillmap2=0.40	
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=0.00	expand=1.00	spillmap=0.30	spillmap2=0.30	
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=0.00	expand=1.00	spillmap=0.40	spillmap2=0.40	
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=0.00	expand=1.00	spillmap=0.30	spillmap2=0.30	
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=0.05	<--
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=0.50	expand=0.00	spillmap=0.00	spillmap2=0.00	
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=0.50	expand=0.00	spillmap=0.05	spillmap2=0.05	
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=0.50	expand=0.00	spillmap=0.00	spillmap2=0.00	
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=0.50	expand=0.00	spillmap=0.05	spillmap2=0.00	<--
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=0.50	expand=0.00	spillmap=0.00	spillmap2=0.00	
Input: R=0.30 G=0.30 B=0.30 gray	mix=0.50	expand=0.50	spillmap=0.08	spillmap2=0.08	
Input: R=0.30 G=0.30 B=0.40 blue-gray	mix=0.50	expand=0.50	spillmap=0.18	spillmap2=0.15	<--
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=0.50	expand=0.50	spillmap=0.05	spillmap2=0.08	<--
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=0.50	expand=0.50	spillmap=0.15	spillmap2=0.15	
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=0.50	expand=0.50	spillmap=0.03	spillmap2=0.03	
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=0.50	expand=0.50	spillmap=0.13	spillmap2=0.10	<--
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=0.50	expand=0.50	spillmap=0.00	spillmap2=0.03	<--
Input: R=0.30 G=0.30 B=0.30 gray	mix=0.50	expand=1.00	spillmap=0.15	spillmap2=0.15	
Input: R=0.30 G=0.30 B=0.40 blue-gray	mix=0.50	expand=1.00	spillmap=0.25	spillmap2=0.25	
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=0.50	expand=1.00	spillmap=0.15	spillmap2=0.15	
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=0.50	expand=1.00	spillmap=0.25	spillmap2=0.25	
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=0.50	expand=1.00	spillmap=0.10	spillmap2=0.10	
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=0.50	expand=1.00	spillmap=0.20	spillmap2=0.20	
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=0.50	expand=1.00	spillmap=0.10	spillmap2=0.10	

Input: R=0.30 G=0.30 B=0.30 gray	mix=1.00	expand=0.00	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.30 B=0.40 blue-gray	mix=1.00	expand=0.00	spillmap=0.10	spillmap2=0.10
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=1.00	expand=0.00	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=1.00	expand=0.00	spillmap=0.10	spillmap2=0.10
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=1.00	expand=0.00	spillmap=0.00	spillmap2=0.00
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=1.00	expand=0.00	spillmap=0.00	spillmap2=0.00
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=1.00	expand=0.00	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.30 B=0.30 gray	mix=1.00	expand=0.50	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.30 B=0.40 blue-gray	mix=1.00	expand=0.50	spillmap=0.10	spillmap2=0.10
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=1.00	expand=0.50	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=1.00	expand=0.50	spillmap=0.10	spillmap2=0.10
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=1.00	expand=0.50	spillmap=0.00	spillmap2=0.00
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=1.00	expand=0.50	spillmap=0.00	spillmap2=0.00
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=1.00	expand=0.50	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.30 B=0.30 gray	mix=1.00	expand=1.00	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.30 B=0.40 blue-gray	mix=1.00	expand=1.00	spillmap=0.10	spillmap2=0.10
Input: R=0.30 G=0.40 B=0.30 green-gray	mix=1.00	expand=1.00	spillmap=0.00	spillmap2=0.00
Input: R=0.30 G=0.40 B=0.40 cyan-gray	mix=1.00	expand=1.00	spillmap=0.10	spillmap2=0.10
Input: R=0.40 G=0.30 B=0.30 red-gray	mix=1.00	expand=1.00	spillmap=0.00	spillmap2=0.00
Input: R=0.40 G=0.30 B=0.40 magenta-gray	mix=1.00	expand=1.00	spillmap=0.00	spillmap2=0.00
Input: R=0.40 G=0.40 B=0.30 yellow-gray	mix=1.00	expand=1.00	spillmap=0.00	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:

```
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");
        }

        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.



mix07_bright0_blue-1.png



mix07_bright1_blue-1.png

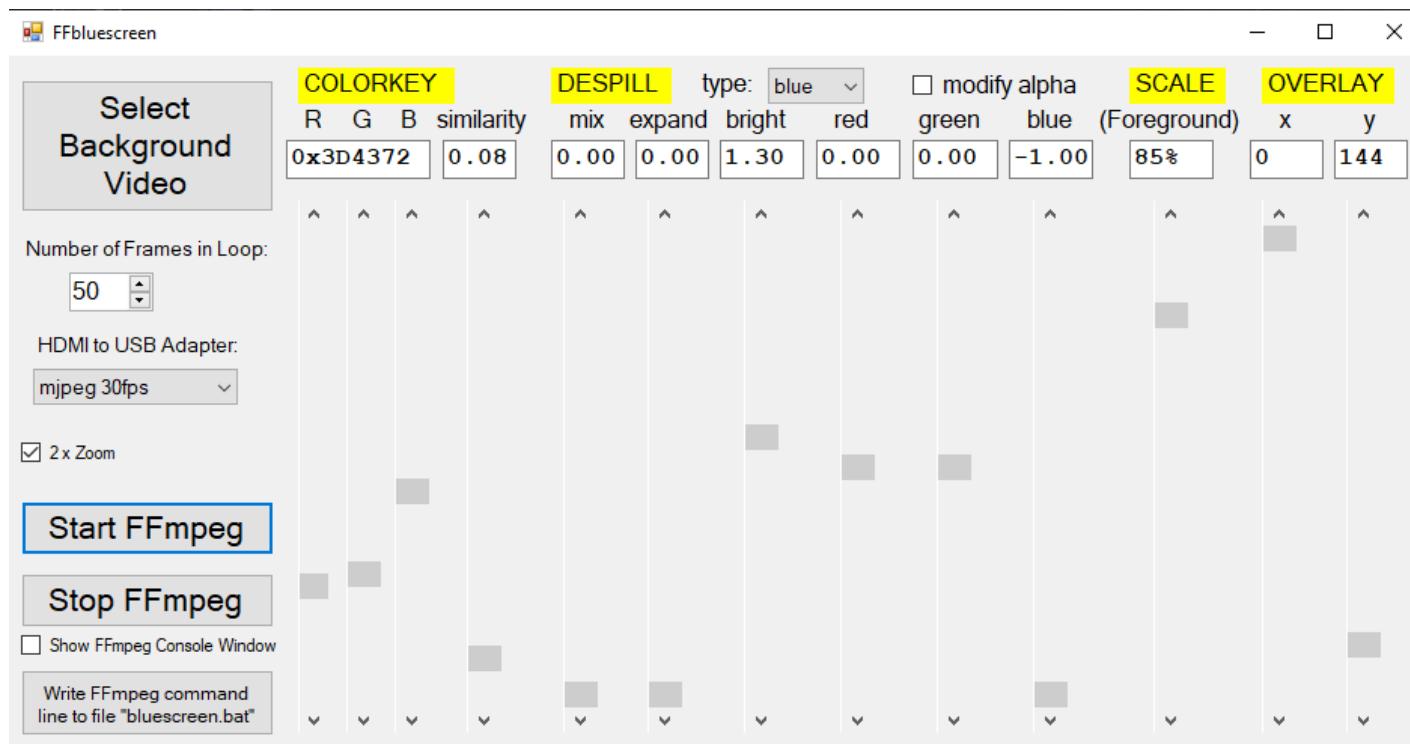


mix07_bright2_blue-1.png

2.91 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).



The source code can be downloaded here: <http://www.astro-electronic.de/source/FFbluescreen.zip>

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:

```
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".

2.92 Extract moving objects from a video

```
ffmpeg -ss 2 -i in.mov -lavfi tmix=10 -frames 1 -y background.jpg  
  
ffmpeg -i background.jpg -i in.mov -i wald.jpg -lavfi "[0]smartblur=3,format=rgb24[blurred_ref];[1]format=rgb24,split[a][b];[2]format=rgb24[c],[a]smartblur=3,format=rgb24[blurred_in];[blurred_ref][blurred_in]blend=all_mode=difference,eq=brightness=0.4:saturation=0,eq=contrast=1000,format=rgb24[mask];[c][b][mask]maskedmerge,format=rgb24" -shortest -pix_fmt yuv422p -y out.mkv  
  
pause
```

Note: It's not working very good.

See also: <http://oioiiooxiii.blogspot.com/2017/06/ffmpeg-predator-1987-movie-adaptive.html>

See also: <http://oioiiooxiii.blogspot.com/2016/09/ffmpeg-extractforeground-moving.html>

Another example with webcam input:

```
rem Create the background image without a person in front of it:  
  
ffmpeg -f dshow -video_size 640x480 -framerate 30 -pixel_format yuyv422 -i video="BisonCam,NB Pro" -vf tmix=25 -frames 1 -y bg.png  
  
rem Replace the original background by a green screen:  
  
ffmpeg -loop 1 -i bg.png -f lavfi -i color=green:size=640x480,format=rgb24 -f dshow -video_size 640x480 -framerate 30 -pixel_format yuyv422 -i video="BisonCam,NB Pro" -lavfi [2]format=rgb24,split[cam1][cam2];[cam1]smartblur=5,format=rgb24[blurredcam];[0]smartblur=5,format=rgb24[blurredbg];[blurredbg][blurredcam]blend=all_mode=difference,curves=m="0/0 .03/0 .04/1 1/1",format=gray,smartblur=5,format=rgb24[mask];[1][cam2][mask]maskedmerge,format=rgb24 -f sdl2 -  
  
pause
```

Note: It's also not working very good.

2.93 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.

See also: "pixscope" filter

2.94 Video line jitter effect

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

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  
pause
```

The problem is that this is a pixel-wise jitter. Each pixel gets its 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  
pause
```

The random() function uses the variable 0 to save its 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  
pause
```

2.95 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(ld(0)+2*%F%*(random(1)-0.5),0,%J%*ih))' -y %OUT%
pause
```

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.

This is another approach for creating a vertical jitter, using the "rgbashift" filter::

```
ffmpeg -f lavfi -i testsrc2=d=10 -lavfi sendcmd=f=cmd.txt,rgbashift=rv=0:gv=0:bv=0 -y jitter.mp4
pause
```

This is the content of the file cmd.txt:

```
0 [expr] rgbashift rv 'mod(3*N,10)';
0 [expr] rgbashift gv 'mod(3*N,10)';
0 [expr] rgbashift bv 'mod(3*N,10)';
```

2.96 CRT Screen Effect

Found here: <http://oioiiooixiii.blogspot.com/2019/04/ffmpeg-crt-screen-effect.html>

2.97 Macroblock motion vectors

Found here: <http://oioiiooixiii.blogspot.com/2018/04/ffmpeg-colour-animation-from-macroblock.html>

See also: <http://oioiiooixiii.blogspot.com/2016/09/ffmpeg-create-video-composite-of.html>

See also: <http://oioiiooixiii.blogspot.com/2016/04/ffmpeg-display-and-isolate-macroblock.html>

2.98 Deblock filter

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

2.99 Gradfun filter

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

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

ffmpeg -i %IN% -vf gradfun=3.5:8 -y %OUT%

pause
```

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.

2.100 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 limage. You can avoid this by using the dilation filter prior to scaling down.

See also the "morpho" filter and especially the "pixelize" filter, which has a "max" mode.

2.101 Morphological transforms

See also: https://en.wikipedia.org/wiki/Mathematical_morphology

This is an example for morphological transforms. The structure is a 9x9 image with black background and a white circular mask:

```
rem Make an input image for testing

ffmpeg -f lavfi -i color=black:s=400x160,format=rgb24 -lavfi drawtext=text="Test":font="arial
black":fontcolor=white:fontsize=140:x=20:y=25,rgbashift=gh=20:bh=10:bv=10 -frames 1 -y rgb_text.png

rem Make a 9x9 structure with a white circle

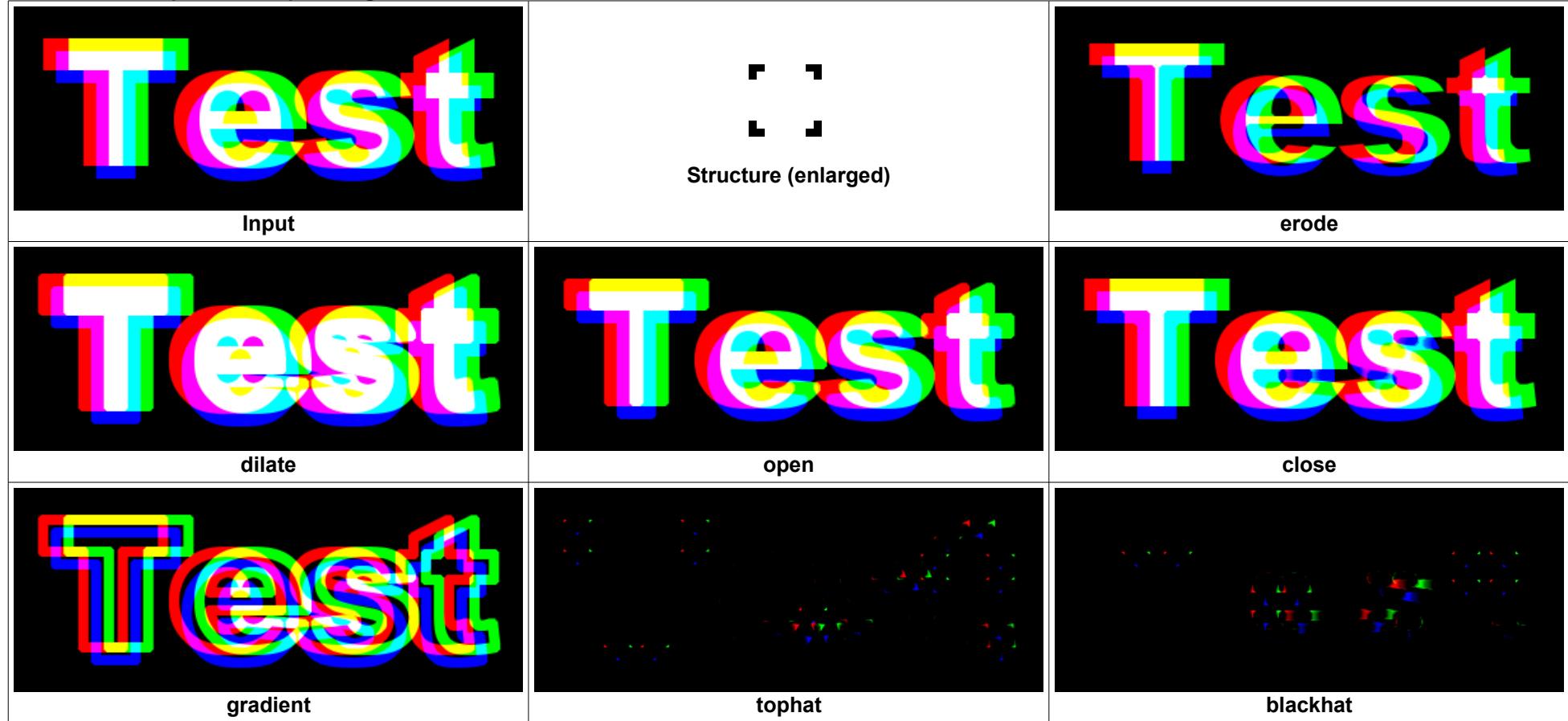
ffmpeg -f lavfi -i color=black:s=9x9,format=gray8 -lavfi geq=lum='255*lte(hypot(((2*X+1)/H-1),((2*Y+1)/H-
1)),1)',format=rgb24 -frames 1 -y structure.png

rem Test all modes of the "morpho" filter

ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=erode -y erode.png
ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=dilate -y dilate.png
ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=open -y open.png
ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=close -y close.png
ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=gradient -y gradient.png
ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=tophat -y tophat.png
ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=blackhat -y blackhat.png

pause
```

These are the input and output images:



If the structure is green instead of white, then the "morpho" filter is only applied to the green channel:

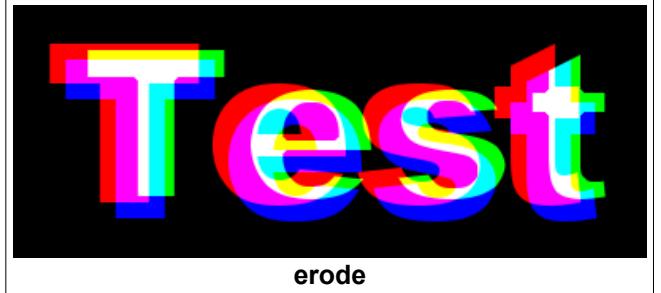
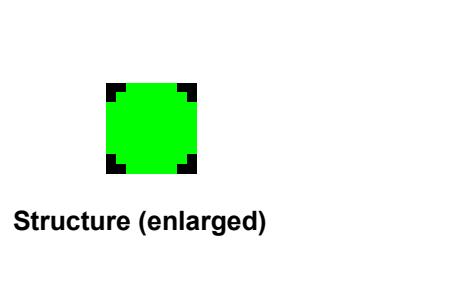
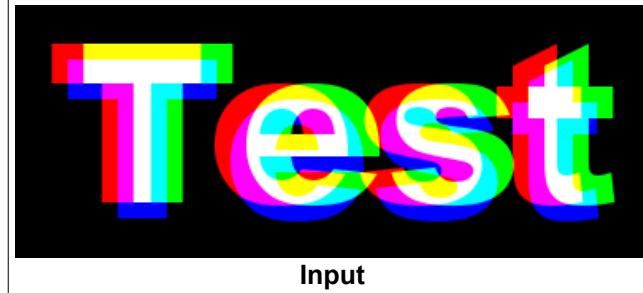
```
ffmpeg -f lavfi -i color=black:s=400x160,format=rgb24 -lavfi drawtext=text="Test":font="arial black":fontcolor=white:fontsize=140:x=20:y=25,rgbashift=gh=20:bh=10:bv=10 -frames 1 -y rgb_text.png

ffmpeg -f lavfi -i color=black:s=9x9,format=rgb24 -lavfi geq=g='255*lte(hypot(((2*x+1)/H-1),((2*y+1)/H-1)),1)' -frames 1 -y structure.png

ffmpeg -i rgb_text.png -i structure.png -lavfi morpho=erode -y erode.png

pause
```

These are the input and output images:



2.102 Correct the radial distortion of (fisheye-) lenses

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

cx	Relative x coordinate of the center of distortion, in range [0...1], default is 0.5
cy	Relative y coordinate of the center of distortion, in range [0...1], default is 0.5
k1	Coefficient of the quadratic correction term, in range [-1...1]. 0 means no correction, default is 0.
k2	Coefficient of the ^4 correction term, in range [-1...1]. 0 means no correction, default is 0.

The formula that generates the correction is:

$$r_{src} = r_{tgt} * (1 + k1 * (r_{tgt} / r_0)^2 + k2 * (r_{tgt} / r_0)^4)$$

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:

$$\text{ratio} = (w^2 + h^2) / h^2$$

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

$$k1 = c1 * \text{ratio}$$

$$k2 = c2 * \text{ratio}^2$$

See also <http://www.paulbourke.net/dome/fisheyecorrect/>

This is an example for the Entaniya M12 280° lens, from Paul Bourke's website:

$$y = 0.5229 x - 0.043 x^2 + 0.0253 x^3 - 0.0109 x^4$$

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 FOV * 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.

```
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 1:
rem Apply the fisheye distortion Example for Entaniya M12 280° lens
rem y = A * x + B * x^2 + C * x^3 + D * x^4
rem where x is in the [0...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 you want to transform this image into an undistorted fisheye image, with other words: You want to linearize it.

In this case you need the inverse of the above 4th 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.

```
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(-ld(1)+%A%*ld(0)+%B%*pow(ld(0),2)+%C%*pow(ld(0),3)+%D%*pow(ld(0),4),1));%S
%/2+0.5+ld(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
```

Note: It might be possible that the second argument of the root function (1) must be replaced by (%FOV%/360*PI), I'm not sure.

Here are the linearization formulas for a few fisheye lenses (Source: Paul Bourke's website):

Sigma 4.5mm f/2.5 180°	$y = 0.7159 x - 0.0048 x^2 - 0.032 x^3 + 0.0021 x^4$
Sigma 8mm f/3.5 180°	$y = 0.723 x + 0.0252 x^2 - 0.0499 x^3 - 0.0004325 x^4$
Canon 8-15mm f/4 180° @8mm	$y = 0.7189 x + 0.00079342 x^2 - 0.0289 x^3 - 0.001 x^4$
Canon 8-15mm f/4 180° @12mm	$y = 0.6645 x + 0.0071 x^2 - 0.0223 x^3 - 0.0024 x^4$
Meike 3.5mm 220°	$y = 0.6475 x - 0.002 x^2 - 0.0331 x^3 - 0.00010171 x^4$
Meike 6.5mm 190°	$y = 0.5165 x + 0.1106 x^2 + 0.0617 x^3 - 0.0601 x^4$
Laowa 4mm 210°	$y = 0.6241 x - 0.0252 x^2 + 0.024 x^3 - 0.0183 x^4$

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 FOV * PI / 360°.

2.103 Lensfun

Lensfun is a library for correcting geometric distortion, chromatic aberration and vignetting of lenses.

The lensfun filter needs one or more database files. These are *.xml files and can be found in the data/db folder at Github:

<https://github.com/lensfun/lensfun>

Copy one or more of these files to a local folder and specify the path in the "db_path" option. I did use the path c:/ffmpeg/lensfun

If this option isn't specified, lensfun will search at a default path (which I don't know).

The escaping in the path is tricky and hard to understand. The following versions are all working in a Windows batch file, however there are many more versions that don't work:

ffmpeg -i in.png -vf lensfun=db_path=C\\:/ffmpeg/lensfun out.png	Without " " double quotes
ffmpeg -i in.png -vf lensfun=db_path=\'C:/ffmpeg/lensfun\' out.png	Without " " double quotes, but path encapsulated in single '' quotes
ffmpeg -i in.png -vf lensfun=db_path=\'C://ffmpeg//lensfun\' out.png	
ffmpeg -i in.png -vf lensfun=db_path=\'C:\\\\ffmpeg\\\\lensfun\' out.png	
ffmpeg -i in.png -vf lensfun=db_path="C\\:/ffmpeg/lensfun" out.png	Path encapsulated in " " double quotes
ffmpeg -i in.png -vf lensfun=db_path="C\\://ffmpeg//lensfun" out.png	
ffmpeg -i in.png -vf "lensfun=db_path=C\\:/ffmpeg/lensfun" out.png	Whole filtergraph encapsulated in " " double quotes
ffmpeg -i in.png -vf "lensfun=db_path=\'C:/ffmpeg/lensfun\'" out.png	Whole filtergraph encapsulated in " " double quotes and path encapsulated in single '' quotes
ffmpeg -i in.png -vf "lensfun=db_path=\'C://ffmpeg//lensfun\'" out.png	
ffmpeg -i in.png -vf "lensfun=db_path=\'C:\\\\ffmpeg\\\\lensfun\'" out.png	

A path must be encapsulated in " " double quotes if it contains space characters. Colons ":" must be escaped as "\:", commas "," must be escaped as "\", and backslashes "\" must be escaped as "\\".

If one of both of the "make" or "model" options aren't specified, the lensfun filter will print a list of all available values in the console window:

```
rem List all available values for "make" and "model":  
  
ffmpeg -i 100mm.png -vf lensfun=db_path=C\\:/ffmpeg/lensfun -y out.png  
  
pause
```


If "make" and "model" are specified, but "lens_model" isn't specified, the lensfun filter will print a list of all available lenses in the console window:

```
rem List all available values for "lens_model":  
  
ffmpeg -i 100mm.png -vf lensfun=make=Canon:model="Canon EOS 5D Mark IV":db_path=C:\\ffmpeg\\lensfun -y out.png  
  
pause
```

Some lenses may be listed multiple times. That's because they are listed in the *.xml file for several different crop factors. Unfortunately FFmpeg doesn't list the crop factor. It's important that the crop factor of the lens' data fits to the crop factor of the camera. If in doubt, check the *.xml file with an editor.

Now if you have confirmed that data for the camera and lens is available, you can correct the distortion of an image. If it's a zoom lens, the actual focal length must be specified, because geometric distortion of the lens depends on the actual focal length. "scale=0" means automatic scaling, so that no-data areas at the borders are automatically cropped. "scale=1" means no scaling. "scale" is a floating point value and can also be set to other values.

```
rem Correct the geometrical distortion of a lens:  
  
ffmpeg -i 100mm.png -vf lensfun=make=Canon:model="Canon EOS 5D Mark IV":lens_model="Canon EF 100-400mm f/4.5-5.6L IS II  
USM":focal_length=100:scale=1:db_path=C\\ffmpeg\\lensfun -y out.png  
  
pause
```

Distortion correction can also be made with Darktable (I didn't test it myself): <https://www.darktable.org/>

There is a "Lensfun" plugin for Gimp, but it seems to be outdated and doesn't work with the latest version of Gimp (April 2022):
<https://seebk.github.io/GIMP-Lensfun/>

Lensfun is also available within RawTherapee: <https://www.rawtherapee.com/>

The Lensfun database used by RawTherapee is located in this path: C:\Program Files\RawTherapee\5.8\share\lensfun

How to update the Lensfun database (this didn't work correctly when I tested it):

https://rawpedia.rawtherapee.com/Lens/Geometry#Updating_your_Lensfun_database_in_Windows

More details about Lensfun in RawTherapee: <http://rawpedia.rawtherapee.com/Lens/Geometry>

If the Adobe DNG Converter is installed, then camera profiles (*.dcp) and lens profiles (*.lcp) are available in this path:

C:\ProgramData\Adobe\CameraRaw

This is a completely different approach for distortion correction and has nothing to do with Lensfun.

Lens distortion can also be corrected in DaVinci Resolve, use the "LensDistort" node in Fusion. The various correction models are explained here:

https://www.3dequalizer.com/user_daten/tech_docs/pdf/lstk.pdf

2.104 Replace the lensfun filter by the remap filter

If the lensfun filter doesn't work as expected, it's possible to replace it by the remap filter.

The Lensfun library provides many different mathematical models for lens distortion. The most important models are "ptlens" and "poly3".

The "ptlens" model uses a fourth-order polynomial:

$$R_d = a * R_u^4 + b * R_u^3 + c * R_u^2 + (1 - a - b - c) * R_u$$

Here R_u is the radius of the undistorted pixel and R_d is the radius of the distorted pixel. The radius is normalized to half of the smaller size of the image, or with other words to the radius of a circle that fully fits inside the image. That means all pixels on this unit circle remain stationary.

Source: http://sid.ethz.ch/debian/hugin/hugin-0.8.0/src/hugin1/hugin/xrc/data/help_en_EN/Lens_correction_model.html

Warning: It seems that Lensfun does internally use a simplified model, where the term $(1 - a - b - c)$ is approximated by 1. This results in a slightly different scale of the output image.

Another useful model is the "poly3" model, which is a simplified 3rd order polynomial:

$$R_d = k1 * R_u^3 + (1 - k1) * R_u$$

"poly3" is a simplified variant of the ptlens model, with $a=0$, $b=k1$ and $c=0$.

Source: <http://lensfun.sourceforge.net/calibration-tutorial/lens-distortion.html>

The coefficients for the distortion models can be found in the Lensfun database. These are *.xml files and can be found in the /data/db folder:

<https://github.com/lensfun/lensfun>

For example, in the *.xml file the section for my Canon zoom lens looks like this (I did remove the additional chromatic aberration and vignetting data). It's very important that the correct crop factor is listed. It must be the same crop factor as for the camera, in this case 1 for full frame:

```
<lens>
  <maker>Canon</maker>
  <model>Canon EF 100-400mm f/4.5-5.6L IS II USM</model>
  <mount>Canon EF</mount>
  <cropfactor>1</cropfactor>
  <calibration>
    <distortion model="ptlens" focal="100" a="-0.00104" b="-0.00264" c="-0.00045"/>
    <distortion model="ptlens" focal="135" a="-0.0049" b="0.0167" c="-0.02249"/>
    <distortion model="ptlens" focal="200" a="0.00118" b="-0.00395" c="0.00649"/>
    <distortion model="ptlens" focal="300" a="-0.0007" b="0.00406" c="-0.00001"/>
    <distortion model="ptlens" focal="400" a="-0.00355" b="0.01547" c="-0.00989"/>
  </calibration>
</lens>
```

You see that the "ptlens" coefficients a, b and c do strongly depend of the focal length. In the following example I'm correcting an image that was taken at 100mm focal length:

```
set "IN=100mm.jpg"          :: Input image
set "SX=6720"                :: Size X
set "SY=4480"                :: Size Y
set "A=-0.00104"             :: Distortion coefficient a for ptlens model
set "B=-0.00264"             :: Distortion coefficient b for ptlens model
set "C=-0.00045"             :: Distortion coefficient c for ptlens model
set "R=0.003"                 :: Rotation angle in radians, positive is counter-clockwise
set "OUT=out.jpg"             :: Output image

rem Rd = a * Ru^4 + b * Ru^3 + c * Ru^2 + (1 - a - b - c) * Ru
rem Ru is the radius of the undistorted pixel, Rd is the radius of the distorted pixel
rem The radius is normalized to half of the smaller size of the image, with other words
rem to the radius of a circle that fully fits inside the image.

rem Create the xmap file

ffmpeg -f lavfi -i nullsrc=size=%SX%x%SY% -vf format=pix_fmts=gray16le,geq='^
st(0,hypot(Y-(%SY%-1)/2,X-(%SX%-1)/2)/(0.5*%SY%));^
st(1,%R%+atan2(Y-(%SY%-1)/2,X-(%SX%-1)/2));^
st(2,%A%*pow(ld(0),4)+%B%*pow(ld(0),3)+%C%*pow(ld(0),2)+(1-%A%-%B%-%C%)*ld(0));^
```

```
%SX%/2+ld(2)*0.5*%SY%*cos(ld(1))' -frames 1 -y xmap.pgm

rem Create the ymap file

ffmpeg -f lavfi -i nullsrc=size=%SX%x%SY% -vf format=pix_fmts=gray16le,geq='^
st(0,hypot(Y-(%SY%-1)/2,X-(%SX%-1)/2)/(0.5*%SY%));^
st(1,%R%+atan2(Y-(%SY%-1)/2,X-(%SX%-1)/2));^
st(2,%A%*pow(ld(0),4)+%B%*pow(ld(0),3)+%C%*pow(ld(0),2)+(1-%A%-%B%-%C%)*ld(0));^
%SY%/2+ld(2)*0.5*%SY%*sin(ld(1))' -frames 1 -y ymap.pgm

rem Apply the remap filter to the image

ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=pix_fmts=rgb24,remap" -frames 1 -y %OUT%
pause
```

This is not the fastest solution, but it works fine. The only drawback is that there is no pixel interpolation.

The scale of the resulting image is a little bit different from the result of the "lensfun" filter (with option scale=1).

After I did set scale to approximately 0.996, both images were identical. It seems that in Lensfun the term $(1 - a - b - c)$ is approximated by 1.

Hint: In my example $(1 + a + b + c) = 0.99587$

If you want that all points on the unit circle remain stationary, you should use the term $(1 - a - b - c)$.

However if you want exactly the same result as with Lensfun (with scale=1), then replace the term $(1 - a - b - c)$ by 1, or alternatively use the term $(1 - a - b - c)$ and set scale to $(1 + a + b + c)$.

2.105 Deep-Zoom-In

still working on this chapter...

Let's assume you have two images of the same scene, one taken with 100mm focal length and the other with 200mm.

The task is to enlarge a region of the first image so that it becomes identical to the second image. This is only possible if lens distortion is corrected perfectly in both images. Unfortunately the correction with Lensfun isn't good enough, as the remaining errors are still visible.

```
set "SX=6720"      :: Image size X
set "SY=4480"      :: Image size Y
set "X1A=3210"     :: Point A in image 1
set "Y1A=1480"     ::
set "X1B=1747"     :: Point B in image 1
set "Y1B=1338"     ::
set "X2A=3016"     :: Point A in image 2
set "Y2A=1213"     ::
set "X2B=126"       :: Point B in image 2
set "Y2B=911"       ::

set "ZOOM=(hypot(%Y2A%-%Y2B%,%X2A%-%X2B%)/hypot(%Y1A%-%Y1B%,%X1A%-%X1B%))"
set "ROT=(atan2(%Y2A%-%Y2B%,%X2A%-%X2B%)-atan2(%Y1A%-%Y1B%,%X1A%-%X1B%))"

rem Print the zoom factor and rotation angle:
ffmpeg -f lavfi -i nullsrc=size=1x1 -vf format=gray16le,geq='print(%ZOOM%);print(%ROT%)' -frames 1 -y -f null NUL

rem Apply the zoom factor and rotation angle:
ffmpeg -i image1.jpg -vf zoompan=zoom='%ZOOM%':x=1682:y=880:s=%SX%x%SY% -frames 1 -y out.jpg

pause
```

2.106 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 "order" 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

pause
```



t_original.jpg



t_pitch90.jpg



t_pitch90_roll90.jpg



t_roll90.jpg



t_yaw90.jpg



t_yaw90_pitch90.jpg



t_yaw90_roll90.jpg

Parameters of the v360 filter:

input, output	e, equirect	Equirectangular projection, see https://en.wikipedia.org/wiki/Equirectangular_projection Warning: There is a small rounding error in v360 if the input is "e", but in most cases this error can be neglected. See https://trac.ffmpeg.org/ticket/9617
	c3x2	3x2 cubemap projection, also used by Youtube, see http://paulbourke.net/panorama/youtubeformat/
	c6x1	6x1 cubemap projection
	c1x6	1x6 cubemap projection
	eac	Equi-angular cubemap
	flat, gnomonic, rectilinear	Regular video projection, see https://en.wikipedia.org/wiki/Gnomonic_projection Warning: There is a small rounding error in v360 if the input is "flat", but in most cases this error can be neglected. See https://trac.ffmpeg.org/ticket/9617
	dfisheye	Double fisheye projection
	barrel, fb, barrelsplit	Facebook's "transform360" projection, see also https://github.com/facebook/transform360
	sg	Stereographic fisheye projection, see https://en.wikipedia.org/wiki/Stereographic_map_projection
	mercator	Mercator projection, see https://en.wikipedia.org/wiki/Mercator_projection
	ball	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.
	hammer	Hammer-Aitoff map projection, see https://en.wikipedia.org/wiki/Hammer_projection
	sinusoidal	Sinusoidal map projection projection, see https://en.wikipedia.org/wiki/Sinusoidal_projection
	fisheye	Single fisheye projection (equidistant)
	pannini	Pannini projection (output only), see https://en.wikipedia.org/wiki/Image_stitching#Pannini
	cylindrical	Cylindrical projection Warning: There is a small rounding error in v360 if the input is "cylindrical", but in most cases this error can be neglected. See https://trac.ffmpeg.org/ticket/9617
	cylindricalea	Cylindrical equal area projection, see https://en.wikipedia.org/wiki/Cylindrical_equal-area_projection
	perspective	Perspective projection, this is like watching a sphere from large distance (output only). Warning: Don't use the v360 filter's "perspective" projection if you need mathematically correct behaviour. The "perspective" projection depends somehow on the option "v_fov", but the exact meaning of this option in this context is unknown. Better use the workaround with "remap" filter. I did try to reverse-engineer the

		source code to find out the meaning of the "v_fov" option, but this approach wasn't successful. Probably the implementation of the "v_fov" option is wrong.
	tetrahedron	Tetrahedron projection
	octahedron	Octahedron projection
	tsp	Truncated square pyramid projection
	he, hequirect	Half equirectangular projection
	equisolid	Equisolid fisheye projection, see https://en.wikipedia.org/wiki/Fisheye_lens#Mapping_function
	og	Orthographic fisheye projection, see https://en.wikipedia.org/wiki/Orthographic_map_projection
interp	near, nearest	Nearest neighbour interpolation
	line, linear	Bilinear interpolation, this is the default
	cube, cubic	Bicubic interpolation
	lanc, lanczos	Lanczos interpolation
	sp16, spline16	Spline16 interpolation
	gauss, gaussian	Gaussian interpolation
	mitchell	Mitchell interpolation
w, h	in pixels	Width and height of the output image or video, default size depends on output format
yaw, pitch, roll	in degrees	<p>Rotation angles</p> <p>Warning: In the command line these are absolute angles, however if they are sent via sendcmd, then they are interpreted as relative angles. That means you can't test the angles in the command line and then use the same values for sendcmd. What a stupid idea!</p>
rorder	'ypr', 'yp', 'pyr', 'pr', '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
alpha_mask	0, 1	Make all unmapped pixels transparent

<code>h_offset, v_offset</code>	-1 ... 1	Output horizontal and vertical off-axis offset, the default values are 0. Note: In fact, this isn't output offset correction but input offset correction. Output offset correction isn't yet implemented in the V360 filter. For more details see chapter "How to replace the v360 filter by the remap filter".
---------------------------------	----------	--

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: If the transformation is from "equirectangular" to "ball", you must use different coordinates for the filled pixel:

```
-vf drawbox=x=3*iw/4:y=ih/2:w=1:h=1:color=green,v360=input=e:output=ball...
```

Note: Even if the input pixel format is `rgb24`, the output format is `gbpr` which is a planar pixel format.

See also:

List of map projections: https://en.wikipedia.org/wiki/List_of_map_projections#Cylindrical

Fisheye mapping functions: https://en.wikipedia.org/wiki/Fisheye_lens#Mapping_function

Paul Bourke's classification of fisheye mappings: <http://paulbourke.net/dome/fisheyetypes/>

$r = 2 * f * \sin(\theta / 2)$	equisolid	These are all fisheye projections
$r = f * \theta$	equidistant	
$r = 2 * f * \tan(\theta / 2)$	stereographic	
$r = f * \sin(\theta)$	orthographic	
$r = f * \tan(\theta)$	rectilinear	

Theta is the angle between the incoming ray of light and the optical axis (in radians), f is the focal length, and r is the distance of the resulting dot on the sensor from the sensor center.

2.107 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>

<https://space.jpl.nasa.gov/tmaps/stars.html>

https://www.solarsystemscope.com/textures/download/8k_stars_milky_way.jpg

2.108 Equirectangular images of the earth and planet surfaces

Solar system, high resolution images: <https://www.solarsystemscope.com/textures/>

Mercury: <https://space.jpl.nasa.gov/tmaps/mercury.html>

Earth without clouds: https://de.wikipedia.org/wiki/Blue_Marble#/media/File:Equirectangular-projection.jpg

Earth with clouds, video: <https://www.shutterstock.com/video/clip-22510927-equirectangular-map-precipitation-on-earth-planet-seamless>

Moon: https://www.physics.unlv.edu/~jeffery/astro/moon/map/moon_map_mercator.jpg

https://www.solarsystemscope.com/textures/download/8k_moon.jpg

Mars: https://astrogeology.usgs.gov/search/map/Mars/Viking/MDIM21/Mars_Viking_MDIM21_ClrMosaic_global_232m

2.109 Undo spherical rotations

If you want to undo spherical rotations, you must apply the same rotation angles with different sign, and you must apply the rotations in the reverse order. For example, this spherical rotation

```
v360=e:e:yaw=10:pitch=20:roll=30
```

can be undone by this spherical rotation:

```
v360=e:e:yaw=-10:pitch=-20:roll=-30:rorder=rpy
```

Note: It doesn't care in which order you write the angles in the command line. The rotation order is only determined by the "rorder" option. The default rotation order is yaw-pitch-roll.

Note: The order of the rotations is important. We are talking about a spherical coordinate system, like the geographical coordinates on earth.

Case 1: You walk 1km north, then turn 90° left and walk 1km to the west.

Case 2: You walk 1km to the west, then turn 90° right and walk 1km north.

You might think that in both cases you arrive at the same place. But that's wrong. It becomes clear when you walk longer distances:

Case 3: You start at the equator in Africa and walk north until you reach the north pole, then turn 90° left and walk the same distance. You will arrive somewhere in central America.

Case 4: You start at the equator in Africa and walk west until you are in central America, then turn 90° right and walk the same distance. You will arrive at the north pole.

That means the order of rotations is important. When you have made yaw and pitch rotations (in this order) and want to undo these rotations, you must undo them in reverse order. First pitch and then yaw.

2.110 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/

More informations about the remap filter can be found here: <https://trac.ffmpeg.org/wiki/RemapFilter>

The color of unmapped pixels can be specified with the "fill" option, the default color is black.

The remap filter uses framesync, which by default repeats the last frame of the secondary input if the primary input is still alive. That's why "-loop 1" is not required before the mapping file inputs.

What's better, -loop 1 before the input files, or "loop" filter in the filtergraph? For single image input, filter looping is better as it avoids file I/O and repeat decoding of the input. (Source: Gyan Doshi in FFmpeg user list, June 16, 2022)

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.

```
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:ih_fov=%FOV%:iv_fov=%FOV%:output=equirect:pitch=-90 -y
%OUT%

pause
```

Note: For image output, add "-frames 1"

Note: If required, the lower part of the equirectangular output can be cut off with the crop filter.

2.111 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%
pause
```

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

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

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

Unfortunately the v360 filter with fisheye output fills not only the image circle with data, but instead the whole quadratic image. The workaround is to overlay a circular mask:

```
set "IN=equirectangular.png"    :: Input image
set "SIZE=1200x1200"           :: Size of the mask image
set "FOV=180"                  :: Output field of view in degrees
set "OUT=fish.png"             :: Output image

ffmpeg -f lavfi -i color=black:s=%SIZE% -lavfi format=argb,geq=a='255*gt(hypot(((2*x+1)/H-1),((2*y+1)/H-1)),1)':r=0:g=0:b=0 -frames 1 -y mask.png

ffmpeg -i %IN% -i mask.png -lavfi v360=e:fisheye:h_fov=%FOV%:v_fov=%FOV%:pitch=90,overlay -y %OUT%

pause
```

2.112 Realtime remapping from equirectangular to fisheye

This example is for a computer with two or more monitors. That means the desktop is wider than the monitor. It's assumed that the main monitor contains an equirectangular content (for example SpaceEngine). This content is grabbed by the "gdigrab" input device, converted to 180° fisheye format and then shown in the center of the other monitor (or beamer).

```
ffmpeg -f gdigrab -video_size 1920x1080 -offset_x 0 -offset_y 0 -i desktop -lavfi  
v360=e:fisheye:ih_fov=360:iv_fov=180:h_fov=180:v_fov=180:w=1080:h=1080,format=bgra -window_borderless 1 -window_x 2340  
-window_y 0 -f sdl2 -  
  
pause
```

Note: "-offset_x 0 -offset_y 0" can be omitted because the top left corner is the default, but "-video_size 1920x1080" is important because in a multi-monitor system the desktop is larger than the monitor, however we want to grab only the first screen.

Note: You can see the framerate at the bottom of the console window.

Note: The "format=bgra" conversion is probably required because that's the screen pixel format which is set in the operating system. "rgba" does also work. "rgb0" or "bgr0" can also be used. These are pixel formats without alpha channel, but they are a little bit slower than "bgra".

The above example is running on my computer with 20fps and can be improved to 27fps by grabbing only the square central part of the equirectangular input video:

```
ffmpeg -f gdigrab -video_size 1080x1080 -offset_x 420 -i desktop -lavfi  
v360=e:fisheye:ih_fov=180:iv_fov=180:h_fov=180:v_fov=180:w=1080:h=1080,format=bgra -window_borderless 1 -window_x 2340  
-window_y 0 -f sdl2 -  
  
pause
```

Note: The v360 filter becomes faster if interpolation is deactivated by using "interp=near".

The remap filter is faster than the v360 filter. This example runs with 30fps (the mapping files are simple identity maps, but that shouldn't affect the speed):

```
rem Create the xmap file (this is a simple identity map only for testing)
ffmpeg -f lavfi -i nullsrc=size=1080x1080 -vf format=gray16,geq='X' -frames 1 -y xmap.pgm

rem Create the ymap file (this is a simple identity map only for testing)
ffmpeg -f lavfi -i nullsrc=size=1080x1080 -vf format=gray16,geq='Y' -frames 1 -y ymap.pgm

ffmpeg -f gdigrab -video_size 1080x1080 -offset_x 420 -i desktop -i xmap.pgm -i ymap.pgm -lavfi fps=30,remap
>window_borderless 1 -window_x 2340 -window_y 0 -f sdl2 -

pause
```

Note: Before I inserted the "fps" filter, there were numerous errors "Application provided invalid, non-monotonically increasing dts to muxer". It wasn't possible to remove them with the bitstream filter `-bsf:v setts=dts=DTS-STARTDTS`. But with `fps=30` it works fine. But now you can't see any more the maximum possible framerate in the console window.

Note: Bitstream filters transform encoded media data without decoding it.

This is an example with fisheye to equirectangular conversion (wrong direction...)

```
rem Create the xmap file

ffmpeg -f lavfi -i nullsrc=size=1080x1080 -vf format=pix_fmts=gray16le,geq='st(0,PI/2*(X/539.5-1));st(1,PI/2*(Y/539.5-1));st(2,cos(ld(1))*cos(ld(0)));st(3,cos(ld(1))*sin(ld(0)));st(4,sin(ld(1)));st(5,ld(2));st(6,ld(3));st(7,ld(4));st(8,hypot(ld(6),ld(7)));st(9,atan(ld(8)/ld(5))/(PI/2));539.5+539.5*ld(6)/ld(8)*ld(9)' -frames 1 -y xmap.pgm

rem Create the ymap file

ffmpeg -f lavfi -i nullsrc=size=1080x1080 -vf format=pix_fmts=gray16le,geq='st(0,PI/2*(X/539.5-1));st(1,PI/2*(Y/539.5-1));st(2,cos(ld(1))*cos(ld(0)));st(3,cos(ld(1))*sin(ld(0)));st(4,sin(ld(1)));st(5,ld(2));st(6,ld(3));st(7,ld(4));st(8,hypot(ld(6),ld(7)));st(9,atan(ld(8)/ld(5))/(PI/2));539.5+539.5*ld(7)/ld(8)*ld(9)' -frames 1 -y ymap.pgm

rem Realtime remapping

ffmpeg -f gdigrab -video_size 1080x1080 -offset_x 420 -i desktop -i xmap.pgm -i ymap.pgm -lavfi fps=30,remap -window_borderless 1 -window_x 2340 -window_y 0 -f sdl2 -

pause
```

This is an example for half-equirectangular (960x1080) to 180° fisheye (1080x1080) conversion. The input is the 1920x1080 screen from SpaceEngine, because the aspect ratio isn't 2:1, only the central 960x1080 rectangle is grabbed. The output can be sent to a beamer with 180° fisheye lens. The framerate is about 30fps on my notebook.

```

rem From equirectangular to fisheye

set "IN_W=960"      :: Input width in pixels
set "IN_H=1080"      :: Input height in pixels
set "IN_H_FOV=180"   :: Input horizontal field of view in degrees
set "IN_V_FOV=180"   :: Input vertical field of view in degrees
set "OUT_W=1080"     :: Output width in pixels
set "OUT_H=1080"     :: Output height in pixels
set "OUT_H_FOV=180"  :: Output horizontal field of view in degrees
set "OUT_V_FOV=180"  :: Output vertical field of view in degrees


rem Create the xmap file

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='st(0,%OUT_H_FOV%/180*(2*X/(%OUT_W%-1)-1));st(1,%OUT_V_FOV%/180*(2*Y/(%OUT_H%-1)-1));st(2,atan2(ld(1),ld(0)));st(3,PI/2*(1-hypot(ld(0),ld(1))));st(4,cos(ld(3))*cos(ld(2)));(%IN_W%-1)/2*(1+atan2(ld(4),sin(ld(3)))/PI*360/%IN_H_FOV%)' -frames 1 -y xmap.pgm


rem Create the ymap file

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='st(0,%OUT_H_FOV%/180*(2*X/(%OUT_W%-1)-1));st(1,%OUT_V_FOV%/180*(2*Y/(%OUT_H%-1)-1));st(2,atan2(ld(1),ld(0)));st(3,PI/2*(1-hypot(ld(0),ld(1))));st(4,asin(cos(ld(3))*sin(ld(2))));(%IN_H%-1)/2*(1+ld(4)/PI*360/%IN_V_FOV%)' -frames 1 -y ymap.pgm


rem Apply the remap filter in realtime

ffmpeg -f gdigrab -video_size %IN_W%x%IN_H% -offset_x 480 -i desktop -i xmap.pgm -i ymap.pgm -lavfi fps=30,remap -window_borderless 1 -window_x 2400 -window_y 0 -f sdl2 -


pause

```

2.113 How to replace the v360 filter by the remap filter

In some cases it's better to make spherical transformations with the "remap" filter instead of "v360" filter, for example if input or output formats or special effects are required that aren't available in "v360", or as a workaround for some errors in "v360". Also the "remap" filter is faster than "v360", however the speed improvement is quite small if the v360 filter is used with the "interp=near" option. **The following formulae have been carefully tested for correct rounding and absence of +1 errors.**

Comments	For which input / output format is it?	Command line in batch file
Specify the parameters for the spherical transformation. Of course, you only have to specify those parameters that are actually used.	For all formats	<pre> set "IN_W=1080" :: Input width in pixels set "IN_H=1080" :: Input height in pixels set "IN_H_FOV=180" :: Input horizontal field of view in degrees set "IN_V_FOV=180" :: Input vertical field of view in degrees set "OUT_W=1080" :: Output width in pixels set "OUT_H=1080" :: Output height in pixels set "OUT_H_FOV=180" :: Output horizontal field of view in degrees set "OUT_V_FOV=180" :: Output vertical field of view in degrees set "YAW=0" :: Yaw angle in degrees set "PITCH=0" :: Pitch angle in degrees set "ROLL=0" :: Roll angle in degrees set "OFF_IN_X=0" :: Input X offset, normalized to the dome's radius set "OFF_IN_Y=0" :: Input Y offset, normalized to the dome's radius set "OFF_IN_Z=0" :: Input Z offset, negative is closer to dome's vertex set "OFF_OUT_X=0" :: Output X offset, normalized to the dome's radius set "OFF_OUT_Y=0" :: Output Y offset, normalized to the dome's radius set "OFF_OUT_Z=0" :: Output Z offset, negative is closer to dome's vertex set "LENS_RAD=60" :: Radius of forward enlarging effect in degrees (1-90°) set "LENS_PWR=0" :: Power of forward enlarging effect, 0 = neutral set "DIST=2" :: Distance from camera to center of sphere, normalized :: to the sphere's radius, must be larger than 1 set "ELLIPSE=0.5" :: Ellipsoid distortion factor </pre>
The size of the xmap and ymap files is the size of the output image. The size of the input image may be different.	For all formats	<pre> rem Create the xmap and ymap files ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^ </pre>

<p>Convert from pixel coordinates to X,Y,Z space, depending on the desired output format.</p>	<p>For equirectangular output (see equirect_to_xyz in v360.c)</p>	<pre>st(0,PI/360*%OUT_H_FOV%*((2*X+1)/%OUT_W%-1));^ st(1,PI/360*%OUT_V_FOV%*((2*Y+1)/%OUT_H%-1));^ st(4,cos(ld(1))*sin(ld(0)));^ st(5,sin(ld(1)));^ st(6,cos(ld(1))*cos(ld(0)));^</pre>
<p>The X,Y,Z output coordinates are saved in variables 4,5,6.</p> <p>Note: If the pixel is unmapped, variable (9) is set to OUT_H which means out of range. The line st(9,...) can be omitted if the color of unmapped pixels doesn't care.</p>	<p>For fisheye (equidistant) output (see fisheye_to_xyz in v360.c)</p>	<pre>st(0,%OUT_H_FOV%/180*((2*X+1)/%OUT_W%-1));^ st(1,%OUT_V_FOV%/180*((2*Y+1)/%OUT_H%-1));^ st(2,atan2(ld(1),ld(0)));^ st(3,PI/2*hypot(ld(0),ld(1)));^ st(4,sin(ld(3))*cos(ld(2)));^ st(5,sin(ld(3))*sin(ld(2)));^ st(6,cos(ld(3)));^ st(9,if(lt(ld(6),cos(%OUT_H_FOV%/360*PI)),%OUT_H%,0));^</pre>
<p>For nonlinear fisheye output according to Paul Bourke's formula, see http://www.paulbourke.net/dome/fisheyecorrect/</p> <p>Set the parameters A, B, C, D and the correct field of view.</p>		<pre>st(0,%OUT_H_FOV%/180*((2*X+1)/%OUT_W%-1));^ st(1,%OUT_V_FOV%/180*((2*Y+1)/%OUT_H%-1));^ st(2,atan2(ld(1),ld(0)));^ st(3,PI/2*hypot(ld(0),ld(1)));^ st(9,if(gt(ld(3),PI/2*%OUT_H_FOV%/180),%OUT_H%,0));^ st(3,%A%*ld(3)+%B%*pow(ld(3),2)+%C%*pow(ld(3),3)+%D%*pow(ld(3),4));^ st(3,ld(3)*%OUT_H_FOV%/360*PI);^ st(4,sin(ld(3))*cos(ld(2)));^ st(5,sin(ld(3))*sin(ld(2)));^ st(6,cos(ld(3)));^</pre>
	<p>For double fisheye (equidistant) output (see dfisheye_to_xyz in v360.c)</p>	<pre>st(0,%OUT_H_FOV%/180*((4*if(lt(X,%OUT_W%/2),X,%OUT_W%-1-X)+1)/%OUT_W%-1));^ st(1,%OUT_V_FOV%/180*((2*Y+1)/%OUT_H%-1));^ st(2,atan2(ld(1),ld(0)));^ st(3,PI/2*hypot(ld(0),ld(1)));^ st(9,if(gt(ld(3),PI/2),%OUT_H%,0));^ if(gte(X,%OUT_W%/2),st(3,PI-ld(3)));^ st(4,sin(ld(3))*cos(ld(2)));^ st(5,sin(ld(3))*sin(ld(2)));^ st(6,cos(ld(3)));^</pre>
	<p>For flat (rectilinear) output (see flat_to_xyz in v360.c)</p>	<pre>st(0,tan(%OUT_H_FOV%/360*PI)*((2*X+1)/%OUT_W%-1));^ st(1,tan(%OUT_V_FOV%/360*PI)*((2*Y+1)/%OUT_H%-1));^ st(6,1/sqrt(1+ld(0)*ld(0)+ld(1)*ld(1)));^ st(4,ld(0)*ld(6));^ st(5,ld(1)*ld(6));^</pre>

	For cylindrical output (see cylindrical_to_xyz in v360.c)	<pre>st(0,PI/360*%OUT_H_FOV%*((2*X+1)/%OUT_W%-1));^ st(1,atan(tan(%OUT_V_FOV%/360*PI)*((2*Y+1)/%OUT_H%-1)));^ st(4,cos(ld(1))*sin(ld(0)));^ st(5,sin(ld(1)));^ st(6,cos(ld(1))*cos(ld(0)));^</pre>
	For planet output, this is a sphere seen from infinite distance. Exactly half of the sphere is visible. It's the same as orthographic output with 180° field of view.	<pre>st(0,(2*X+1)/%OUT_W%-1);^ st(1,(2*Y+1)/%OUT_H%-1);^ st(4,ld(0));^ st(5,ld(1));^ st(6,sqrt(1-ld(4)*ld(4)-ld(5)*ld(5)));^ st(9,if(gt(ld(4)*ld(4)+ld(5)*ld(5),1),%OUT_H%,0));^</pre>
	For planet output, this is a sphere seen from the finite distance %DIST%. Less than half of the sphere is visible. (The algorithm perspective_to_xyz in v360.c is probably wrong)	<pre>st(0,((2*X+1)/%OUT_W%-1));^ st(1,((2*Y+1)/%OUT_H%-1));^ st(2,hypot(ld(0),ld(1)));^ st(9,if(gt(ld(2),1),%OUT_H%,0));^ st(2,ld(2)*tan(asin(1/%DIST%)));^ st(3,1+ld(2)*ld(2)*(1-%DIST%*%DIST%));^ st(2,2*atan((1-sqrt(abs(ld(3))))/((1+%DIST%)*ld(2))));^ st(3,atan2(ld(0),ld(1)));^ st(4,sin(ld(2))*sin(ld(3)));^ st(5,sin(ld(2))*cos(ld(3)));^ st(6,cos(ld(2)));^</pre>
	For mirrorsphere output, this is a reflecting sphere seen from infinite distance. (see ball_to_xyz in v360.c)	<pre>st(0,((2*X+1)/%OUT_W%-1));^ st(1,((2*Y+1)/%OUT_H%-1));^ st(9,if(gt(hypot(ld(0),ld(1)),1),%OUT_H%,0));^ st(2,2*asin(clip(hypot(ld(0),ld(1)),0,1)));^ st(3,atan2(ld(0),ld(1)));^ st(4,sin(ld(2))*sin(ld(3)));^ st(5,sin(ld(2))*cos(ld(3)));^ st(6,cos(ld(2)));^</pre>

	For mirrorsphere output, this is a reflecting sphere seen from the finite distance %DIST%. This isn't implemented in the v360 filter.	<pre> st(0,((2*X+1)/%OUT_W%-1));^ st(1,((2*Y+1)/%OUT_H%-1));^ st(2,hypot(ld(0),ld(1)));^ st(9,if(gt(ld(2),1),%OUT_H%,0));^ st(2,ld(2)*tan(asin(1/%DIST%)));^ st(3,1+ld(2)*ld(2)*(1-%DIST%*%DIST%));^ st(2,4*atan((1-sqrt(abs(ld(3))))/((1+%DIST%)*ld(2))));^ st(3,atan2(ld(0),ld(1)));^ st(4,sin(ld(2))*sin(ld(3)));^ st(5,sin(ld(2))*cos(ld(3)));^ st(6,cos(ld(2)));^ </pre>
	For equisolid output, (see equisolid_to_xyz in v360.c)	<pre> st(0,sin(%OUT_H_FOV%/720*PI)*((2*X+1)/%OUT_W%-1));^ st(1,sin(%OUT_V_FOV%/720*PI)*((2*Y+1)/%OUT_H%-1));^ st(2,atan2(ld(1),ld(0)));^ st(3,hypot(ld(0),ld(1)));^ st(3,if(lt(ld(3),1),2*asin(ld(3)),PI));^ st(9,if(gt(ld(3),%OUT_H_FOV%/360*PI),%OUT_H%,0));^ st(4,sin(ld(3))*cos(ld(2)));^ st(5,sin(ld(3))*sin(ld(2)));^ st(6,cos(ld(3)));^ </pre>
	For stereographic output, (see stereographic_to_xyz in v360.c)	<pre> st(0,tan(%OUT_H_FOV%/720*PI)*((2*X+1)/%OUT_W%-1));^ st(1,tan(%OUT_V_FOV%/720*PI)*((2*Y+1)/%OUT_H%-1));^ st(2,atan2(ld(1),ld(0)));^ st(3,hypot(ld(0),ld(1)));^ st(3,2*atan(ld(3)));^ st(9,if(gt(ld(3),%OUT_H_FOV%/360*PI),%OUT_H%,0));^ st(4,sin(ld(3))*cos(ld(2)));^ st(5,sin(ld(3))*sin(ld(2)));^ st(6,cos(ld(3)));^ </pre>
	For orthographic output, (see orthographic_to_xyz in v360.c)	<pre> st(0,sin(%OUT_H_FOV%/360*PI)*((2*X+1)/%OUT_W%-1));^ st(1,sin(%OUT_V_FOV%/360*PI)*((2*Y+1)/%OUT_H%-1));^ st(2,atan2(ld(1),ld(0)));^ st(3,hypot(ld(0),ld(1)));^ st(3,if(lt(ld(3),1),asin(ld(3)),PI));^ st(9,if(gt(ld(3),%OUT_H_FOV%/360*PI),%OUT_H%,0));^ st(4,sin(ld(3))*cos(ld(2)));^ st(5,sin(ld(3))*sin(ld(2)));^ st(6,cos(ld(3)));^ </pre>

<p>Optional check for debugging: If the X,Y,Z vector in variables 4,5,6 isn't normalized, then print some variables and enter an endless loop.</p>	<p>For all formats</p>	<pre>if(gt(abs(1-sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6))),1e-6),print(66666666);print(X);print(Y);print(ld(4));print(ld(5));print(ld(6));while(1,0));^</pre>
<p>Optional input offset correction, this is usable if the input image was taken with a fisheye lens off-center in a dome. Input and output are in variables 4,5,6. For details see below.</p>	<p>For all formats This is the same as the h_offset and v_offset options in v360.</p>	<pre>st(4,ld(4)+%OFF_IN_X%);^ st(5,ld(5)+%OFF_IN_Y%);^ st(6,ld(6)+%OFF_IN_Z%);^ st(0,sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6)));^ st(4,ld(4)/ld(0));^ st(5,ld(5)/ld(0));^ st(6,ld(6)/ld(0));^</pre>
<p>Optional forward looking enlarging effect. %LENS_RAD% is the radius in degrees and %LENS_PWR% is the power (0=neutral). Input and output are in variables 4,5,6.</p>	<p>For all formats</p>	<pre>st(0,cos(%LENS_RAD%/180*PI));^ if(gt(ld(6),ld(0)),st(6,ld(6)+%LENS_PWR%*pow((ld(6)-ld(0))/(1-ld(0)),2)));^ st(0,sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6)));^ st(4,ld(4)/ld(0));^ st(5,ld(5)/ld(0));^ st(6,ld(6)/ld(0));^</pre>
<p>Optional ellipsoid distortion effect. %ELLIPSE% is the factor for enlarging the Y axis (1=neutral). Input and output are in variables 4,5,6.</p>	<p>For all formats</p>	<pre>st(5,%ELLIPSE%*ld(5));^ st(0,sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6)));^ st(4,ld(4)/ld(0));^ st(5,ld(5)/ld(0));^ st(6,ld(6)/ld(0));^</pre>
<p>Optional height over ground correction (for drone videos), only applied to one hemisphere. Makes a slope discontinuity at the horizon. %H% is the factor for the height (1=neutral). Input and output are in variables 4,5,6.</p>	<p>For all formats</p>	<pre>st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^ if(lt(ld(8),PI/2),st(8,atan(%H%*tan(ld(8))));^ st(4,sin(ld(8))*cos(ld(7)));^ st(5,sin(ld(8))*sin(ld(7)));^ st(6,cos(ld(8)));^</pre>

Optional height over ground correction, applied to both hemispheres. %H% is the factor for the height (1=neutral). Input and output are in variables 4,5,6.	For all formats	<pre> st(7,atan2(ld(5),ld(4))) ;^ st(8,acos(ld(6))) ;^ st(8,if(lt(ld(8),PI/2),atan(%H%*tan(ld(8))),PI-atan(%H%*tan(PI-ld(8)))));^ st(4,sin(ld(8))*cos(ld(7))) ;^ st(5,sin(ld(8))*sin(ld(7))) ;^ st(6,cos(ld(8))) ;^ </pre>
Optional black hole distortion effect, %RS% is the Schwarzschild radius in degrees. Deviation angle = $2 * RS / (R - RS)$ where R is the angular distance from the black hole. The parameter %S% does make the distortion smaller, 1 = realistic	For all formats	<pre> st(7,atan2(ld(5),ld(4))) ;^ st(8,acos(ld(6))) ;^ st(9,if(lte(ld(8),%RS%/180*PI),%OUT_W%,0));^ st(8,if(gt(ld(8),%RS%/180*PI),ld(8)-%S%*(%RS%/90*PI/(ld(8)-%RS%/180*PI)),0));^ st(4,sin(ld(8))*cos(ld(7))) ;^ st(5,sin(ld(8))*sin(ld(7))) ;^ st(6,cos(ld(8))) ;^ </pre>
Optional roll rotation (around Z axis), input and output are in variables 4,5,6.	For all formats	<pre> st(0,ld(4)*cos(%ROLL%/180*PI)-ld(5)*sin(%ROLL%/180*PI)) ;^ st(5,ld(5)*cos(%ROLL%/180*PI)+ld(4)*sin(%ROLL%/180*PI)) ;^ st(4,ld(0)) ;^ </pre>
Optional pitch rotation (around X axis), input and output are in variables 4,5,6.	For all formats	<pre> st(0,ld(5)*cos(%PITCH%/180*PI)-ld(6)*sin(%PITCH%/180*PI)) ;^ st(6,ld(6)*cos(%PITCH%/180*PI)+ld(5)*sin(%PITCH%/180*PI)) ;^ st(5,ld(0)) ;^ </pre>
Optional yaw rotation (around Y axis), input and output are in variables 4,5,6.	For all formats	<pre> st(0,ld(4)*cos(%YAW%/180*PI)+ld(6)*sin(%YAW%/180*PI)) ;^ st(6,ld(6)*cos(%YAW%/180*PI)-ld(4)*sin(%YAW%/180*PI)) ;^ st(4,ld(0)) ;^ </pre>

<p>Optional output offset correction, this is usable if the fisheye projector is placed off-center in the dome. Input and output are in variables 4,5,6.</p> <p>For details see below.</p>	<p>For all formats This isn't implemented in the v360 filter.</p>	<pre>st(0,%OFF_OUT_X%*ld(4)+%OFF_OUT_Y%*ld(5)+%OFF_OUT_Z%*ld(6));^ st(1,2*(%OFF_OUT_X%*%OFF_OUT_Y%*ld(4)*ld(5)));^ st(1,ld(1)+2*(%OFF_OUT_X%*%OFF_OUT_Z%*ld(4)*ld(6)));^ st(1,ld(1)+2*(%OFF_OUT_Y%*%OFF_OUT_Z%*ld(5)*ld(6)));^ st(1,ld(1)+ld(4)*ld(4)*(1-%OFF_OUT_Y%*%OFF_OUT_Z%*%OFF_OUT_Z%));^ st(1,ld(1)+ld(5)*ld(5)*(1-%OFF_OUT_X%*%OFF_OUT_X%-%OFF_OUT_Z%*%OFF_OUT_Z%));^ st(1,ld(1)+ld(6)*ld(6)*(1-%OFF_OUT_X%*%OFF_OUT_X%-%OFF_OUT_Y%*%OFF_OUT_Y%));^ st(0,ld(0)+sqrt(ld(1)));^ st(4,ld(4)*ld(0)-%OFF_OUT_X%);^ st(5,ld(5)*ld(0)-%OFF_OUT_Y%);^ st(6,ld(6)*ld(0)-%OFF_OUT_Z%);^</pre>
<p>Convert from X,Y,Z space to desired input format. Here the formulas are different for xmap and ymap files, the differences are marked in yellow. Adding ld(9) to the output in the last line is only required in one file, either xmap or ymap, and it's only required if unmapped pixels are possible for the selected output format. If variable 9 is set to a value larger than the output size, the color will be determined by remap=fill=...</p>	<p>For equirectangular input (see xyz_to_equirect in v360.c)</p>	<p>This is only for the xmap file: <code>st(7,atan2(ld(4),ld(6)));^ 0.5*%IN_W%*(1+ld(7)/%IN_H_FOV%*360/PI)' -frames 1 -y xmap.pgm</code></p> <p>This is only for the ymap file: <code>st(8,asin(ld(5)));^ 0.5*%IN_H%*(1+ld(8)/%IN_V_FOV%*360/PI)' -frames 1 -y ymap.pgm</code></p> <p>This is only for the ymap file, if non-mapped pixels are possible: <code>st(8,asin(ld(5)));^ ld(9)+0.5*%IN_H%*(1+ld(8)/%IN_V_FOV%*360/PI)' -frames 1 -y ymap.pgm</code></p>
	<p>For fisheye (equidistant) input (see xyz_to_fisheye in v360.c)</p>	<p><code>st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^</code></p> <p>This is only for the xmap file: <code>0.5*%IN_W%*(1+360/PI/%IN_H_FOV%*ld(8)*cos(ld(7)))' -frames 1 -y xmap.pgm</code></p> <p>This is only for the ymap file: <code>ld(9)+0.5*%IN_H%*(1+360/PI/%IN_V_FOV%*ld(8)*sin(ld(7)))' -frames 1 -y ymap.pgm</code></p>

	<p>For nonlinear fisheye input according to Paul Bourke's formula, see http://www.paulbourke.net/dome/fisheyecorrect/</p> <p>Set the parameters A, B, C, D and the correct field of view. This is slow because "root" is used.</p>	<pre>st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^ st(8,ld(8)/%IN_H_FOV%*360/PI);^ st(8,root(-ld(8)+%A%*ld(0)+%B%*pow(ld(0),2)+%C%*pow(ld(0),3)+%D %*pow(ld(0),4),1));^ This is only for the xmap file: 0.5*%IN_W%*(1+360/PI/%IN_H_FOV%*ld(8)*cos(ld(7)))' -frames 1 -y xmap.pgm This is only for the ymap file: ld(9)+0.5*%IN_H%*(1+360/PI/%IN_V_FOV%*ld(8)*sin(ld(7)))' -frames 1 -y ymap.pgm</pre>
	<p>For double fisheye (equidistant) input (see xyz_to_dfisheye in v360.c)</p>	<pre>st(7,hypot(ld(4),ld(5)));^ This is only for the xmap file: st(7,if(lt(ld(6),0),-1,1));^ st(8,hypot(ld(4),ld(5)));^ st(8,atan2(ld(7)*ld(8),ld(7)*ld(6))/ld(8));^ 0.25*%IN_W%*(2-ld(7)+ld(4)*ld(8)/%IN_H_FOV%*360/PI)' -frames 1 -y xmap.pgm This is only for the ymap file: st(8,atan2(ld(7),abs(ld(6)))/ld(7));^ 0.5*%IN_H%*(1+ld(5)*ld(8)/%IN_V_FOV%*360/PI)' -frames 1 -y ymap.pgm</pre>
	<p>For flat (rectilinear) input (see xyz_to_flat in v360.c)</p>	<pre>st(7,tan(acos(ld(6))));^ st(8,hypot(ld(4),ld(5))/ld(7));^ This is only for the xmap file: 0.5*%IN_W%*(1+ld(4)/ld(8)/tan(%IN_H_FOV%*PI/360))' -frames 1 -y xmap.pgm This is only for the ymap file: 0.5*%IN_H%*(1+ld(5)/ld(8)/tan(%IN_V_FOV%*PI/360))' -frames 1 -y ymap.pgm</pre>
	<p>For cylindrical input (see xyz_to_cylindrical in v360.c)</p>	<pre>This is only for the xmap file: st(7,atan2(ld(4),ld(6)));^ 0.5*%IN_W%*(1+ld(7)/%IN_H_FOV%*360/PI)' -frames 1 -y xmap.pgm This is only for the ymap file: st(8,asin(ld(5)));^ 0.5*%IN_H%*(1+tan(ld(8))/tan(%IN_V_FOV%/360*PI))' -frames 1 -y ymap.pgm</pre>

	<p>For planet input, this is a sphere seen from infinite distance. Exactly half of the sphere is visible. It's the same as orthographic input with 180° field of view.</p>	<p>This is only for the xmap file: 0.5*%IN_W%*(1+ld(4))' -frames 1 -y xmap.pgm</p> <p>This is only for the ymap file: st(9,if(lt(ld(6),0),%OUT_H%,0));^ ld(9)+0.5*%IN_H%*(1+ld(5))' -frames 1 -y ymap.pgm</p>
	<p>For planet input, this is a sphere seen from the finite distance %DIST%. Less than half of the sphere is visible. This isn't implemented in the v360 filter.</p>	<p>st(7,acos(ld(6)));^ st(9,if(gt(ld(7),PI/2-asin(1/%DIST%)),%OUT_H%,0));^ st(7,sin(ld(7))/(%DIST%-cos(ld(7))));^ st(7,ld(7)/tan(asin(1/%DIST%)));^ st(8,atan2(ld(4),ld(5)));^</p> <p>This is only for the xmap file: 0.5*%IN_W%*(1+ld(7)*sin(ld(8)))' -frames 1 -y xmap.pgm</p> <p>This is only for the ymap file: ld(9)+0.5*%IN_H%*(1+ld(7)*cos(ld(8)))' -frames 1 -y ymap.pgm</p>
	<p>For mirrorsphere input, this is a reflecting sphere seen from infinite distance. (see xyz_to_ball in v360.c)</p>	<p>st(7,atan2(ld(5),ld(4)));^ st(8,0.5*(PI/2+asin(ld(6))));^</p> <p>This is only for the xmap file: 0.5*%IN_W%*(1+cos(ld(8))*cos(ld(7)))' -frames 1 -y xmap.pgm</p> <p>This is only for the ymap file: ld(9)+0.5*%IN_H%*(1+cos(ld(8))*sin(ld(7)))' -frames 1 -y ymap.pgm</p>

	<p>For mirrorsphere input, this is a reflecting sphere seen from the finite distance %DIST%. This isn't implemented in the v360 filter.</p>	<pre>st(7,0.5*acos(ld(6)));^ st(9,if(gt(ld(7),PI/2-asin(1/%DIST%)),%OUT_H%,0));^ st(7,sin(ld(7))/(%DIST%-cos(ld(7))));^ st(7,ld(7)/tan(asin(1/%DIST%)));^ st(8,atan2(ld(4),ld(5)));^ This is only for the xmap file: 0.5*%IN_W%*(1+ld(7)*sin(ld(8)))' -frames 1 -y xmap.pgm</pre> <p>This is only for the ymap file: <code>ld(9)+0.5*%IN_H%*(1+ld(7)*cos(ld(8)))' -frames 1 -y ymap.pgm</code></p>
	<p>For equisolid input, (see xyz_to_equisolid in v360.c)</p>	<pre>st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^ This is only for the xmap file: st(8,sin(ld(8)/2)/sin(%IN_H_FOV%/720*PI));^ 0.5*%IN_W%*(1+ld(8)*cos(ld(7)))' -frames 1 -y xmap.pgm</pre> <p>This is only for the ymap file: <code>st(8,sin(ld(8)/2)/sin(%IN_V_FOV%/720*PI));^ ld(9)+0.5*%IN_H%*(1+ld(8)*sin(ld(7)))' -frames 1 -y ymap.pgm</code></p>
	<p>For stereographic input, (see xyz_to_stereographic in v360.c)</p>	<pre>st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^ This is only for the xmap file: st(8,tan(ld(8)/2)/tan(%IN_H_FOV%/720*PI));^ 0.5*%IN_W%*(1+ld(8)*cos(ld(7)))' -frames 1 -y xmap.pgm</pre> <p>This is only for the ymap file: <code>st(8,tan(ld(8)/2)/tan(%IN_V_FOV%/720*PI));^ ld(9)+0.5*%IN_H%*(1+ld(8)*sin(ld(7)))' -frames 1 -y ymap.pgm</code></p>

	For orthographic input, (see xyz_to_orthographic in v360.c)	<pre> st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^ This is only for the xmap file: st(8,sin(ld(8))/sin(%IN_H_FOV%/360*PI));^ 0.5*%IN_W%*(1+ld(8)*cos(ld(7)))' -frames 1 -y xmap.pgm This is only for the ymap file: st(8,sin(ld(8))/sin(%IN_V_FOV%/360*PI));^ ld(9)+0.5*%IN_H%*(1+ld(8)*sin(ld(7)))' -frames 1 -y ymap.pgm </pre>
Apply the remap filter	For all formats	<pre>ffmpeg -i in.png -i xmap.pgm -i ymap.pgm -lavfi remap=fill=green -frames 1 -y out.png</pre>

Note: You can choose the order of the yaw, pitch and roll rotations as desired, or you can omit the rotations if you don't need them. But keep in mind that here in "remap" the order is reversed. You have the output at the beginning and the input at the end. As shown above, the rotation order is yaw, pitch, roll.

Note: For clarity, each equation is written here in a new line. However in the command line there are no line feeds allowed. Either it must all be written in one long line, or you must terminate each line with a suitable character, depending on your batch interpreter. For example ^ for Windows batch files, as shown here.

Note: For double fisheye, the parameters IN_W, IN_H, OUT_W and OUT_H refer to the size of the whole image, however the parameters IN_H_FOV, IN_V_FOV, OUT_H_FOV and OUT_V_FOV refer to the field of view of each of the fisheye images. Not the total horizontal angle of the whole image.

Convert from pixel coordinates [0 ... W-1] to angle in radians [-FOV/2 ... +FOV/2] The angle is calculated for the center of the pixel.	<pre> angle = FOV * PI / 360 * ((2 * X + 1) / W - 1) with FOV = field of view in radians X = pixel coordinate W = image width in pixels </pre>
Convert from pixel coordinates [0 ... W-1] to normalized coordinates [-1 ... +1]: Result for X = 0 is (-1 + 1/W) Result for X = (W-1) is (1 - 1/W)	<pre> XN = (2 * X + 1) / W - 1 with X = pixel coordinate W = image width in pixels </pre>
Convert from normalized coordinates [-1 ... +1] to angle in radians [-FOV/2 ... +FOV/2]	<pre>angle = FOV * PI / 360 * XN</pre>
Convert from angle in radians [-FOV/2 ... +FOV/2]	<pre>X = 0.5 * W * (1 + angle * 360 / PI / FOV)</pre>

+FOV/2] to pixel coordinates [0 ... W-1]:

with
W = image width in pixels
FOV = field of view in degrees
angle = angle in degrees

When writing the numbers to a *.pgm file, is it safe to assume that the numbers are always rounded down to the next integer? Yes it is, as can be shown with this simple test:

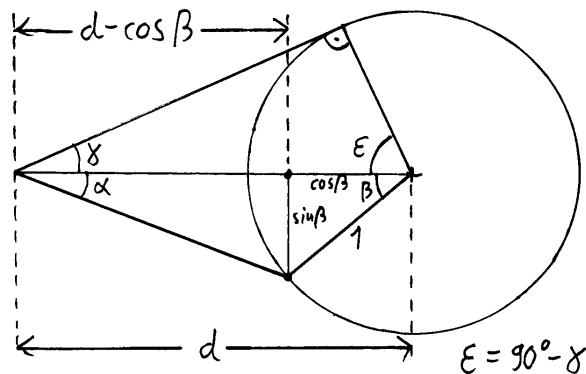
```
ffmpeg -f lavfi -i nullsrc=size=20x1 -vf format=pix_fmts=gray16le,geq='X/10' -frames 1 -y test.pgm
```

```
pause
```

This is the resulting file in a hex editor. It contains 10 pixels with 16-bit value 0 followed by 10 pixels with 16-bit value 1:

```
0x00: 50 35 0A 32 30 20 31 0A 36 35 35 33 35 0A 00 00
0x10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20: 00 00 00 01 00 01 00 01 00 01 00 01 00 01 00 01
0x30: 00 01 00 01 00 01
```

These are the formulas for the planet projection, with the camera at the left side:



$$\alpha = \arctan\left(\frac{\sin \beta}{d - \cos \beta}\right) \quad \gamma = \arcsin \frac{1}{d}$$

$$\beta = 2 \arctan\left(\frac{1 - \sqrt{1 + (1-d^2)\tan^2 \alpha}}{(d+1) \tan \alpha}\right)$$

See also: <https://www.wolframalpha.com/input/?i=tan%28alpha%29+%3D+sin%28beta%29+2F+28d+-+cos%28beta%29%29>

Distance d (%DIST% in the batch file)	gamma	epsilon
1.0353	75°	15°
1.1547	60°	30°
1.4142	45°	45°
2	30°	60°
3.8637	15°	75°
221 (Moon seen from earth, average distance)	0.26°	89.74°

2.114 Off-center fisheye projection

See <http://www.paulbourke.net/dome/>

and especially <http://paulbourke.net/dome/offaxisfisheyeprojection/> where unfortunately you can't find the formulas...

Example:

Let's assume input and output formats are fisheye. If the fisheye projector is shifted 0.5 to the side (halfway between the dome's center and edge), and if the optical axis of the projection lens is tilted 26.565° so that the optical axis is pointing to the dome's vertex, use these parameters to create the image:

```
set "YAW=26.565"      :: Yaw angle in degrees
set "OFF_IN_X=0"       :: Input X offset, normalized to the dome's radius
set "OFF_OUT_X=0.5"    :: Output X offset, normalized to the dome's radius
```

However if you have taken an image with a fisheye lens inside the dome, with the camera at the same place as above and also pointing to the dome's vertex, then this image can be converted back to a normal fisheye image with these parameters:

```
set "YAW=-26.565"     :: Yaw angle in degrees
set "OFF_IN_X=0.5"      :: Input X offset, normalized to the dome's radius
set "OFF_OUT_X=0"       :: Output X offset, normalized to the dome's radius
```

A similar effect can be realized with the "h_offset" and "v_offset" options of the v360 filter. But that's the inverse algorithm. It's usable if a fisheye image was taken off-center in a dome, and shall be corrected to the center of the dome.

```
set "IN=1200.png"        :: Fisheye test pattern from http://www.paulbourke.net/dome/testpattern/1200.png
set "OUT=out.png"         :: Fisheye output image

ffmpeg -i %IN% -lavfi v360=fisheye:fisheye:h_offset=0.5 -y %OUT%

pause
```

At first I thought that the offset correction for a off-center fisheye projector could be done very easy as follows:

```

st(4,ld(4)+%OFFSET_X%);^          (add the offset vector to the normalized input vector)
st(5,ld(5)+%OFFSET_Y%);^
st(6,ld(6)+%OFFSET_Z%);^
st(0,sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6)));^ (calculate the length)
st(4,ld(4)/ld(0));^              (re-normalize the vector)
st(5,ld(5)/ld(0));^
st(6,ld(6)/ld(0));^

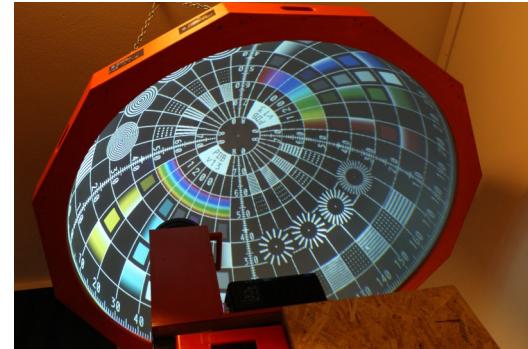
```

The above algorithm is the same as the "h_offset" and "v_offset" options of the v360 filter. But this approach doesn't work. It turned out that the inverse algorithm is required. The output is given and the input has to be found. First the normalization must be undone by a suitable unknown factor, so that after subtracting the offsets the result is already normalized. Undoing the normalization is possible, but complicated.

For details, see this discussion (in german): <https://groups.google.com/g/de.sci.mathematik/c/Dyaif04tgzQ>



Here the projection lens is at the dome's center. This is the 1.5m dome in the Sankt Andreasberg Observatory in the Harz Mountains in Germany.



Here the projection lens it at the OFF_OUT_Y=0.5 position. It's clearly visible that the image becomes brighter at the left side and darker at the right side.

This is an example for transformation from fisheye to off-center fisheye, where the lens is shifted half the dome's radius to the side and tilted 26.565°, so that the optical axis is pointing to the dome's vertex:

```

set "IN_W=1200"      :: Input width in pixels
set "IN_H=1200"      :: Input height in pixels
set "IN_H_FOV=180"    :: Input horizontal field of view in degrees

```

```

set "IN_V_FOV=180"    :: Input vertical field of view in degrees
set "OUT_W=1200"       :: Output width in pixels
set "OUT_H=1200"       :: Output height in pixels
set "OUT_H_FOV=180"    :: Output horizontal field of view in degrees
set "OUT_V_FOV=180"    :: Output vertical field of view in degrees
set "YAW=26.565"       :: Yaw angle in degrees
set "PITCH=0"          :: Pitch angle in degrees
set "ROLL=0"            :: Roll angle in degrees
set "OFF_IN_X=0"        :: Input X offset, normalized to the dome's radius
set "OFF_IN_Y=0"        :: Input Y offset, normalized to the dome's radius
set "OFF_IN_Z=0"        :: Input Z offset, negative is closer to dome's vertex
set "OFF_OUT_X=0.5"     :: Output X offset, normalized to the dome's radius
set "OFF_OUT_Y=0"        :: Output Y offset, normalized to the dome's radius
set "OFF_OUT_Z=0"        :: Output Z offset, negative is closer to dome's vertex

rem Create the xmap file

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^
st(0,%OUT_H_FOV%/180*((2*X+1)/%OUT_W%-1));^
st(1,%OUT_V_FOV%/180*((2*Y+1)/%OUT_H%-1));^
st(2,atan2(ld(1),ld(0)));^
st(3,PI/2*(1-hypot(ld(0),ld(1))));^
st(4,cos(ld(3))*cos(ld(2)));^
st(5,cos(ld(3))*sin(ld(2)));^
st(6,sin(ld(3)));^
st(4,ld(4)+%OFF_IN_X%);^
st(5,ld(5)+%OFF_IN_Y%);^
st(6,ld(6)+%OFF_IN_Z%);^
st(0,sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6)));^
st(4,ld(4)/ld(0));^
st(5,ld(5)/ld(0));^
st(6,ld(6)/ld(0));^
st(0,ld(4)*cos(%ROLL%/180*PI)-ld(5)*sin(%ROLL%/180*PI));^
st(5,ld(5)*cos(%ROLL%/180*PI)+ld(4)*sin(%ROLL%/180*PI));^
st(4,ld(0));^
st(0,ld(5)*cos(%PITCH%/180*PI)-ld(6)*sin(%PITCH%/180*PI));^
st(6,ld(6)*cos(%PITCH%/180*PI)+ld(5)*sin(%PITCH%/180*PI));^
st(5,ld(0));^
st(0,ld(4)*cos(%YAW%/180*PI)+ld(6)*sin(%YAW%/180*PI));^

```

```

st(6,ld(6)*cos(%YAW%/180*PI)-ld(4)*sin(%YAW%/180*PI));^
st(4,ld(0));^
st(0,%OFF_OUT_X%*ld(4)+%OFF_OUT_Y%*ld(5)+%OFF_OUT_Z%*ld(6));^
st(1,2*(%OFF_OUT_X%*%OFF_OUT_Y%*ld(4)*ld(5));^
st(1,ld(1)+2*(%OFF_OUT_X%*%OFF_OUT_Z%*ld(4)*ld(6));^
st(1,ld(1)+2*(%OFF_OUT_Y%*%OFF_OUT_Z%*ld(5)*ld(6));^
st(1,ld(1)+ld(4)*ld(4)*(1-%OFF_OUT_Y%*%OFF_OUT_Y%-%OFF_OUT_Z%*%OFF_OUT_Z%));^
st(1,ld(1)+ld(5)*ld(5)*(1-%OFF_OUT_X%*%OFF_OUT_X%-%OFF_OUT_Z%*%OFF_OUT_Z%));^
st(1,ld(1)+ld(6)*ld(6)*(1-%OFF_OUT_X%*%OFF_OUT_X%-%OFF_OUT_Y%*%OFF_OUT_Y%));^
st(0,ld(0)+sqrt(ld(1)));^
st(4,ld(4)*ld(0)-%OFF_OUT_X%);^
st(5,ld(5)*ld(0)-%OFF_OUT_Y%);^
st(6,ld(6)*ld(0)-%OFF_OUT_Z%);^
st(7,hypot(ld(5),ld(4)));^
st(8,atan2(ld(7),ld(6))/ld(7));^
0.5*%IN_W%*(1+ld(4)*ld(8)/%IN_H_FOV%*360/PI)' -frames 1 -y xmap.pgm

rem Create the ymap file

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^
st(0,%OUT_H_FOV%/180*((2*X+1)/%OUT_W%-1));^
st(1,%OUT_V_FOV%/180*((2*Y+1)/%OUT_H%-1));^
st(2,atan2(ld(1),ld(0)));^
st(3,PI/2*(1-hypot(ld(0),ld(1))));^
st(4,cos(ld(3))*cos(ld(2)));^
st(5,cos(ld(3))*sin(ld(2)));^
st(6,sin(ld(3)));^
st(4,ld(4)+%OFF_IN_X%);^
st(5,ld(5)+%OFF_IN_Y%);^
st(6,ld(6)+%OFF_IN_Z%);^
st(0,sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6)));^
st(4,ld(4)/ld(0));^
st(5,ld(5)/ld(0));^
st(6,ld(6)/ld(0));^
st(0,ld(4)*cos(%ROLL%/180*PI)-ld(5)*sin(%ROLL%/180*PI));^
st(5,ld(5)*cos(%ROLL%/180*PI)+ld(4)*sin(%ROLL%/180*PI));^
st(4,ld(0));^
st(0,ld(5)*cos(%PITCH%/180*PI)-ld(6)*sin(%PITCH%/180*PI));^
st(6,ld(6)*cos(%PITCH%/180*PI)+ld(5)*sin(%PITCH%/180*PI));^

```

```

st(5,ld(0));^
st(0,ld(4)*cos(%YAW%/180*PI)+ld(6)*sin(%YAW%/180*PI));^
st(6,ld(6)*cos(%YAW%/180*PI)-ld(4)*sin(%YAW%/180*PI));^
st(4,ld(0));^
st(0,%OFF_OUT_X%*ld(4)+%OFF_OUT_Y%*ld(5)+%OFF_OUT_Z%*ld(6));^
st(1,2*(%OFF_OUT_X%*%OFF_OUT_Y%*ld(4)*ld(5)));^
st(1,ld(1)+2*(%OFF_OUT_X%*%OFF_OUT_Z%*ld(4)*ld(6)));^
st(1,ld(1)+2*(%OFF_OUT_Y%*%OFF_OUT_Z%*ld(5)*ld(6)));^
st(1,ld(1)+ld(4)*ld(4)*(1-%OFF_OUT_Y%*%OFF_OUT_Y%-%OFF_OUT_Z%*%OFF_OUT_Z%));^
st(1,ld(1)+ld(5)*ld(5)*(1-%OFF_OUT_X%*%OFF_OUT_X%-%OFF_OUT_Z%*%OFF_OUT_Z%));^
st(1,ld(1)+ld(6)*ld(6)*(1-%OFF_OUT_X%*%OFF_OUT_X%-%OFF_OUT_Y%*%OFF_OUT_Y%));^
st(0,ld(0)+sqrt(ld(1)));^
st(4,ld(4)*ld(0)-%OFF_OUT_X%);^
st(5,ld(5)*ld(0)-%OFF_OUT_Y%);^
st(6,ld(6)*ld(0)-%OFF_OUT_Z%);^
st(7,hypot(ld(5),ld(4)));^
st(8,atan2(ld(7),ld(6))/ld(7));^
0.5*%IN_H%*(1+ld(5)*ld(8)/%IN_V_FOV%*360/PI)' -frames 1 -y ymap.pgm

```

```
ffmpeg -i 1200.png -i xmap.pgm -i ymap.pgm -lavfi remap -frames 1 -y out.png
```

```
pause
```

2.115 DLP Beamer output

A DLP beamer does always throw an off-axis image. The optical axis of the beamer is below the image (or above the image, if the beamer is mounted upside down).

(I'm still working on this chapter)

```
ffmpeg -f lavfi -i color=red:s=320x180 -f lavfi -i color=yellow:s=320x220 -lavfi vstack -frames 1 -y test1.png  
ffmpeg -f lavfi -i color=red:s=320x180 -frames 1 -y test2.png  
pause
```

test1.png has size 320x400 and the optical axis of the beamer is in the center. The red part at the top is the visible part.

Field of view from 500 pixels distance: $35.49^\circ \times 43.60^\circ$ $(2 * \text{atan}(320 / 2 / 500)) = 35.49^\circ$

test2.png has size 320x180 (16:9) and is only the visible part.

Field of view from 500 pixels distance: $35.40^\circ \times 20.41^\circ$

2.116 Remap from equirectangular to double-fisheye

This is the simple version with the v360 filter:

```
set "IN=equirectangular.png"    :: Input image
set "FOV=180"                  :: Output field of view in degrees
set "OUT=double_fish.png"       :: Output image

ffmpeg -i %IN% -lavfi v360=e:dfisheye:h_fov=%FOV%:v_fov=%FOV%:pitch=0 -y %OUT%

pause
```

Note: Pitch can be 0 or 90, depending on your needs.

Unfortunately the v360 filter with dfisheye output fills not only the two image circles with data, but instead also the outer areas. The workaround is to overlay a double-circular mask:

```
set "IN=equirectangular.png"    :: Input image
set "SIZE=1200x1200"            :: Size of half mask
set "FOV=180"                  :: Output field of view in degrees
set "OUT=double_fish.png"       :: Output image

ffmpeg -f lavfi -i color=black:s=%SIZE% -lavfi format=argb,geq=a='255*gt(hypot(((2*X+1)/H-1),((2*Y+1)/H-1)),1)':r=0:g=0:b=0,split,hstack -frames 1 -y mask.png

ffmpeg -i %IN% -i mask.png -lavfi v360=e:dfisheye:h_fov=%FOV%:v_fov=%FOV%:pitch=0,overlay -y %OUT%

pause
```

Note: Pitch can be 0 or 90, depending on your needs.

2.117 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.

```
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*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%
pause
```

The values in the xmap and ymap files can't be negative. If a value is greater 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:

```
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%*(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:

```
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=eqiurect:output=fisheye:h_fov=%FOV%:v_fov=%FOV%:pitch=-90 -y %OUT%  
  
pause
```

2.118 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.

```
set "IN=equirectangular_test.png"    :: Equirectangular input video
set "H=1200"                         :: Height of input video (width = 2 * height)
set "S=900"                           :: Size of square mirror sphere output video
set "OUT=mirror.png"                 :: 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%*(1-2/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:

```
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%*(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:

```
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

pause
```

Pitch=90 is for the sky in the center, pitch=-90 is for the ground in the center.

This batch file converts a double-fisheye video from Ricoh Theta V to a "mirror sphere" video:

```
set "IN=R0010017.mp4"      :: Input video
set "H=1920"                :: 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=20"                   :: Duration in seconds
set "S=1000"                 :: Output size
set "FPS=24"                  :: Output framerate
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=fisheye:e:ih_fov=%FOV%:iv_fov=%FOV%" -frames 1 -y mergemap.png

rem Merge the two fisheye images from the double-fisheye input video and transform it to a mirror-sphere 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=fisheye:e:yaw=180:ih_fov=%FOV%:iv_fov=%FOV%[d];[1]format=gbrp[e];[c][d]
[e]maskedmerge,drawbox=x=3*iw/4:y=ih/2:w=1:h=1:color=black,v360=e:ball:roll=-90:w=%S%:h=%S%" -r %FPS% -t %T% -y %OUT%
pause
```

Note: For "equirectangular" to "ball" transformation, the color that's used for unmapped pixels is not located at position 0, 0 (as with most other transformations), but instead at $3*iw/4, ih/2$.

This batch file does the same thing as the previous one, but uses another method for filling the unmapped pixels with a color (alpha_mask, scale2ref, overlay). Surprisingly this method is a little bit faster.

```
set "IN=R0010017.mp4"          :: Input video
set "H=1920"                   :: 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=20"                      :: Duration in seconds
set "S=1000"                    :: Output size
set "FPS=24"                    :: Output framerate
set "OUT=out17.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=fisheye:e:ih_fov=%FOV%:iv_fov=%FOV%" -frames 1 -y mergemap.png

rem Merge the two fisheye images from the double-fisheye input video and transform it to a mirror-sphere video

ffmpeg -i %IN% -i mergemap.png -f lavfi -i color=black:s=2x2 -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=fisheye:e:yaw=180:ih_fov=%FOV%:iv_fov=%FOV%[d];[1]format=gbrp[e];[c][d]
[e]maskedmerge,v360=e:ball:roll=-90:w=%S%:h=%S%:alpha_mask=1[f], [2][f]scale2ref,overlay=shortest=1" -r %FPS% -t %T% -y
%OUT%

pause
```

Note:

- First input of scale2ref: The video that shall be scaled
- Second input of scale2ref: The video which has the reference size
- First output of scale2ref: The scaled video
- Second output of scale2ref: A copy of the second input
- First input of overlay: The main (background) video
- Second input of overlay: The overlay (foreground) video

In this example scale2ref has no labels at its two output, which means overlay uses the same two streams in the same order as inputs.

This batch file does the same thing as the previous one, but uses the "geq" filter for filling the unmapped pixels with a color. This method is slower.

```
set "IN=R0010017.mp4"          :: Input video
set "H=1920"                   :: 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=20"                      :: Duration in seconds
set "S=1000"                    :: Output size
set "FPS=24"                    :: Output framerate
set "OUT=out17.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=fisheye:e:ih_fov=%FOV%:iv_fov=%FOV%" -frames 1 -y mergemap.png

rem Merge the two fisheye images from the double-fisheye input video and transform it to a mirror-sphere 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=fisheye:e:yaw=180:ih_fov=%FOV%:iv_fov=%FOV%[d];[1]format=gbrp[e];[c][d]
[e]maskedmerge,v360=e:ball:roll=-90:w=%S%:h=%S
%:alpha_mask=1,geq=r='if(gt(alpha(X,Y),127),r(X,Y),0)':g='if(gt(alpha(X,Y),127),g(X,Y),0)':b='if(gt(alpha(X,Y),127),b(X,Y),0)'"
-r %FPS% -t %T% -y %OUT%
pause
```

2.119 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

```

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/

2.120 How the "drawbox" filter works

```

ffmpeg -f lavfi -i color=gray:s=20x20 -vf format=rgb24,drawbox=x=4:y=4:width=6:height=6:thickness=1:color=red -frames 1
-y test.png

pause

```

The top left pixel of the box is at coordinates x, y.

The bottom right pixel of the box is at coordinates (x+width-1), (y+height-1).

The width and height of the box are exactly "width" and "height", independent of the thickness.

Setting the thickness greater than 1 doesn't change the outer dimensions of the box.

The number of pixels inside the box is (width-2*thickness), (height-2*thickness).

If you want a number of A pixels inside the box, you must set width or height to (A+2*thickness).

Note for "drawbox" filter: This filter doesn't support RGB formats!

2.121 Stitching together double-fisheye videos

The result is an equirectangular panorama video.

```
set "IN=double_fisheye.jpg"      :: Input video or picture
set "X1=198"                     :: X coordinate of center of left fisheye image
set "Y1=210"                     :: Y coordinate of center of left fisheye image
set "X2=595"                     :: X coordinate of center of right fisheye image
set "Y2=210"                     :: Y coordinate of center of right fisheye image
set "SR=192"                     :: Radius that is actually used from the source video
set "PW=1920"                    :: Width of panorama video
set "PH=960"                     :: Height of panorama video
set "OUT=out.jpg"                :: Output video or picture

rem Create the xmap file

ffmpeg -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

ffmpeg -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

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

pause
```

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.

Parameter	Result when decreasing the parameter	Result when increasing the parameter
X1	+-----+ upper half from left fisheye ← ↑ → ↓ ← +-----+ lower half from right fisheye +-----+	+-----+ upper half from left fisheye → ↓ ← ↑ → +-----+ lower half from right fisheye +-----+
Y1	+-----+ upper half from left fisheye ↑ → ↓ ← ↑ +-----+ lower half from right fisheye +-----+	+-----+ upper half from left fisheye ↓ ← ↑ → ↓ +-----+ lower half from right fisheye +-----+
X2	+-----+ upper half from left fisheye +-----+ → ↑ ← ↓ → lower half from right fisheye +-----+	+-----+ upper half from left fisheye +-----+ ← ↓ → ↑ ← lower half from right fisheye +-----+
Y2	+-----+ upper half from left fisheye +-----+ ↓ → ↑ ← ↓ lower half from right fisheye +-----+	+-----+ upper half from left fisheye +-----+ ↑ ← ↓ → ↑ lower half from right fisheye +-----+
SR	+-----+ upper half from left fisheye ↓ ↓ ↓ ↓ ↓ +-----+ ↑ ↑ ↑ ↑ ↑ lower half from right fisheye +-----+	+-----+ upper half from left fisheye ↑ ↑ ↑ ↑ ↑ +-----+ ↓ ↓ ↓ ↓ ↓ lower half from right fisheye +-----+

2.122 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

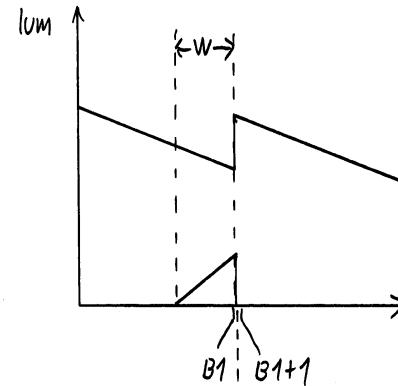
ffmpeg -i %IN% -vf "geq=cb_expr='cb(X,Y) :cr_expr='cr(X,Y)' :lum_expr='clip(lum(X,Y)+between(X,%B1%-1-%W%,%B1%-
1)*lerp(0,lum(%B1%,Y)-lum(%B1%-1,Y),(X-%B1%-1+%W%)/%W%)+between(X,%B2%-1-%W%,%B2%-1)
*lerp(0,lum(%B2%,Y)-lum(%B2%-1,Y),(X-%B2%-1+%W%)/%W%),0,255)',format=rgb24" -y out.png

pause
```

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:

```
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%,%B2%+%W%)*lerp(lum(%B2%-1,Y)-lum(%B2%,Y),0,(X-%B2%)/%W%),0,255)',format=rgb24" -y out.png

pause
```

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:

```
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

pause
```

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%

pause
```

Tested with this input video, downloaded in 1280x640 size: <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.

```
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%

pause

```

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.

```

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=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" -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%
```

2.123 Stitch double-fisheye images with alignment errors

If the optical axes of the lenses of a 360° camera aren't aligned to exactly opposite direction, the images don't fit together at the borders. We can assume that the first camera's axis is perfect and the second camera has three alignment errors: Yaw, pitch and roll. These errors can be corrected before stitching the images together.

Step 1: Make perfect double-fisheye and equirectangular test images (that means without alignment errors)

```
set "IN=1200.png"          :: Test pattern from http://www.paulbourke.net/dome/testpattern/1200.png
set "OUT=double_fisheye.png"

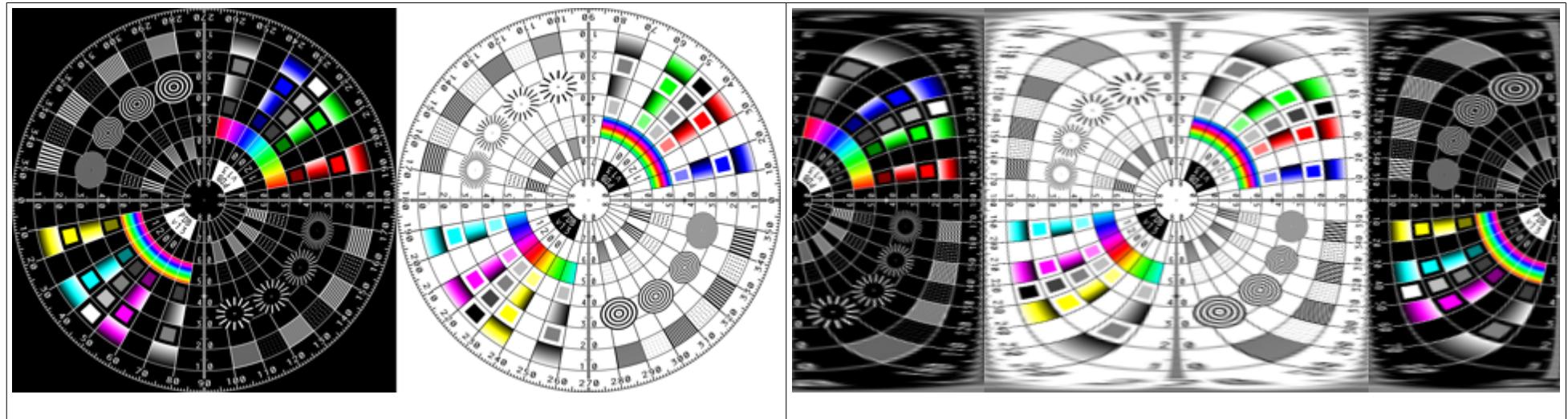
ffmpeg -i %IN% -i %IN% -lavfi "[0]transpose=1[left];[1]transpose=2,negate[right];[left][right]hstack" -y %OUT%

set "IN=double_fisheye.png"
set "OUT=equirectangular.png"

ffmpeg -i %IN% -lavfi "v360=input=dfisheye:output=e:ih_fov=180:iv_fov=180" -y %OUT%

pause
```

These are the perfect double-fisheye and equirectangular test images:



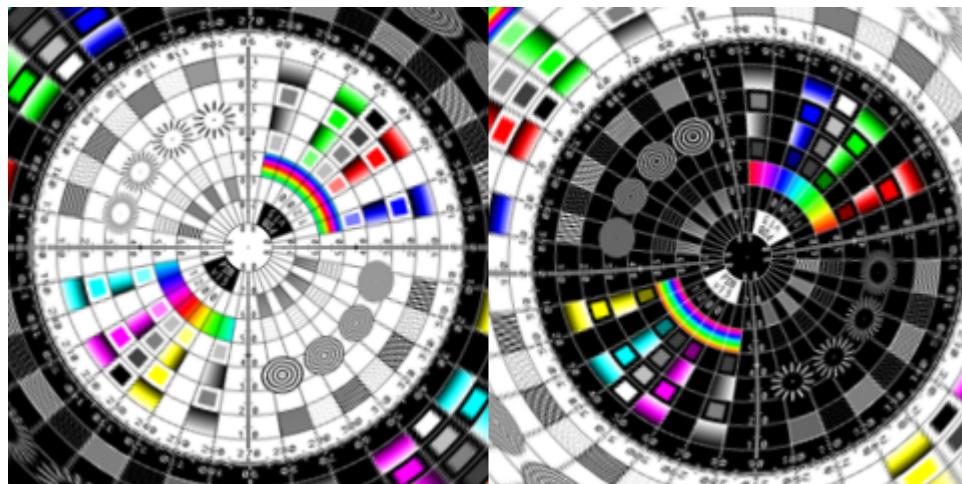
Step 2: Make a double-fisheye test image with alignment errors

```
set "IN=equiangular.png"
set "OUT=misaligned_double_fisheye.png"
set "FOV=200"      :: field of view
set "Y=-8"         :: yaw error of right fisheye lens
set "P=5"          :: pitch error of right fisheye lens
set "R=7"          :: roll error of right fisheye lens

ffmpeg -i %IN% -lavfi split[a][b];[a]v360=e:fisheye:h_fov=%FOV%:v_fov=%FOV%;[b]v360=e:fisheye:h_fov=%FOV%:v_fov=%FOV%:rorder=ypr:yaw='180+%Y%':pitch=%P%:roll=%R%;[a][b]hstack -y %OUT%

pause
```

This is the double-fisheye test image with alignment errors in the right half. You can see that it's shifted to the right and bottom, and rotated counter-clockwise:



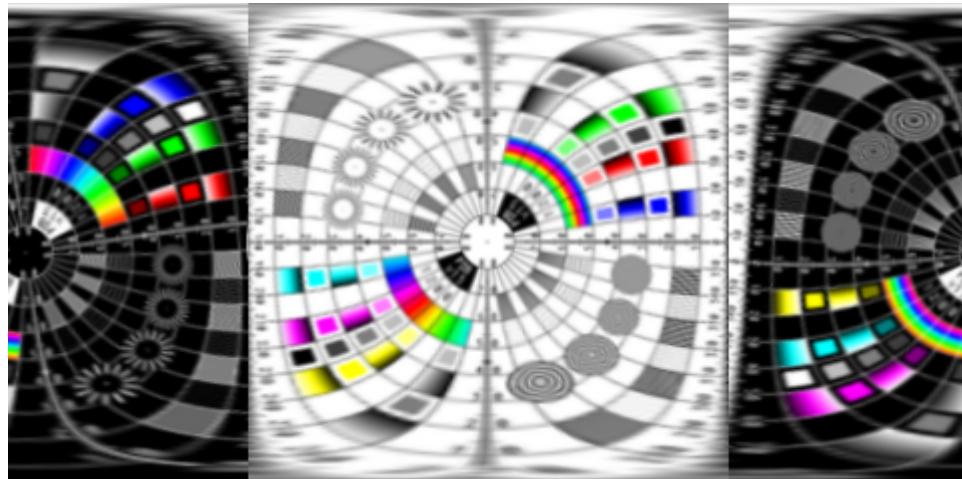
Step 3: When you now stich the fisheye images together, you will see the alignment errors:

```
set "IN=misaligned_double_fisheye.png"
set "OUT=eqirect_misaligned.png"
set "FOV=200"    :: field of view

ffmpeg -i %IN% -lavfi "split[a][b];[a]crop=ih:ih:0:0,v360=fisheye:fisheye:ih_fov=%FOV%:iv_fov=%FOV%
%:h_fov=180:v_fov=180[a];[b]crop=ih:ih:0,0,v360=fisheye:fisheye:ih_fov=%FOV%:iv_fov=%FOV%:h_fov=180:v_fov=180[b];[b]
[a]hstack,v360=dfisheye:e:ih_fov=180:iv_fov=180" -y %OUT%

pause
```

This is the stitched equirectangular image, you can clearly see the alignment errors:



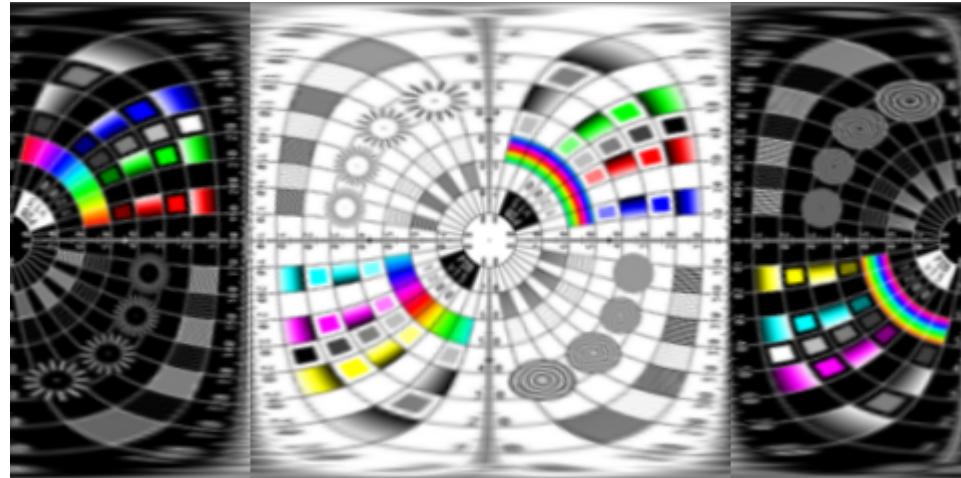
Step 4: Compensate the alignment errors before stitching the images together:

```
set "IN=misaligned_double_fisheye.png"
set "OUT=equirect_corrected.png"
set "FOV=200"      :: field of view
set "Y=8"          :: yaw error of right fisheye lens
set "P=-5"         :: pitch error of right fisheye lens
set "R=-7"         :: roll error of right fisheye lens

ffmpeg -i %IN% -lavfi "split[a][b];[a]crop=ih:ih:0:0,v360=fisheye:fisheye:ih_fov=%FOV%:iv_fov=%FOV
%:h_fov=180:v_fov=180[a];[b]crop=ih:ih:0,v360=fisheye:fisheye:ih_fov=%FOV%:iv_fov=%FOV
%:h_fov=180:v_fov=180:rorder=rpy:yaw=%Y%:pitch=%P%:roll=%R%[b];[b][a]hstack,v360=dfisheye:e:ih_fov=180:iv_fov=180" -y
%OUT%

pause
```

This is the corrected equirectangular output image:



The tricky part is to find the best values for FOC, Y, P and R by try and error. In this example I did already know the correct values, they are the same as in step 2 but with opposite sign. Please note that the rotation order must also be reversed.

2.124 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.

```
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%
pause
```

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:

```
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%
pause
```

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%' -windowFullscreen 1 -f sdl2 -
```

Please note that the sdl2 output doesn't play audio. The Windows taskbar remains visible in fullscreen 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.

In this example a 180° fisheye image is used as the background and a flat image is overlaid above the horizon. This should also work for videos instead of images.

```
set "BG=fisheye.jpg"      :: Fisheye background image (or video)
set "FG=flat.jpg"         :: Flat foreground image (or video)
set "UP=15"                :: This angle is the height of the center of the flat image above the horizon (in degrees)
set "ID=45"                :: Diagonal of the foreground image in degrees
set "S=3648"               :: Size of the square fisheye input and output images (or videos)
set "OUT=out.jpg"          :: Output image (or video)

ffmpeg -i %BG% -i %FG% -lavfi "[1]v360=input=flat:output=fisheye:id_fov=%ID%:h_fov=180:v_fov=180:w=%S%:h=%S%:pitch='90-
%UP%':alpha_mask=1[fg];[0][fg]overlay" -y %OUT%

pause
```

This is the same as before, but the flat image is duplicated, so that the same image is shown at the north and south horizon:

```
set "BG=fisheye.jpg"          :: Fisheye background image (or video)
set "FG=flat.jpg"             :: Flat foreground image (or video)
set "UP=15"                   :: This angle is the height of the center of the flat image above the horizon (in degrees)
set "ID=45"                   :: Diagonal of the foreground image in degrees
set "S=3648"                  :: Size of the square fisheye input and output images (or videos)
set "OUT=out.jpg"             :: Output image (or video)

ffmpeg -i %BG% -i %FG% -lavfi "[1]v360=input=flat:output=fisheye:id_fov=%ID%:h_fov=180:v_fov=180:w=%S%:h=%S%:pitch='90-%UP%':alpha_mask=1,split[fg1][fg2];[fg2]rotate=PI[fg3];[0][fg1]overlay[a];[a][fg3]overlay" -y %OUT%

pause
```

Some explanations for this example:

[0] and [1] are predefined labels for the input files.

[0] is the first input, in this case "in.png"

[1] is the second input, in this case "in.mp4"

[fg1] [fg2] [fg3] and [a] are just labels and you can change them if you want.

"alpha_mask" is an option of the v360 filter. In this case it's used to make the outer area of the rectangular video transparent.

<https://www.ffmpeg.org/ffmpeg-all.html#v360>

"split" is a filter that has one input and two (or more) outputs. It is here used to duplicate the foreground video.

https://www.ffmpeg.org/ffmpeg-all.html#split_002c-asplit

"rotate" is a filter that rotates the video. In this case it's used to rotate one of the foreground videos to the other side of the dome.

<https://www.ffmpeg.org/ffmpeg-all.html#rotate>

"overlay" is a filter that has two inputs and one output. Because in this case we want to overlay two videos, we must use it two times.

<https://www.ffmpeg.org/ffmpeg-all.html#overlay-1>

2.125 Rotating the earth, moon or planet

If the surface of the planet is given as an equirectangular image, then things are quite simple:

```
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 "XP=900"                 :: X position of planet  
set "YP=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%:alpha_mask=1,scale=%D%: %D%[a],[0][a]overlay=x=%XP%:y=%YP%" -t %T% -y out.mp4  
  
pause
```

This is the same as the previous example, but with day/night and twilight zone added:

```
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 "SIZE=1024x512"           :: Size for mergemap, must be the same as the equirectangular input video  
set "TW=20"                   :: Width of twilight zone in degrees  
set "DEC=23.5"                :: Declination of light source (sun), 0° for spring equinox, +23.5° for summer solstice,  
                                :: 0° for fall equinox, -23.5° for winter solstice  
set "DS=10"                   :: Brightness of the dark side, 0 for black, 255 for full brightness  
set "S=-0.005"                :: Rotation speed, 1.0 means one full revolution per frame  
set "Y=90"                    :: Yaw angle  
set "P=50"                    :: Pitch angle  
set "R=0"                     :: Roll angle  
set "D=400"                   :: Diameter of planet  
set "XP=800"                  :: X position of planet  
set "YP=400"                  :: y position of planet  
set "T=10"                    :: Length of output video  
  
ffmpeg -f lavfi -i nullsrc=size=%SIZE% -vf "format=gray8,geq='clip((1+180*(Y-H/2)*128/(%TW%*(H/2))),%DS  
%,255)',v360=e:e:pitch=%DEC%+90,format=rgb24" -frames 1 -y mergemap.png  
  
ffmpeg -loop 1 -i %BG% -loop 1 -i %IN% -f lavfi -i color=black:size=%SIZE%,format=rgb24 -i mergemap.png -lavfi  
"[1]scroll=h=%S%,format=rgb24[a];[2][a][3]maskedmerge,v360=e:perspective:yaw=%Y%:pitch=%P%:roll=%R%:alpha_mask=1,scale=  
%D%: %D%[a],[0][a]overlay=x=%XP%:y=%YP%" -t %T% -y out.mp4  
  
pause
```

Warning: Don't use the v360 filter's "perspective" projection if you need mathematically correct behaviour. The "perspective" projection depends somehow on the option "v_fov", but the exact meaning of this option is unknown in this context. Better use a workaround with "remap" filter. I did try to reverse-engineer the source code to find out the meaning of the "v_fov" option, but this approach wasn't successful.

If a normal perspective image of the planet is given, then things are more complicated. It's clear that the perspective image contains only half of the planet's surface. The other half must be replaced by a color.

Another problem is that FFmpeg's "v360" filter allows perspective images only for output, but not for input. In this case a workaround with the "remap" filter is required. As described above, it's also recommended to use the "remap" workaround for "perspective" output.

Now I'd like to describe how to create an image of the moon, as it appears from the side. It's impossible to take such an image from the earth, because the moon is always facing (roughly) the same side towards the earth. The first step is to take an image of the moon with a telescope or a telephoto lens. The second step is to measure in this image the moon's center coordinates and its radius. In case of a full moon this is quite simple, but for other moon phases it's more difficult. One approach is to measure the coordinates of three points on the moon's edge. These points should have wide spacing from each other. From the coordinates of these three points it's possible to calculate the center coordinates and the radius.

Because it's time consuming to do this with a pocket calculator, I wrote a small C# program for the job. The source code can be downloaded here:
<http://www.astro-electronic.de/source/FindCenterOfCircle.zip>

Now the center coordinates and the radius are known and can be inserted in this batch file. In the output image the observer is rotated 61° to the left:

```

set "IN=moon.jpg"          :: Input image
set "XC=927.7"            :: X coordinate of center of planet in input image
set "YC=2310.3"           :: Y coordinate of center of planet in input image
set "R=2070.0"             :: Radius of planet in input image
set "W=4000"               :: Width of intermediate equirectangular image
set "H=2000"               :: Height of intermediate equirectangular image
set "FILL=blue"            :: Fill color for no-data area
set "YAW=61"                :: Rotation angle
set "B=250"                 :: Width of added border
set "OUT=out.png"          :: Output image

rem Create the xmap file, from "perspective" input image to intermediate equirectangular image
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16le,geq='st(0,(X-%H%)*PI/%H%);if(between(ld(0),-0.5*PI,0.5*PI),%XC%+%R%*sin(ld(0))*cos((Y-0.5*%H%)*PI/%H%),-1)' -frames 1 -y xmap.pgm

rem Create the ymap file, from "perspective" input image to intermediate equirectangular image
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16le,geq='st(0,(X-%H%)*PI/%H%);if(between(ld(0),-0.5*PI,0.5*PI),%YC%+%R%*sin((Y-0.5*%H%)*PI/%H%),-1)' -frames 1 -y ymap.pgm

rem Rotate the moon
ffmpeg -i %IN% -i xmap.pgm -i ymap.pgm -lavfi "format=rgb24,remap=fill=%FILL%,v360=e:perspective:yaw=%YAW%:alpha_mask=1,pad=iw+2*B%:ih+2*B%:B%:B%" -y %OUT%

pause

```

Please note that the above batch file uses the incorrect "perspective" output of the v360 filter. The next batch file is better. It uses the "remap" workaround instead of the "perspective" projection of the "v360" filter:

```

set "IN=moon.jpg"          :: Input image
set "XC=927.7"             :: X coordinate of center of planet in input image
set "YC=2310.3"             :: Y coordinate of center of planet in input image
set "R=2070.0"              :: Radius of planet in input image
set "H=5000"                :: Height of intermediate equirectangular image
set "S=5000"                :: Width and height of output image
set "R2=2400"               :: Radius of moon in output image
set "FILL=blue"              :: Fill color for no-data area
set "LON=60"                 :: Rotation angle in longitude
set "LAT=0"                  :: Rotation angle in latitude
set "OUT=out.png"            :: Output image
set /a "W=2*%H%"           :: Width of output image
set /a "S2=%S%/2"           :: Height of output image

rem Create the xmap1 file, from "perspective" input image to intermediate equirectangular image
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16le,geq='st(0,(X-%H%)*PI/%H%);if(between(ld(0),-0.5*PI,0.5*PI),%XC%+%R%*sin(ld(0))*cos((Y-0.5*%H%)*PI/%H%),-1)' -frames 1 -y xmap1.pgm

rem Create the ymap1 file, from "perspective" input image to intermediate equirectangular image
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16le,geq='st(0,(X-%H%)*PI/%H%);if(between(ld(0),-0.5*PI,0.5*PI),%YC%+%R%*sin((Y-0.5*%H%)*PI/%H%),-1)' -frames 1 -y ymap1.pgm

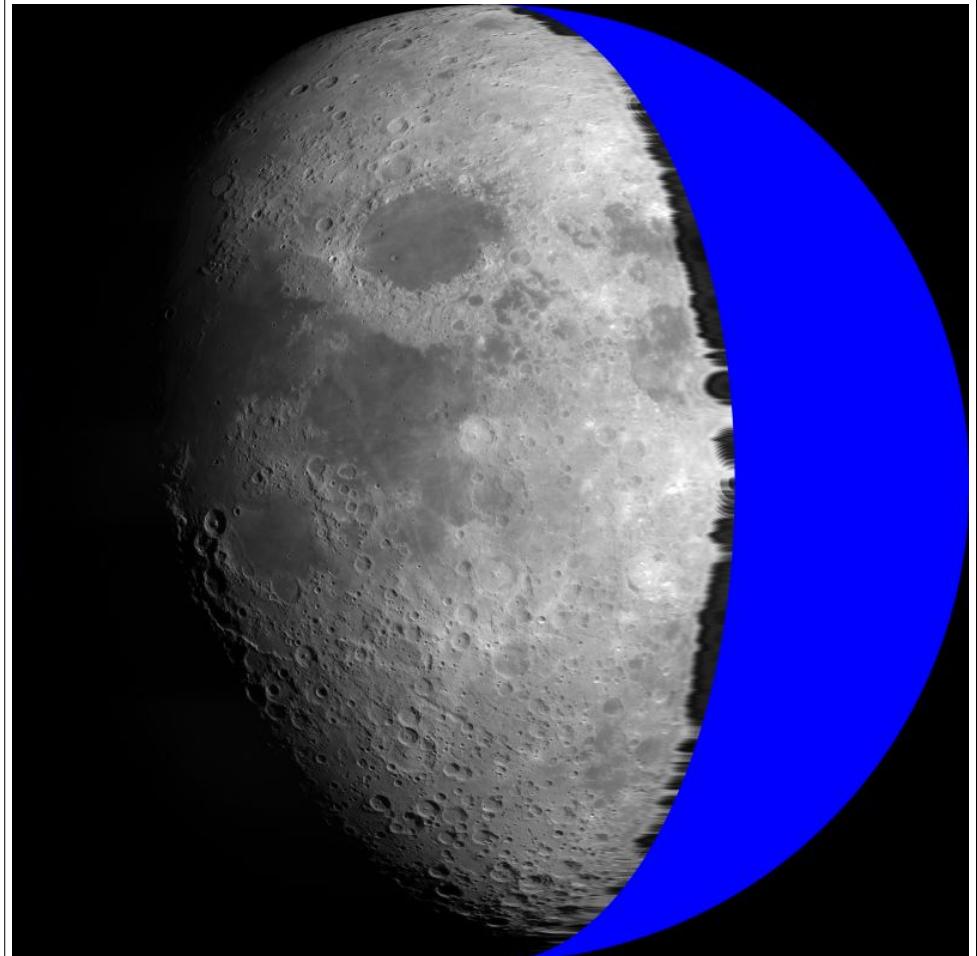
rem Create the xmap2 file, from intermediate equirectangular image to "perspective" output image
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=gray16le,geq='if(lt(hypot(X-%S2%,Y-%S2%),%R2%),%H%*(1+asin((X-%S2%)/%R2%)/cos(asin((Y-%S2%)/%R2%))/PI),-1)' -frames 1 -y xmap2.pgm

rem Create the ymap2 file, from intermediate equirectangular image to "perspective" output image
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=gray16le,geq='if(lt(hypot(X-%S2%,Y-%S2%),%R2%),%H%*(0.5+asin((Y-%S2%)/%R2%)/PI),-1)' -frames 1 -y ymap2.pgm

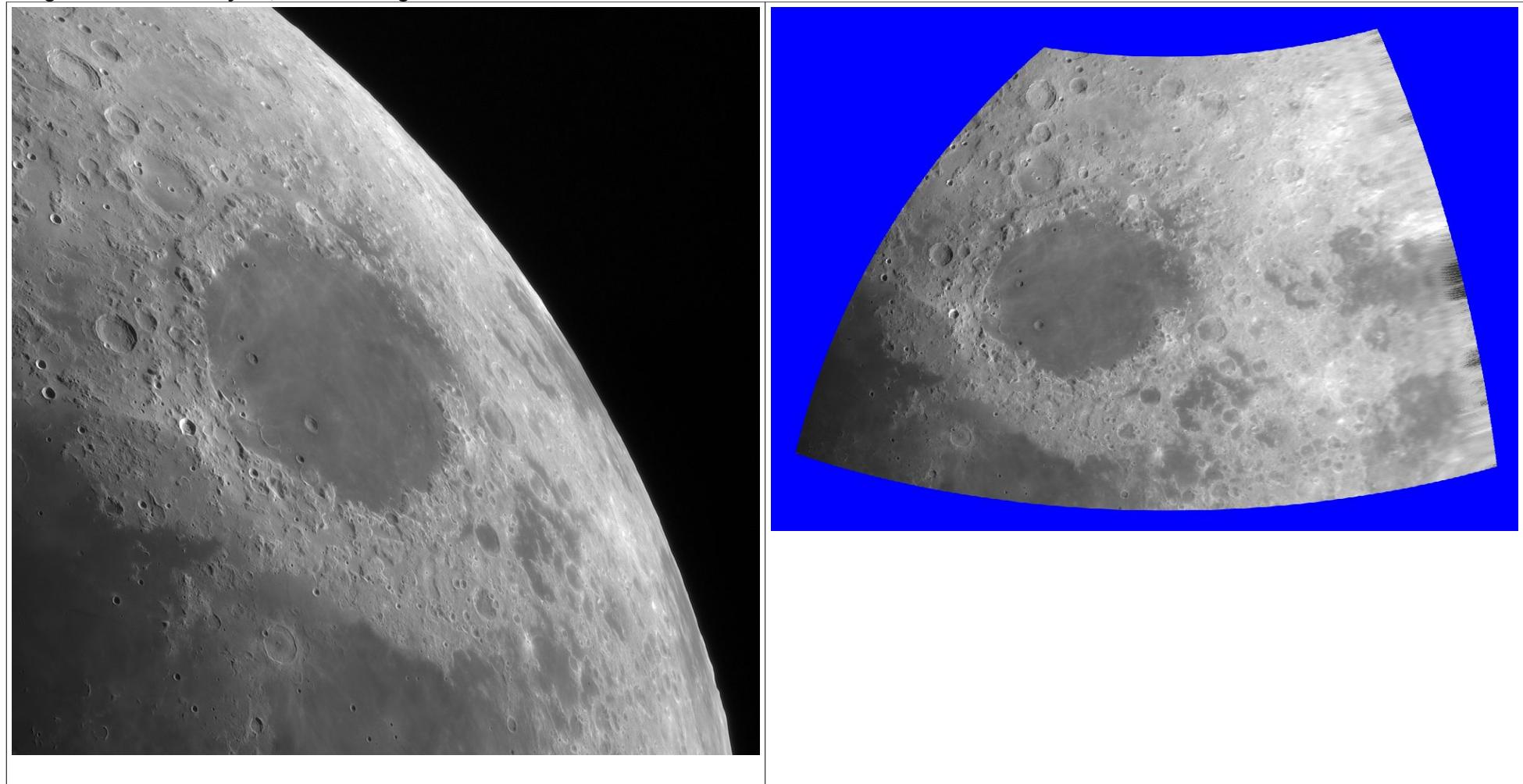
rem Rotate the moon
ffmpeg -i %IN% -i xmap1.pgm -i ymap1.pgm -i xmap2.pgm -i ymap2.pgm -lavfi "format=rgb24[5];[5][1][2]remap=fill=%FILL%,v360=e:e:yaw=%LON%:pitch=%LAT%[6];[6][3][4]remap" -y %OUT%
pause

```

Here are the input and output images. Input image taken by me 2018-05-20. You can see in the input image that Mare Crisium (at 2 o'clock) appears elongated in N-S direction, but when the point of view was moved 60° to the right you realize that in fact it's elongated in E-W direction. The no-data area is shown blue:



The script does also work if the input image contains only a part of the moon. The only requirement is that a large enough part of the moon's edge is visible, for calculating the center coordinates and the radius. It's no problem if the moon's center is outside the image. In this image you see Mare Marginis and Mare Smythii, which are right of Mare Crisium:



Compare with a real map of the moon (select the projection "Lunar Globe 3D"): <https://quickmap.lroc.asu.edu/projections>

2.126 Live rotation of the moon

A live video of the moon is taken with the GH5S camera, connected to a telescope. The signal is captured with a HDMI to USB converter, and then processed in realtime so that the libration area becomes visible. The left half of the display shows the input video with a red frame. The telescope must be positioned so that the moon fits exactly in the frame. The right half of the display shows the rotated moon. Some variables must be set in the batch file before running: Moon radius, latitude and longitude rotation angles.

```
set "IW=1920"          :: Width of the input video
set "IH=1080"          :: Height of the input video
set "R=512"            :: Radius of moon in input image, should be less than IH/2
                      :: R = W * D * pi * F / 21600 / C
                      :: with W = width in pixels (output of HDMI converter), here 1920
                      :: D = diameter of moon in arc minutes, here 31.75'
                      :: F = focal length of telescope in mm, here 1040
                      :: C = chip width in mm, here 18.8 for GH5S in FHD or 4K mode
set "H=1700"           :: Height of intermediate equirectangular image, a good value is input height * pi / 2
set "S=960"             :: Width and height of each half of the output image
set "R2=450"            :: Radius of moon in output image, should be smaller than S/2
set "FRAME=red"          :: Color of frame
set "FILL=blue"          :: Fill color for no-data area
set "LON=60"             :: Longitude rotation angle
set "LAT=0"              :: Latitude rotation angle
set /a "IH2=%IH%/2"
set /a "W=2*%H%"
set /a "S2=%S%/2"

rem Create the xmap1 file, from "perspective" input image to intermediate equirectangular image
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16le,geq='st(0,(X-%H%)*PI/%H%);if(between(ld(0),-0.5*PI,0.5*PI),%IH2%+%R%*sin(ld(0))*cos((Y-0.5*%H%)*PI/%H%),-1)' -frames 1 -y xmap1.pgm

rem Create the ymap1 file, from "perspective" input image to intermediate equirectangular image
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16le,geq='st(0,(X-%H%)*PI/%H%);if(between(ld(0),-0.5*PI,0.5*PI),%IH2%+%R%*sin((Y-0.5*%H%)*PI/%H%),-1)' -frames 1 -y ymap1.pgm

rem Create the xmap2 file, from intermediate equirectangular image to "perspective" output image
ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=gray16le,geq='if(lt(hypot(X-%S2%,Y-%S2%),%R2%),%H%*(1+asin((X-%S2%)/
```

```

%R2%)/cos(asin((Y-%S2%)/%R2%))/PI),-1)' -frames 1 -y xmap2.pgm

rem Create the ymap2 file, from intermediate equirectangular image to "perspective" output image

ffmpeg -f lavfi -i nullsrc=size=%S%x%S% -vf format=gray16le,geq='if(lt(hypot(X-%S2%,Y-%S2%),%R2%),%H%*(0.5+asin((Y-%S2%)/%R2%)/PI),-1)' -frames 1 -y ymap2.pgm

rem Live processing

ffmpeg -f dshow -video_size %IW%x%IH% -framerate 5 -pixel_format yuyv422 -i video="USB Video" -i xmap1.pgm -i ymap1.pgm
-i xmap2.pgm -i ymap2.pgm -lavfi "crop=ih:ih,format=rgb24,split[a][b];[a]drawbox=x=ih/2-%R%-3:y=ih/2-%R%-3:w=2*%R%
+6:h=2*%R%+6:color=%FRAME%,format=rgb24,scale=%S%:%S%[left];[b][1][2]remap=fill=%FILL%,v360=e:e:yaw=%LON%:pitch=%LAT%
[e];[e][3][4]remap,format=rgb24[right];[left][right]hstack" -window_x 0 -window_y 0 -f sdl2 -
pause

```

This is a live screenshot:



2.127 Insert a crosshair in a live video

A live video is captured with a HDMI to USB converter, and then a crosshair is inserted before the video is shown on the computer screen.

This can be used for checking if a telescope tracks the stars correctly.

```
set "IW=1920"      :: Width
set "IH=1080"      :: Height
set "C=white"       :: Color of crosshair
set "G=10"          :: Radius of clear area in the center

ffmpeg -f dshow -video_size %IW%x%IH% -framerate 5 -pixel_format yuyv422 -i video="USB Video" -lavfi
"format=rgb24,drawbox=x=iw/2:y=0:w=1:h=ih/2-%G%:color=%C%,drawbox=x=iw/2:y=ih/2+%G%:w=1:h=ih/2-%G%:color=%C
%,drawbox=x=0:y=ih/2:w=iw/2-%G%:h=1:color=%C%,drawbox=x=iw/2+%G%:y=ih/2:w=iw/2-%G%:h=1:color=%C%,format=rgb24" -window_x
0 -window_y 0 -f sdl2 -

pause
```

2.128 Enlarge the corners of a live video

This script enlarges the four corners of a live video by an adjustable factor. If you set the factor to 1, then the video is shown unchanged. The live video is captured by a HDMI to USB converter. Can be used for adjusting telescope optics.

```
set "IW=1920"    :: Width
set "IH=1080"    :: Height
set "F=5"         :: Enlarging factor

ffmpeg -f dshow -video_size %IW%x%IH% -framerate 5 -pixel_format yuyv422 -i video="USB Video" -lavfi
"format=rgb24,split=4[a][b][c][d];[a]crop=iw/2/%F%:ih/2/%F%:0:0,scale=iw*%F%:ih*%F%[aa],[b]crop=iw/2/%F%:ih/2/%F%:iw-
iw/2/%F%:0,scale=iw*%F%:ih*%F%[bb],[c]crop=iw/2/%F%:ih/2/%F%:0:ih-ih/2/%F%,scale=iw*%F%:ih*%F%[cc],[d]crop=iw/2/%F%
:ih/2/%F%:iw-iw/2/%F%:ih-ih/2/%F%,scale=iw*%F%:ih*%F%[dd],[aa][bb][cc][dd]xstack=inputs=4:layout=0_0|w0_0|0_h0|
w0_h0,format=rgb24" -window_x 0 -window_y 0 -f sdl2 -

pause
```

This is the same as before, but the input video is grabbed from the first monitor and the result is shown on the second monitor. You can use this script with any software that uses the full screen of the first monitor.

```
set "IW=1920"    :: Width of first monitor
set "IH=1080"    :: Height of first monitor
set "F=5"         :: Enlarging factor

ffmpeg -f gdigrab -video_size %IW%x%IH% -i desktop -lavfi "format=rgb24,split=4[a][b][c][d];[a]crop=iw/2/%F%:ih/2/%F%
:0:0,scale=iw*%F%:ih*%F%[aa],[b]crop=iw/2/%F%:ih/2/%F%:iw-iw/2/%F%:0,scale=iw*%F%:ih*%F%[bb],[c]crop=iw/2/%F%:ih/2/%F%
:0:ih-ih/2/%F%,scale=iw*%F%:ih*%F%[cc],[d]crop=iw/2/%F%:ih/2/%F%:iw-iw/2/%F%:ih-ih/2/%F%,scale=iw*%F%:ih*%F%[dd],[aa]
[bb][cc][dd]xstack=inputs=4:layout=0_0|w0_0|0_h0|w0_h0,format=rgb24" -window_x %IW% -window_y 0 -f sdl2 -

pause
```

Note: "-window_x 1920" near the end of the command line means that the output begins at x=1920, that's the left edge of the second monitor (if the width of the first monitor is 1920).

This is an example for enlarging 9 windows from topleft / topcenter / topright / centerleft / center / centerright / bottomleft / bottomcenter / bottomright:

```
set "P=200"      :: Window size in pixels
set "F=2"        :: Enlarging factor

ffmpeg -i input.mp4 -lavfi "format=rgb24,split=3[a][b][c];[a]crop=iw:%P%:0:0,scale=iw:ih*%F%[aa],[b]crop=iw:%P%:0:ih/2-%P%/2,scale=iw:ih*%F%[bb],[c]crop=iw:%P%:0:ih-%P%,scale=iw:ih*%F%[cc],[aa][bb][cc]vstack=3,split=3[a][b][c];[a]crop=%P%:ih:0:0,scale=iw*%F%:ih[aa],[b]crop=%P%:ih:iw/2-%P%/2:0,scale=iw*%F%:ih[bb],[c]crop=%P%:ih:iw-%P%:0,scale=iw*%F%:ih[cc],[aa][bb][cc]hstack=3,format=rgb24" -frames 1 -y test.png

pause
```

2.129 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.

```
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:

```

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

pause

```

Note: For $C \rightarrow \infty$ 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:

```

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

pause

```

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

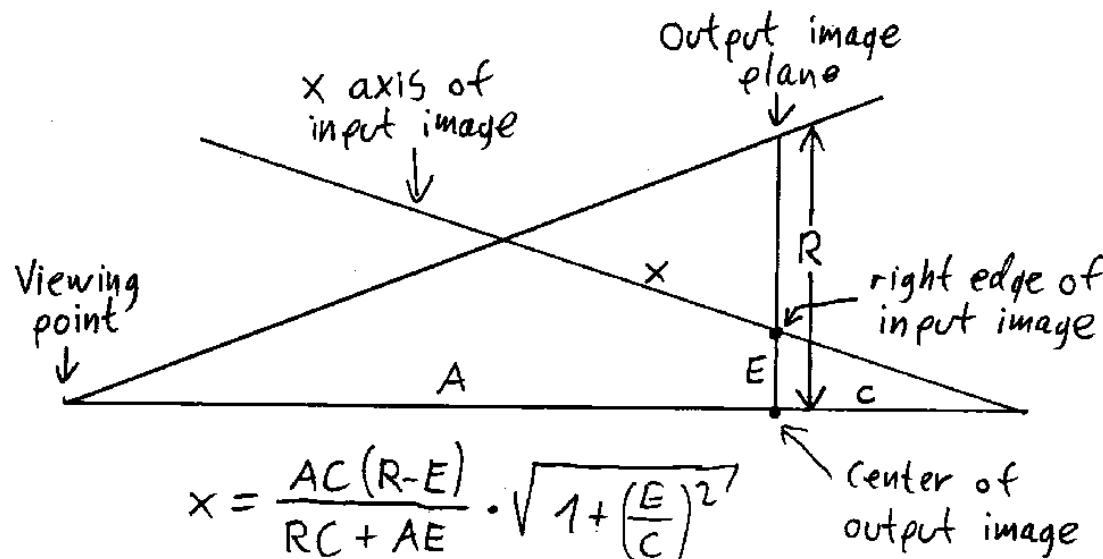
ffmpeg -f lavfi -i nullsrc=size=%PW%x%PH% -vf format=pix_fmts=gray16le,geq='%IH%*(mod(hypot(%PW%/2-X,%PH%/2-Y)/%S%+0.5-atan2(%PH%/2-Y,%PW%/2-X)/(2*PI),1))' -frames 1 -y ymap.pgm

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 ($C \rightarrow \infty$) the formula simplifies to: $X = A(1 - E/R)$

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.

2.130 Black hole simulation with remap filter

An introduction to Kerr black holes, including mathematics: <http://www.madore.org/~david/math/kerr.html>

An exact mathematical approach for the simulation of spinning black holes can be found here:
Oliver James, Eugenie von Tunzelmann, Paul Franklin, Kip S. Thorne: "Gravitational Lensing by Spinning Black Holes in Astrophysics, and in the Movie Interstellar" <https://arxiv.org/abs/1502.03808>

There are many informations about black hole simulations on Simon Tyran's website: <http://www.yukterez.net/>

A black hole simulation by Ziri Younsi:

- Falling into a black hole (Realistic Ultra HD 360 VR movie) [8K] https://www.youtube.com/watch?v=S6qw5_YA8iE

A video from Alessandro Roussel about how to make an exact simulation of a black hole (in french language):

- <https://www.youtube.com/watch?v=PjWjZFwz3rQ>

FFmpeg's remap filter can be used to simulate the light deviation near black holes.

As an approximation, when a beam of light passes near a black hole, it will be deviated by angle alpha (in Radians):

$$\alpha = 2 * rs / (r - rs) \quad \text{or} \quad \alpha = 2 / ((r / rs) - 1)$$

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.

r / rs	alpha in radians	alpha in degrees
1	infinite	infinite
1.2	10	573°
1.5	4	229°
2	2	115°
2.5	1.333	76.4°
3	1	57.3°
4	0.666	38.2°
5	0.5	28.6°
6	0.4	22.9°
8	0.286	16.4°
10	0.222	12.7°
15	0.143	8.2°
20	0.105	6.0°

Assuming we have a 180° fisheye image, we can express the light deviation in pixels: $c = \text{height} / \pi * 2 * rs / (r - rs)$

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

$r = \sqrt{(x - xc)^2 + (y - yc)^2}$

$c = \text{shape} / (r - rs)$ where shape is a constant that defines the "strength" of the distortion

if $r > rs$:

$x_{\text{remap}} = x - c * (x - xc)$

$y_{\text{remap}} = y - c * (y - yc)$

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:

```
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.

```
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
```

Calculate the xmap and ymap files for all frames. This is done by a C# program, which can be downloaded here:
http://www.astro-electronic.de/source/Moving_Black_Hole.zip

Example of a simulated black hole:



Black hole simulation with FFmpeg, no C# code required:

```
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 it can all 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)'[x];[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)'[y];[v][x]
[y]remap -frames 1 -y out.jpg

pause
```

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=1600"              :: 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='^
st(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));^
st(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));^
st(2,X-ld(0));^
st(3,hypot(ld(2),ld(1)-Y));^
st(4,X-(ld(2)*%SH%*2*%RS%/(ld(3)-%RS%)));^
if(lt(%RS%,ld(3)),clip(ld(4),0,%W%),0)'[x];^
[2]format=pix_fmts=gray16le,geq='^
st(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));^
st(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));^
st(2,Y-ld(1));^
st(3,hypot(ld(0)-X,ld(2)));^
st(4,Y-(ld(2)*%SH%*2*%RS%/(ld(3)-%RS%)));^
if(lt(%RS%,ld(3)),clip(ld(4),0,%H%),0)'[y];^
[v][x][y]remap -t 3 -y %OUT%'

pause
```

"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:

```
st(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.

2.131 Wormhole simulation

A wormhole is a hypothetical window to another place in space or time, or even in another universe.

For more informations see <https://en.wikipedia.org/wiki/Wormhole>

Short story from Rudy Rucker:

- "The Last Einstein-Rosen Bridge" http://www.rudyrucker.com/transrealbooks/completestories/#_Toc14

An easy-level book, unfortunately without any mathematics:

- Kip Thorne: "The Science of Interstellar"

The exact mathematical approach can be found in this paper. Unfortunately the mathematics exceed my capabilities by far:

- Oliver James, Eugenie von Tunzelmann, Paul Franklin, Kip S. Thorne: "Visualizing Interstellar's Wormhole" <https://arxiv.org/pdf/1502.03809.pdf>
- The same paper is also available here: <https://aapt.scitation.org/doi/10.1119/1.4916949>

Two very interesting videos by Scott Manley:

- What Would Travelling Through A Wormhole Look Like? (VR/360) <https://www.youtube.com/watch?v=V7e-1bRpweo>
- Wormholes Get Weirder - Watch Other Objects Near Wormholes & Learn How I Created The Images <https://www.youtube.com/watch?v=PVO8nzb1o2w>

The math is a little bit explained beginning at 7:40. Two pages of the code are visible at 12:40 (slow 3D version) and 12:44 (much faster 2D version). Somewhere in the comments he mentioned that he did use the "RK4" Runge-Kutta algorithm.
I got his C code and will try to understand that later.

An article from Jason Biggs, Wolfram Alpha: (I'm not familiar with this programming language)

- Visualizing Interstellar's Wormhole: from article to programming <https://community.wolfram.com/groups/-/m/t/852052>
Please note that some characters got lost in the code.
- The same code (without missing characters) is also available here: <https://mathematica.stackexchange.com/questions/110945/interstellar-image-effects>

Here are several wormhole simulations by Pierre-Jean Charpin:

- <https://www.youtube.com/channel/UC51wkO4JFG-a018jIAOh9rA/videos>

A wormhole simulation by Alessandro Roussel:

- 360° Traversing a flat Worm Hole https://www.youtube.com/watch?v=Fqm_OG4dvKs

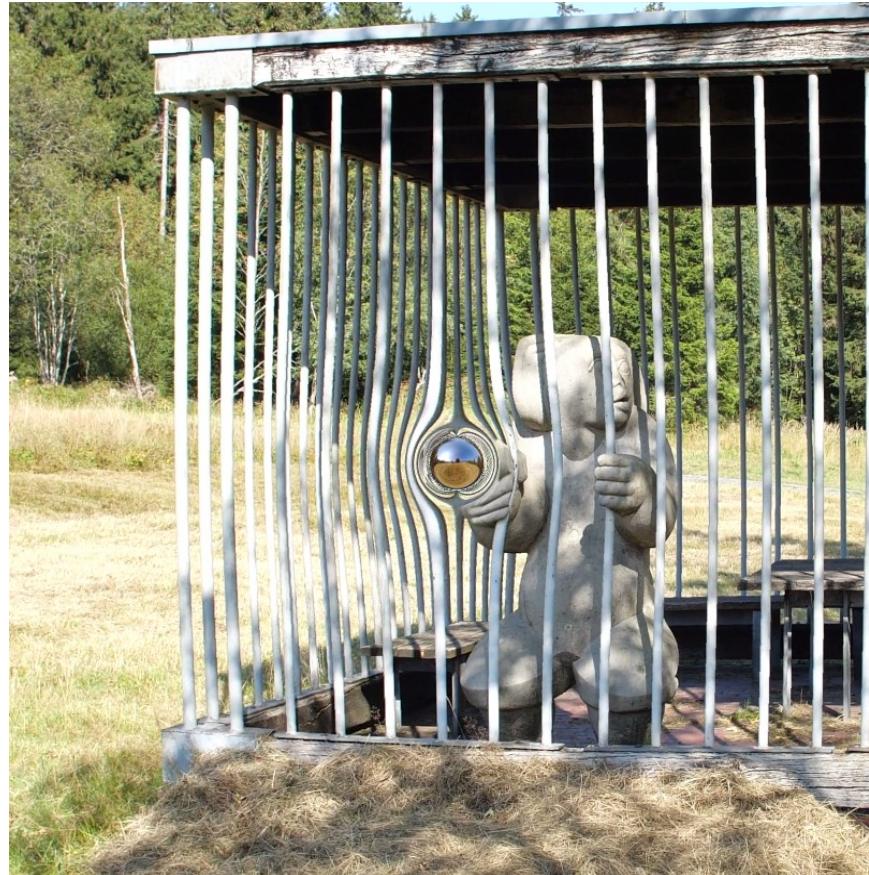
A wormhole simulation by Robert Szili:

- Walking around a wormhole <https://www.youtube.com/watch?v=4Wm3F9s8ThE>

An interactive wormhole project on Github: <https://github.com/sirxemic/wormhole>

A simplified 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).
- The drawback of this simplified simulation is that you can't let the camera or an object fly through the wormhole. You can only look at it from a distance.



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 %LP% -i xmap.pgm -i ymap.pgm -lavfi "v360=output=e:pitch=%P%:yaw=%Y%:roll=%R%,format=pix_fmts=rgb24,remap"
-q:v 2 -t %T% -y lp.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][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

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

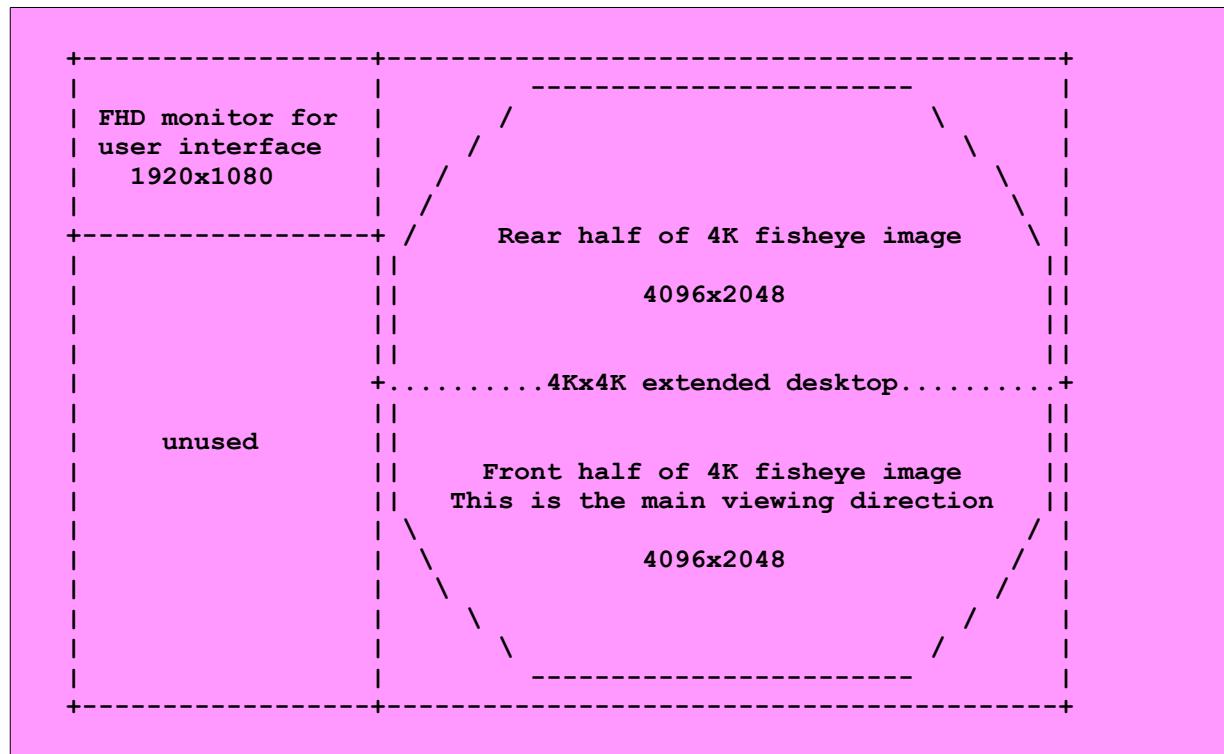
ffmpeg -i %IN% -i xmap0000.pgm -i ymap0000.pgm -i %LP% -filter_complex "[0][1][2]remap[4];[3]v360=output=ball:pitch=%P%:yaw=%Y%:roll=%R%:alpha_mask=1,scale=%LPD%:scale=%LPD%[5];[4][5]overlay=x=%LPX%-%LPD%/2:y=%LPY%-%LPD%/2" -q:v 2 -t %T% -y
out.mp4

pause

```

2.132 Realtime Wormhole in Planetarium (Hackathon 2022)

Desktop layout of the planetarium computer:

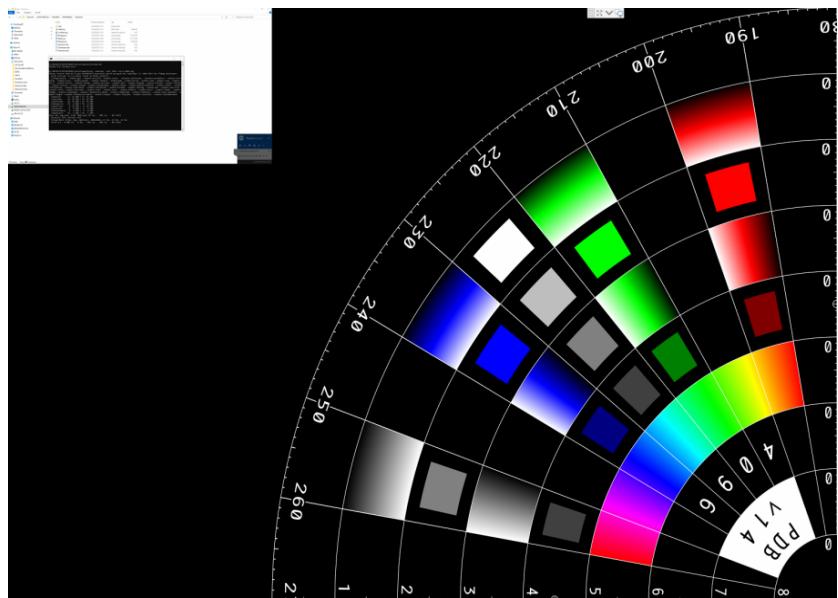


Note: Alternatively it's also possible that the extended desktop is placed on the left side and the user interface on the right side.

First test with a 4096x4096 image:

```
rem Image from http://www.paulbourke.net/dome/testpattern/4096.png  
ffplay -noborder -x 4096 -y 4096 -left 1920 -top 0 4096.png  
pause
```

This is a screenshot of the whole desktop. The white rectangle in the top left corner is the 1920x1080 monitor with the user interface. Obviously the 4096x4096 image is shown twice as large as it should be.



The problem was that in Settings --> Display the scale was set to 200%. With the correct setting (100%) it works as expected.

Example for sending the stream:

```
rem Test pattern from http://www.paulbourke.net/dome/testpattern/4096.png

set "ROT=240"      :: Seconds per full revolution
set "FR=4"         :: Framerate
set "BR=20M"        :: Bitrate

ffmpeg -f lavfi -i sine=frequency=1000:sample_rate=48000 -framerate %FR% -loop 1 -i 4096.png -vf
scale=4096:4096,rotate='2*PI*t/%ROT%' -strict -1 -g 1 -b:v %BR% -f mpegts "udp://192.168.178.45:1234"

pause
```

Note: "-strict -1" is required because multiples of 4096 seem to be forbidden for image size.

Note: "-g 1" means all frames are encoded as keyframes. This is useful for faster startup at the receiver side, but it reduces the framerate.

Note: In this case "-framerate" works much better than "-r".

Example for receiving the stream with FFmpeg (drawback: no audio!):

```
ffmpeg -i "udp://192.168.178.45:1234" -vf scale=960:960 -window_borderless 1 -window_x 0 -window_y 0 -f sdl2 -

pause
```

Example for receiving the stream with FFplay (with audio):

```
ffplay -noborder -x 4096 -y 4096 -left 1920 -top 0 udp://192.168.178.45:1234

pause
```

Note: In both cases, use the IP addresses from the other computer. Check with ipconfig.

See also <https://trac.ffmpeg.org/wiki/StreamingGuide>

See also <https://www.heimnetzwerke.net/netzwerkverbindung-zwischen-zwei-computern-herstellen/>

See also: "How to Increase NDI Bandwidth for High-res Images and Framerate"

<https://www.fddb.org/news/how-to-increase-ndi-bandwidth-for-high-res-images-and-framerate/>

<https://nestimmersion.ca/nestndi.php>

Final results after testing in the planetarium:

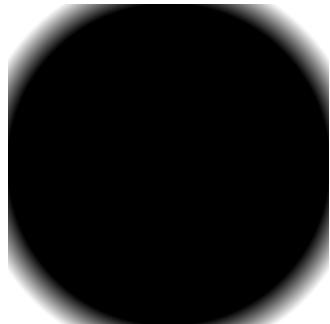
- For fulldome projection it's essential to avoid large bright areas in the image, because light is reflected from one side of the dome to the other side, where it reduces contrast. I knew this in advance and did attenuate the overhead sky. But It would have been even better to attenuate the whole northern sky, because the audience was looking to the south anyway.
- The transmission of the stream over network turned out to be problematic. I did test the transmission from computer A to computer B in advance and it worked well, after I set permissions in the Windows firewall settings in computer B. But that doesn't mean the transmission does also work from the same computer A to computer C (which is the computer in the planetarium). In this configuration the firewall and the antivirus software had to be deactivated in computer A. I don't understand why.
- The transmission of the audio track (together with the video stream in the same Mpeg transport stream) turned out to be very problematic. Audio was always received with many short gaps, even when the video resolution was set to very low 1200x1200 pixels. We finally disabled the audio stream (with option -an in the last FFmpeg command) and sent the audio track from another computer to the planetarium. Synchronization wasn't critical because there are only bird voices in the audio track. But that meant we could not use the sound effect when the wormhole appears. As a workaround, luckily we found a piano of the stage, which we used to generate a live sound effect when the wormhole appeared.
- Without the audio stream, it was possible to send the video stream with 2048x2048 resolution with an acceptable framerate. We did not have the time to check if 2880x2880 (which is the resolution of the background video from the Kodak PixPro camera) would have worked as well.
- If the 360° camera is also located in the planetarium (where it's quite dark), a lamp is required to illuminate the people who are looking into the camera. It's essential not to direct the lamp towards the audience. People are seen best if they are very close (about 5-10cm) to the camera.
- There was about 1 second delay in the video stream from the 360° camera to the dome.

How does the algorithm work?

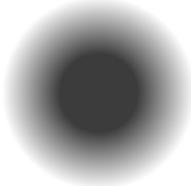
The background video was recorded with a Kodak PixPro SP360 4K camera with resolution 2880x2880 @ 29.97fps:



This video is scaled down to 2048x2048. Then mask1 is applied to fill the corners with a dark green color:



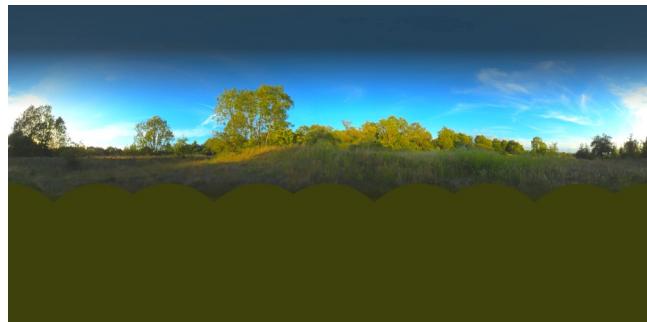
Then mask2 is applied to attenuate the overhead sky. This improves the contrast in fulldome projection:



Then the video is rotated by angle alpha, which is defined by a sine function with 25° amplitude and period 50 seconds.

The resulting video is transformed with the v360 filter from fisheye to equirectangular. The ground is no-data area and filled with dark green color. Why is it necessary to give the ground a color, if it's not visible in the final fulldome projection? That's because some light rays which pass near the wormhole are bent towards the ground. The ground becomes in fact visible in some small areas around the wormhole.

Saturation and gamma is corrected. It's always a good idea to use strong saturation for a planetarium projection:



A sound file is added (bird sounds), and a short sound effect is applied at two times when the wormhole appears and disappears.

This video is saved with resolution 4096x2048 @ 12fps as background.mov.

Now comes the realtime part.

A live video is captured from a Ricoh Theta camera with FHD resolution, using the live streaming driver. This stream is converted to a mirrorsphere with resolution 200x200 @ 12fps.

This stream is stacked vertically under the background.mov video. The resulting stream is 4096 pixels wide and $2048+200=2248$ pixels high.

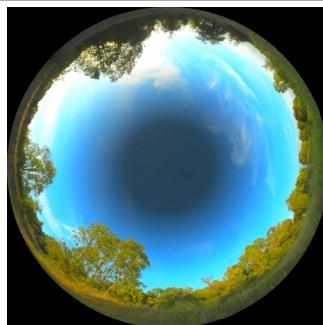
The reason why both streams are stacked together is that the pixels from both streams must be combined for the resulting wormhole video, and the remap filter accepts only one input.

The xmap and ymap files for the remap filter come from two streamselect filters with two inputs each. That's because the wormhole is switched on at time t1 and switched off at time t2. One set of mapping files is for the wormhole simulation (in this case both input streams are used), and the other set of mapping files is without wormhole (in this case the second input stream isn't used). For the calculation of the mapping files, see the next chapter. The output of the remap filter has resolution 2048x2048.

Now the video is rotated by angle -alpha.

Why is the background video rotated by angle alpha and the final stream (with wormhole) by angle -alpha? That's because the wormhole simulation uses static mapping files (the mapping files do not change over time). Dynamic mapping files would be too large to load in realtime. With this technique of rotating and de-rotating the background is stationary and the wormhole does move.

The resulting stream is either saved a *.mov file, or is piped to FFplay, or is streamed to a UDP address:



Without wormhole



With wormhole

This is the batch file:

```
set "IN=133_0006.mp4"    :: Background video
set "SS=20"                :: In point in seconds
set "LENGTH=180"            :: Length in seconds
set "T1=15"                 :: Time where wormhole appears
set "T2=170"                :: Time where wormhole vanishes
set "T3=14"                 :: Time for first sound effect
set "T4=169"                :: Time for second sound effect
set "SIZE=2048"              :: Main size
set "SIZE2=200"              :: Secondary size (this is content of the wormhole)

rem Parameters for wormhole transformation:

set "IN_W=(2*%SIZE%)"      :: Main equirectangular input width in pixels
set "IN_H=%SIZE%"           :: Main equirectangular input height in pixels
set "IN_W2=%SIZE2%"          :: Secondary input size (this is content of the wormhole)
set "OUT_W=%SIZE%"           :: Output width in pixels
```

```

set "OUT_H=%SIZE%"          :: Output height in pixels
set "OUT_H_FOV=180"         :: Output horizontal field of view in degrees
set "OUT_V_FOV=180"         :: Output vertical field of view in degrees
set "RS=8"                  :: Schwarzschild radius in degrees
set "S=0.06"                :: Make the distortion smaller, 1 = realistic
set "PITCH1=18"              :: Pitch rotation in degrees before applying wormhole
set "PITCH2=72"              :: Pitch rotation in degrees after applying wormhole

rem Parameters for filling the missing ground area:

set "R3=100"                :: Fill ground only outside of this diameter (percent of image width)
set "RAMP=10"                 :: Width of the transition band (percent of image width)
set "GROUND=0x181800"         :: Color for filling the ground

rem Parameters for attenuating the overhead sky:

set "R1=12"                  :: Inner radius in pixels (percent of image width)
set "R2=30"                  :: Outer radius in pixels (percent of image width)
set "DARK=60"                 :: Brightness level inside of R1, 0 is black, 255 is original brightness
set "BRIGHT=255"              :: Brightness level outside of R2, 0 is black, 255 is original brightness

rem Parameters for making the background video:

set "FOV=200"                :: Field of view of the input video (from Kodak Pixpro SP360 4K)
set "GAMMA=1.5"               :: Gamma correction
set "SAT=2"                   :: Saturation
set "ROT_S=-150"              :: Rotation angle at start
set "ROT_A=25"                 :: Amplitude of sine oscillation for rotation
set "ROT_T=50"                 :: Period of sine oscillation in seconds
set "VOL=3"                   :: Volume for sound effect
set "FR=8"                     :: Framerate
set "BR=40M"                  :: Bitrate

rem Create the xmap and ymap files with wormhole

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^
st(0,%OUT_H_FOV%/180*((2*X+1)/%OUT_W%-1));^
st(1,%OUT_V_FOV%/180*((2*Y+1)/%OUT_H%-1));^
st(2,atan2(ld(1),ld(0)));^

```

```

st(3,PI/2*hypot(ld(0),ld(1)));^
st(4,sin(ld(3))*cos(ld(2)));^
st(5,sin(ld(3))*sin(ld(2)));^
st(6,cos(ld(3)));^
st(1,if(lt(ld(6),cos(%OUT_H_FOV%/360*PI)),32767,0));^
st(0,ld(5)*cos(%PITCH2%/180*PI)-ld(6)*sin(%PITCH2%/180*PI));^
st(6,ld(6)*cos(%PITCH2%/180*PI)+ld(5)*sin(%PITCH2%/180*PI));^
st(5,ld(0));^
st(7,atan2(ld(5),ld(4)));^
st(8,acos(ld(6)));^
st(9,if(lte(ld(8),%RS%/180*PI),1));^
if(ld(9),st(7,ld(8)*cos(ld(7))));^
ifnot(ld(9),st(8,ld(8)-%S%*(2*%RS%/180*PI/(ld(8)-%RS%/180*PI))));^
ifnot(ld(9),st(4,sin(ld(8))*cos(ld(7))));^
ifnot(ld(9),st(5,sin(ld(8))*sin(ld(7))));^
ifnot(ld(9),st(6,cos(ld(8))));^
ifnot(ld(9),st(0,ld(5)*cos(%PITCH1%/180*PI)-ld(6)*sin(%PITCH1%/180*PI))));^
ifnot(ld(9),st(6,ld(6)*cos(%PITCH1%/180*PI)+ld(5)*sin(%PITCH1%/180*PI))));^
ifnot(ld(9),st(5,ld(0)));^
ifnot(ld(9),st(7,atan2(ld(5),ld(4))));^
ifnot(ld(9),st(8,acos(ld(6))));^
ifnot(ld(9),st(7,atan2(sin(ld(8))*cos(ld(7)),cos(ld(8)))));^
if(ld(9),0.5*%IN_W%*(1+180/PI/%RS%*ld(7)),ld(1)+0.5*%IN_W%*(1+ld(7)/PI))' -frames 1 -y xmap.pgm

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^
st(0,%OUT_H_FOV%/180*((2*X+1)/%OUT_W%-1));^
st(1,%OUT_V_FOV%/180*((2*Y+1)/%OUT_H%-1));^
st(2,atan2(ld(1),ld(0)));^
st(3,PI/2*hypot(ld(0),ld(1)));^
st(4,sin(ld(3))*cos(ld(2)));^
st(5,sin(ld(3))*sin(ld(2)));^
st(6,cos(ld(3)));^
st(1,if(lt(ld(6),cos(%OUT_H_FOV%/360*PI)),32767,0));^
st(0,ld(5)*cos(%PITCH2%/180*PI)-ld(6)*sin(%PITCH2%/180*PI));^
st(6,ld(6)*cos(%PITCH2%/180*PI)+ld(5)*sin(%PITCH2%/180*PI));^
st(5,ld(0));^
st(7,atan2(ld(5),ld(4)));^
st(8,acos(ld(6)));^
st(9,if(lte(ld(8),%RS%/180*PI),1));^
if(ld(9),st(8,ld(8)*sin(ld(7))));^

```

```

ifnot(ld(9),st(8,ld(8)-%S%*(2*%RS%/180*PI/(ld(8)-%RS%/180*PI)))) ;^
ifnot(ld(9),st(4,sin(ld(8))*cos(ld(7))));^
ifnot(ld(9),st(5,sin(ld(8))*sin(ld(7))));^
ifnot(ld(9),st(6,cos(ld(8))));^
ifnot(ld(9),st(0,ld(5)*cos(%PITCH1%/180*PI)-ld(6)*sin(%PITCH1%/180*PI)));^
ifnot(ld(9),st(6,ld(6)*cos(%PITCH1%/180*PI)+ld(5)*sin(%PITCH1%/180*PI)));^
ifnot(ld(9),st(5,ld(0)));^
ifnot(ld(9),st(7,atan2(ld(5),ld(4))));^
ifnot(ld(9),st(8,acos(ld(6))));^
ifnot(ld(9),st(8,asin(sin(ld(8))*sin(ld(7))));^
if(ld(9),%IN_H%+0.5*%IN_W%*(1+180/PI/%RS%*ld(8)),ld(1)+0.5*%IN_H%*(1+ld(8)*2/PI))' -frames 1 -y ymap.pgm

rem Create the xmap and ymap files without wormhole

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^
st(0,%OUT_H_FOV%/180*((2*x+1)/%OUT_W%-1));^
st(1,%OUT_V_FOV%/180*((2*y+1)/%OUT_H%-1));^
st(2,atan2(ld(1),ld(0)));^
st(3,PI/2*hypot(ld(0),ld(1)));^
st(4,sin(ld(3))*cos(ld(2)));^
st(5,sin(ld(3))*sin(ld(2)));^
st(6,cos(ld(3)));^
st(1,if(lt(ld(6),cos(%OUT_H_FOV%/360*PI)),32767,0));^
st(0,-ld(6));^
st(6,ld(5));^
st(5,ld(0));^
st(7,atan2(ld(5),ld(4)));^
st(8,acos(ld(6)));^
st(7,atan2(sin(ld(8))*cos(ld(7)),cos(ld(8))));^
ld(1)+0.5*%IN_W%*(1+ld(7)/PI))' -frames 1 -y xmap2.pgm

ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^
st(0,%OUT_H_FOV%/180*((2*x+1)/%OUT_W%-1));^
st(1,%OUT_V_FOV%/180*((2*y+1)/%OUT_H%-1));^
st(2,atan2(ld(1),ld(0)));^
st(3,PI/2*hypot(ld(0),ld(1)));^
st(4,sin(ld(3))*cos(ld(2)));^
st(5,sin(ld(3))*sin(ld(2)));^
st(6,cos(ld(3)));^

```

```

st(1,if(lt(ld(6),cos(%OUT_H_FOV%/360*PI)),32767,0));^
st(0,-ld(6));^
st(6,ld(5));^
st(5,ld(0));^
st(7,atan2(ld(5),ld(4)));^
st(8,acos(ld(6)));^
st(8,asin(sin(ld(8))*sin(ld(7))));^
ld(1)+0.5*%IN_H%*(1+ld(8)*2/PI)' -frames 1 -y ymap2.pgm

rem Create the mask for filling the ground (mask1.png)
ffmpeg -f lavfi -i nullsrc=size=%SIZE%x%SIZE% -lavfi format=gray8,geq='clip(256/(%RAMP%*W/100)*(hypot(X-%SIZE%/2,Y-%SIZE%/2)-(%R3%*W/200)),0,255)' -frames 1 -y mask1.png

rem Create the mask for attenuating the overhead sky (mask2.png)
ffmpeg -f lavfi -i nullsrc=size=%SIZE%x%SIZE% -lavfi format=gray8,geq='clip(lerp(%DARK%,%BRIGHT%,(hypot(X-%SIZE%/2,Y-%SIZE%/2)-(%R1%*W/100))/((%R2%-%R1%)*W/100)),%DARK%,%BRIGHT%)',format=rgb24 -frames 1 -y mask2.png

rem Make an equirectangular background image (only for testing)
ffmpeg -ss %SS% -i %IN% -f lavfi -i color=%GROUND%:size=%SIZE%*%SIZE% -i mask1.png -i mask2.png -lavfi [0]scale=%SIZE%x%SIZE%,format=gbrp[fg];[2]format=gbrp[mask];[fg][1][mask]maskedmerge[a];[a][3]multiply=offset=0,rotate=(%ROT_S%+%ROT_A%*sin(2*PI/%ROT_T%*t))/180*PI:c=%GROUND%,drawbox=w=1:h=1:color=%GROUND%,v360=fisheye:e:ih_fov=%FOV%:iv_fov=%FOV%:pitch=-90:w=2*%SIZE%:h=%SIZE%,scroll=hpos=0.1,eq=saturation=%SAT%:gamma=%GAMMA% -frames 1 -y background.png

rem Make the equirectangular background video
rem Add the sound track and two sound effects

ffmpeg -ss %SS% -i %IN% -f lavfi -i color=%GROUND%:size=%SIZE%*%SIZE% -i mask1.png -i mask2.png -i birds4min.mp3 -i
soundeffect.mp3 -i soundeffect.mp3 -lavfi [0]scale=%SIZE%x%SIZE%,format=gbrp[fg];[2]format=gbrp[mask];[fg][1]
[mask]maskedmerge[a];[a][3]multiply=offset=0,rotate=(%ROT_S%+%ROT_A%*sin(2*PI/%ROT_T%*t))/180*PI:c=%GROUND%
%,drawbox=w=1:h=1:color=%GROUND%,v360=fisheye:e:ih_fov=%FOV%:iv_fov=%FOV%:pitch=-90:w=2*%SIZE%:h=%SIZE%,eq=saturation=%SAT%:gamma=%GAMMA%;[5]volume=%VOL%,adelay=%T3%s:all=1,apad[a];[6]volume=%VOL%,adelay=%T4%s:all=1,apad[b];[4][a]
[b]amix=3:normalize=0 -r %FR% -b:v %BR% -t %LENGTH% -y background.mov

```

```

rem ****
rem Below is the realtime part
rem *****

rem Vertically stack the input videos together, the second one may be smaller
rem Apply the wormhole effect, switch wormhole on/off, de-rotate

rem a) With output to a file:

rem ffmpeg -i background.mov -f dshow -rtbufsize 200M -i video="RICOH THETA V/Z1 FullHD" -i xmap.pgm -i xmap2.pgm -i
ympg.pgm -i ymap2.pgm -lavfi [0]sendcmd="%T1% streamselect@1 map 1",sendcmd="%T1% streamselect@2 map 1",sendcmd="%T2%
streamselect@1 map 0",sendcmd="%T2% streamselect@2 map 0"[a];[2]loop=-1:1[2a];[3]loop=-1:1[3a];[4]loop=-1:1[4a];
[5]loop=-1:1[5a];[3a][2a]streamselect@1=map=0[x];[5a][4a]streamselect@2=map=0[y];[1]fps=%FR%,v360=e:ball:w=%SIZE2%:h=
%SIZE2%[c];[a][c]xstack=2:"0_0|0_h0"[c];[c][x][y]remap=fill=black,rotate=-(%ROT_A%*sin(2*PI/%ROT_T%*t))/180*PI -r %FR%
-b:v %BR% -t %LENGTH% -y out.mov

rem b) With output to FFplay:

rem ffmpeg -i background.mov -f dshow -rtbufsize 200M -i video="RICOH THETA V/Z1 FullHD" -i xmap.pgm -i xmap2.pgm -i
ympg.pgm -i ymap2.pgm -lavfi [0]sendcmd="%T1% streamselect@1 map 1",sendcmd="%T1% streamselect@2 map 1",sendcmd="%T2%
streamselect@1 map 0",sendcmd="%T2% streamselect@2 map 0"[a];[2]loop=-1:1[2a];[3]loop=-1:1[3a];[4]loop=-1:1[4a];
[5]loop=-1:1[5a];[3a][2a]streamselect@1=map=0[x];[5a][4a]streamselect@2=map=0[y];[1]fps=%FR%,v360=e:ball:w=%SIZE2%:h=
%SIZE2%[c];[a][c]xstack=2:"0_0|0_h0"[c];[c][x][y]remap=fill=black,rotate=-(%ROT_A%*sin(2*PI/%ROT_T%*t))/180*PI -r %FR%
-b:v %BR% -t %LENGTH% -f nut - | ffplay -fs -autoexit -

rem c) With output to UDP stream:

ffmpeg -i background.mov -f dshow -rtbufsize 200M -i video="RICOH THETA V/Z1 FullHD" -i xmap.pgm -i xmap2.pgm -i
ympg.pgm -i ymap2.pgm -lavfi [0]sendcmd="%T1% streamselect@1 map 1",sendcmd="%T1% streamselect@2 map 1",sendcmd="%T2%
streamselect@1 map 0",sendcmd="%T2% streamselect@2 map 0"[a];[2]loop=-1:1[2a];[3]loop=-1:1[3a];[4]loop=-1:1[4a];
[5]loop=-1:1[5a];[3a][2a]streamselect@1=map=0[x];[5a][4a]streamselect@2=map=0[y];[1]fps=%FR%,v360=e:ball:w=%SIZE2%:h=
%SIZE2%[c];[a][c]xstack=2:"0_0|0_h0"[c];[c][x][y]remap=fill=black,rotate=-(%ROT_A%*sin(2*PI/%ROT_T%*t))/180*PI -r %FR%
-b:v %BR% -t %LENGTH% -f mpegtts "udp://169.254.23.82:1234"

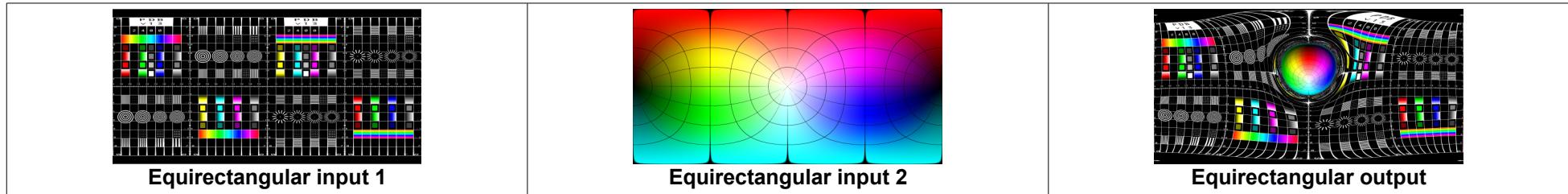
pause

```

2.133 Spherical Transformations with two Inputs

Two input images (or videos) are vertically stacked together, and a remap filter is applied to the resulting frame. Each pixel in the output frame can come either from input 1 or from input 2.

This can be used for wormhole simulation, where two 360° worlds are combined into one output image:



It can also be used for overlaying a (rotating) planet in front of a background video.

Comments	For which input / output format is it?	Command line in batch file
Specify the parameters for the spherical transformation. Of course, you only have to specify those parameters that are actually used.	For all formats	<pre> set "IN_W=2400" :: Main input width in pixels set "IN_H=1200" :: Main input height in pixels set "IN_W2=2400" :: Secondary input width (inside the wormhole) set "IN_H2=1200" :: Secondary input height (inside the wormhole) set "OUT_W=1800" :: Output width in pixels set "OUT_H=900" :: Output height in pixels set "OUT_H_FOV=360" :: Output horizontal field of view in degrees set "OUT_V_FOV=180" :: Output vertical field of view in degrees set "OUT2_H_FOV=180" :: Output2 horizontal field of view in degrees set "OUT2_V_FOV=180" :: Output2 vertical field of view in degrees set "RS=30" :: Schwarzschild radius in degrees set "DIST=2" :: Distance normalized to planet radius set "S=0.1" :: Make the distortion smaller, 1 = realistic set "YAW=0" :: Output yaw rotation in degrees set "PITCH=-30" :: Output pitch rotation in degrees set "ROLL=0" :: Output roll rotation in degrees set "YAW1=0" :: Input 1 yaw rotation in degrees set "PITCH1=30" :: Input 1 pitch rotation in degrees set "ROLL1=0" :: Input 1 roll rotation in degrees set "YAW2=0" :: Input 2 yaw rotation in degrees set "PITCH2=0" :: Input 2 pitch rotation in degrees set "ROLL2=20" :: Input 2 roll rotation in degrees </pre>
The size of the xmap and ymap files is the size of the output image. The size of the two input images may be different.	For all formats	<pre> rem Create the xmap and ymap files ffmpeg -f lavfi -i nullsrc=size=%OUT_W%x%OUT_H% -vf format=pix_fmts=gray16le,geq='^ </pre>
Convert from output pixel coordinates to X,Y,Z space, depending on the desired output format. The X,Y,Z output coordinates are saved	For fisheye output	<pre> st(0,%OUT_H_FOV%/180*((2*x+1)/%OUT_W%-1));^ st(1,%OUT_V_FOV%/180*((2*y+1)/%OUT_H%-1));^ st(2,atan2(ld(1),ld(0)));^ st(3,PI/2*hypot(ld(0),ld(1)));^ st(4,sin(ld(3))*cos(ld(2)));^ st(5,sin(ld(3))*sin(ld(2)));^ st(6,cos(ld(3)));^ st(1,if(lt(ld(6),cos(%OUT_H_FOV%/360*PI)),32767,0));^ </pre>

in variables 4,5,6. Note: If the pixel is unmapped, variable (1) is set to 32767 which means out of range.	For equirectangular output	<pre>st(0,PI/360*%OUT_H_FOV%*((2*X+1)/%OUT_W%-1));^ st(1,PI/360*%OUT_V_FOV%*((2*Y+1)/%OUT_H%-1));^ st(4,cos(ld(1))*sin(ld(0)));^ st(5,sin(ld(1)));^ st(6,cos(ld(1))*cos(ld(0)));^</pre>
Optional output roll rotation (around Z axis), input and output are in variables 4,5,6.	For all formats	<pre>st(0,ld(4)*cos(%ROLL%/180*PI)-ld(5)*sin(%ROLL%/180*PI));^ st(5,ld(5)*cos(%ROLL%/180*PI)+ld(4)*sin(%ROLL%/180*PI));^ st(4,ld(0));^</pre>
Optional output pitch rotation (around X axis), input and output are in variables 4,5,6.	For all formats	<pre>st(0,ld(5)*cos(%PITCH%/180*PI)-ld(6)*sin(%PITCH%/180*PI));^ st(6,ld(6)*cos(%PITCH%/180*PI)+ld(5)*sin(%PITCH%/180*PI));^ st(5,ld(0));^</pre>
Optional output yaw rotation (around Y axis), input and output are in variables 4,5,6.	For all formats	<pre>st(0,ld(4)*cos(%YAW%/180*PI)+ld(6)*sin(%YAW%/180*PI));^ st(6,ld(6)*cos(%YAW%/180*PI)-ld(4)*sin(%YAW%/180*PI));^ st(4,ld(0));^</pre>
Decide from which of the two inputs this pixel comes, and set variable 9 accordingly: 0 = from input 1 1 = from input 2	Based on Schwarzschild radius (in degrees) Based on distance, normalized to planet radius	<pre>st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^ st(9,lte(ld(8),%RS%/180*PI));^</pre> <pre>st(7,atan2(ld(5),ld(4)));^ st(8,acos(ld(6)));^ st(9,lte(ld(8),asin(1/%DIST%)));^</pre>
Optional black hole distortion for input 1, %RS% is the Schwarzschild radius in degrees and %S% is a factor to make the distortion smaller. For realistic simulation use S = 1.	For all formats	<p>This block can be omitted, if no distortion is required.</p> <pre>ifnot(ld(9),st(8,ld(8)-%S%*(2*%RS%/180*PI/(ld(8)-%RS%/180*PI))));^ ifnot(ld(9),st(4,sin(ld(8))*cos(ld(7))));^ ifnot(ld(9),st(5,sin(ld(8))*sin(ld(7))));^ ifnot(ld(9),st(6,cos(ld(8))));^</pre>

Optional input 1 roll rotation (around Z axis), input and output are in variables 4,5,6.	For all formats	<pre>ifnot(ld(9),st(0,ld(4)*cos(%ROLL1%/180*PI)-ld(5)*sin(%ROLL1%/180*PI)));^ ifnot(ld(9),st(5,ld(5)*cos(%ROLL1%/180*PI)+ld(4)*sin(%ROLL1%/180*PI)));^ ifnot(ld(9),st(4,ld(0)));^</pre>
Optional input 1 pitch rotation (around X axis), input and output are in variables 4,5,6.	For all formats	<pre>ifnot(ld(9),st(0,ld(5)*cos(%PITCH1%/180*PI)-ld(6)*sin(%PITCH1%/180*PI)));^ ifnot(ld(9),st(6,ld(6)*cos(%PITCH1%/180*PI)+ld(5)*sin(%PITCH1%/180*PI)));^ ifnot(ld(9),st(5,ld(0)));^</pre>
Optional input 1 yaw rotation (around Y axis), input and output are in variables 4,5,6.	For all formats	<pre>ifnot(ld(9),st(0,ld(4)*cos(%YAW1%/180*PI)+ld(6)*sin(%YAW1%/180*PI)));^ ifnot(ld(9),st(6,ld(6)*cos(%YAW1%/180*PI)-ld(4)*sin(%YAW1%/180*PI)));^ ifnot(ld(9),st(4,ld(0)));^</pre>
Convert from X,Y,Z space to the desired input 1 format.	For equirectangular input 1	<pre>ifnot(ld(9),st(7,atan2(ld(5),ld(4))));^ ifnot(ld(9),st(8,acos(ld(6))));^ Only for xmap file: ifnot(ld(9),st(7,atan2(sin(ld(8))*cos(ld(7)),cos(ld(8))/PI)));^ ifnot(ld(9),st(7,ld(1)+0.5*%IN_W%*(1+ld(7))));^ Only for ymap file: ifnot(ld(9),st(8,2/PI*asin(sin(ld(8))*sin(ld(7))));^ ifnot(ld(9),st(8,ld(1)+0.5*%IN_H%*(1+ld(8))));^</pre>
Transform output 2 coordinates to [-1...+1] range, the result is in variables 5 and 6	Based on Schwarzschild radius (in degrees)	<pre>if(ld(9),st(5,180/PI/%RS%*ld(8)*cos(ld(7))));^ if(ld(9),st(6,180/PI/%RS%*ld(8)*sin(ld(7))));^</pre>
	Based on distance, normalized to planet radius	<pre>if(ld(9),st(5,ld(8)/asin(1/%DIST%)*cos(ld(7))));^ if(ld(9),st(6,ld(8)/asin(1/%DIST%)*sin(ld(7))));^</pre>
Convert from [-1...+1] range to X,Y,Z space, depending on the desired output format.	For fisheye output 2	<pre>if(ld(9),st(5,ld(5)*%OUT2_H_FOV%/180));^ if(ld(9),st(6,ld(6)*%OUT2_V_FOV%/180));^ if(ld(9),st(7,atan2(ld(6),ld(5))));^ if(ld(9),st(8,PI/2*hypot(ld(5),ld(6))));^ if(ld(9),st(4,sin(ld(8))*cos(ld(7))));^ if(ld(9),st(5,sin(ld(8))*sin(ld(7))));^ if(ld(9),st(6,cos(ld(8))));^</pre>
The X,Y,Z output		

coordinates are saved in variables 4,5,6.	For mirrorsphere output 2 (as seen from infinite distance)	<pre>if(ld(9),st(7,atan2(ld(5),ld(6)))) ;^ if(ld(9),st(8,2*asin(clip(hypot(ld(5),ld(6)),0,1)))) ;^ if(ld(9),st(4,sin(ld(8))*sin(ld(7)))) ;^ if(ld(9),st(5,sin(ld(8))*cos(ld(7)))) ;^ if(ld(9),st(6,cos(ld(8)))) ;^</pre>
	For planet output 2 (as seen from infinite distance)	<pre>if(ld(9),st(4,ld(5))) ;^ if(ld(9),st(5,ld(6))) ;^ if(ld(9),st(6,sqrt(1-ld(4)*ld(4)-ld(5)*ld(5)))) ;^</pre>
	For planet output 2 (as seen from the finite distance %DIST%) Less than half of the sphere is visible.	<pre>if(ld(9),st(2,hypot(ld(5),ld(6)))) ;^ if(ld(9),st(2,ld(2)*tan(asin(1/%DIST%)))) ;^ if(ld(9),st(3,1+ld(2)*ld(2)*(1-%DIST%*%DIST%))) ;^ if(ld(9),st(8,2*atan((1-sqrt(abs(ld(3))))/((1+%DIST%)*ld(2))))) ;^ if(ld(9),st(7,atan2(ld(5),ld(6)))) ;^ if(ld(9),st(4,sin(ld(8))*sin(ld(7)))) ;^ if(ld(9),st(5,sin(ld(8))*cos(ld(7)))) ;^ if(ld(9),st(6,cos(ld(8)))) ;^</pre>
Optional input 2 roll rotation (around Z axis), input and output are in variables 4,5,6.	For all formats	<pre>if(ld(9),st(0,ld(4)*cos(%ROLL2%/180*PI)-ld(5)*sin(%ROLL2%/180*PI))) ;^ if(ld(9),st(5,ld(5)*cos(%ROLL2%/180*PI)+ld(4)*sin(%ROLL2%/180*PI))) ;^ if(ld(9),st(4,ld(0))) ;^</pre>
Optional input 2 pitch rotation (around X axis), input and output are in variables 4,5,6.	For all formats	<pre>if(ld(9),st(0,ld(5)*cos(%PITCH2%/180*PI)-ld(6)*sin(%PITCH2%/180*PI))) ;^ if(ld(9),st(6,ld(6)*cos(%PITCH2%/180*PI)+ld(5)*sin(%PITCH2%/180*PI))) ;^ if(ld(9),st(5,ld(0))) ;^</pre>
Optional input 2 yaw rotation (around Y axis), input and output are in variables 4,5,6.	For all formats	<pre>if(ld(9),st(0,ld(4)*cos(%YAW2%/180*PI)+ld(6)*sin(%YAW2%/180*PI))) ;^ if(ld(9),st(6,ld(6)*cos(%YAW2%/180*PI)-ld(4)*sin(%YAW2%/180*PI))) ;^ if(ld(9),st(4,ld(0))) ;^</pre>

Convert from X,Y,Z space to the desired input 2 format.	For equirectangular input 2	<pre> if(ld(9),st(7,atan2(ld(5),ld(4)))) ;^ if(ld(9),st(8,acos(ld(6)))) ;^ Only for xmap file: if(ld(9),st(7,atan2(sin(ld(8))*cos(ld(7)),cos(ld(8))/PI))) ;^ if(ld(9),st(7,0.5*%IN_W2%*(1+ld(7)))) ;^ Only for ymap file: if(ld(9),st(8,2/PI*asin(sin(ld(8))*sin(ld(7))))) ;^ if(ld(9),st(8,%IN_H%+0.5*%IN_H2%*(1+ld(8)))) ;^ </pre>
Finish writing the xmap and ymap files	For all formats	<pre> Only for xmap file: ld(7) ' -frames 1 -y xmap.pgm Only for ymap file: ld(8) ' -frames 1 -y ymap.pgm </pre>
Stack the two images together and apply the remap filter. Note: "-f image2" is only required in rare cases, if the *.pgm file isn't auto-detected correctly as an image.	For all formats	<pre> ffmpeg -i in1.png -i in2.png -f image2 -i xmap.pgm -f image2 -i ymap.pgm -lavfi [0] [1]xstack=2:"0_0 0_h0"[c];[c][2][3]remap -y out.png </pre>

2.134 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%

pause
```

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_object=obj%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 1> 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, which can be downloaded here:

http://www.astro-electronic.de/source/Moving_Black_Hole.zip

Step 5:

The file positions.cmd was also automatically created by the C# program (many lines omitted here):

```
0.00 overlay x 1665.00, overlay y 454.00, scale w 100.00, scale h 100.00;
0.04 overlay x 1665.00, overlay y 454.00, scale w 100.00, scale h 100.00;
0.08 overlay x 1665.00, overlay y 454.00, scale w 100.00, scale h 100.00;
...
6.00 overlay x 939.00, overlay y 449.00, scale w 100.00, scale h 100.00;
6.04 overlay x 935.20, overlay y 448.20, scale w 101.60, scale h 101.60;
6.08 overlay x 932.40, overlay y 447.40, scale w 103.20, scale h 103.20;
6.12 overlay x 928.60, overlay y 446.60, scale w 104.80, scale h 104.80;
6.16 overlay x 925.80, overlay y 445.80, scale w 106.40, scale h 106.40;
6.20 overlay x 922.00, overlay y 445.00, scale w 108.00, scale h 108.00;
...
```

Step 6: Run this batch file to create the final moving wormhole video:

```
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
```

Notes:

`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.

2.135 Dynamically change commands with "sendcmd" or "zmq"

In FFmpeg "commands" are a special case of options, which can be changed dynamically while FFmpeg is running. All commands are options, but not all options are commands. Commands can be changed either with "sendcmd" or with "zmq".

	sendcmd	zmq
Which options can be controlled?	All those that have support for commands.	All those that have support for commands.
When can an option be changed?	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.	Options can be changed in real time while FFmpeg is running, but the timing is not as exact as with sendcmd.
Timing accuracy	exact	about 100-200ms, as a rough estimate
Is it possible to decide in real time to change or not to change an option?	no	yes
Possible sources where data comes from	<ul style="list-style-type: none"> • *.cmd file • FFmpeg command line, as argument of sendcmd filter 	<ul style="list-style-type: none"> • from a second console window via "zmqsend" tool • from a batch file via "zmqsend" tool • zeromq library which is available for many programming languages • for example NetMQ library for C#
Is it possible to change an option for each frame, using an expression?	yes, with [expr] option	no, not with exact timing for each frame. However it's possible if exact timing for each frame isn't required

Note: For almost all commands, it doesn't matter if you set an option directly in the command line, or via sendcmd, or via zmq. The effect is the same in all three cases. But there is one exception from this rule: These are the options yaw, pitch and roll in the v360 filter. Directly in the command line that are absolute angles, however if sent via sendcmd or zmq that are relative angles. Don't try to understand it. It's stupid.

2.136 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 a file (*.cmd is recommended), or could also be defined in the command line.
- To find out which filters support commands, use the `ffmpeg -filters` command, or look in the documentation of the filter.
- 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: <https://stackoverflow.com/questions/56058909/ffmpeg-drawtext-and-live-coordinates-with-sendcmd-zmq>

- FFmpeg accepts a few keys while it's running. This is undocumented and can only be found in the source code in `ffmpeg.c` in the function `check_keyboard_interaction()`

?	show this help
+	increase verbosity
-	decrease verbosity
c	Send command to first matching filter supporting it
C	Send/Queue command to all matching filters
d D	cycle through available debug modes
h	dump packets/hex press to cycle through the 3 states
q	quit
s	Show QP histogram

This is an example for cropping a square region out of a rectangular video, while changing the x,y position of the cropped region as a function of time with linear interpolation:

```
ffmpeg -i in.mp4 -lavfi sendcmd=f=coord.cmd,crop@1=w=800:h=800:x=0:y=0 out.mp4
pause
```

The top left coordinates of the crop window are specified in the file "coord.cmd":

```
0.0-1.0 [expr] crop@1 x 'lerp(0,100,TI)';
0.0-1.0 [expr] crop@1 y 'lerp(0,100,TI)';
1.0-2.0 [expr] crop@1 x 'lerp(100,150,TI)';
1.0-2.0 [expr] crop@1 y 'lerp(100,220,TI)';
2.0-3.0 [expr] crop@1 x 'lerp(150,80,TI)';
2.0-3.0 [expr] crop@1 y 'lerp(220,220,TI)';
and so on...
```

Note: In this example "crop@1" can be replaced by "crop", because there is only one crop filter in the filter chain.

Note: The same effect could also be realized with the 'zoompan' filter, but that's more complicated because 'zoompan' doesn't support commands.

This is an example where the x,y position of the cropped region changes at certain times, without any interpolation:

```
ffmpeg -i in.mp4 -vf sendcmd=f=coord.cmd,crop=800:800:0:36 out.mp4  
pause
```

The top left coordinates of the crop window are specified in the file "coord.cmd":

```
9.50 crop x 30;  
9.50 crop y 60;  
15.00 crop x 100;  
15.00 crop y 100;
```

In most cases it's not possible to change the size of a video stream dynamically, but in some cases it does work, for example if "scale" is immediately followed by "overlay":

```
ffmpeg -f lavfi -i color=red:s=800x600 -f lavfi -i color=yellow:s=2x2 -lavfi [1]sendcmd="2.0 scale w  
400",scale=w=200:h=200[a];[0][a]overlay=x=100:y=100 -t 5 -y out.mp4  
pause
```

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:

```
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=0.005"                :: Rotation speed, 1.0 means one revolution per frame, 0.005 means 8s per rev. at 25fps  
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=f=sendcmd.cmd,v360@1=e:e:pitch=0:yaw=0:roll=0:reset_rot=1,scroll=h=  
%R1%,v360@2=e:perspective:pitch=0:yaw=0:roll=0:alpha_mask=1:reset_rot=1,scale=%D%:%D%[a],[0][a]  
overlay=x=%X%:y=%Y% -t %T% -y out.mp4  
  
pause
```

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 50;
0.00 v360@2 yaw 0;
0.00 v360@2 roll 0;

# 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 yaw 'lerp(0,30,TI)';
8.00-9.001 [expr] v360@1 pitch 'lerp(0,-60,TI)';

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

14.00-15.001 [expr] v360@2 pitch 'lerp(50,90,TI)';

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

18.00-21.001 [expr] v360@1 yaw 'lerp(30,-74,TI)';
18.00-21.001 [expr] v360@1 pitch 'lerp(-60,-49,TI)';
```

Note: If specified in the command line, the v360 rotation angles are absolute angles. However if sent via sendcmd, they become relative angles. That's why the "reset_rot=1" option is used. It automatically resets the rotations to zero before a new relative rotation is applied.

In my opinion "relative rotations per frame" are the same thing as rotational velocity, and I don't understand why the same options (yaw, pitch and roll) are used for absolute angles in the command line and rotational velocities in sendcmd. It would be much clearer if different options would be used for different things.

The "reset_rot" option can be set to 0, 1 or -1. The meaning of -1 is unclear. It's not documented.

A few examples for sendcmd and single / double quotes:

<code>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</code>	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!
<code>ffmpeg -i in1.mp4 -i in2.mp4 -lavfi "[0][1]sendcmd='3.0 streamselect map 1',streamselect=inputs=2:map=0" out.mp4</code>	This doesn't work because sendcmd accepts only one input
<code>ffmpeg -i in1.mp4 -i in2.mp4 -lavfi sendcmd='3.0 streamselect map 1',streamselect=inputs=2:map=0 out.mp4</code>	This is the example from the streamselect documentation, doesn't work under Windows.
<code>ffmpeg -i in1.mp4 -i in2.mp4 -lavfi sendcmd="3.0 streamselect map 1",streamselect=inputs=2:map=0 out.mp4</code>	Single quotes replaced by double quotes, works under Windows.
<code>ffmpeg -i in1.mp4 -i in2.mp4 -lavfi "sendcmd='3.0 streamselect map 1',streamselect=inputs=2:map=0" out.mp4</code>	Double quotes added for the whole filter chain, single quotes for sendcmd argument, works under Windows.
<code>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</code>	[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
<code>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</code>	[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."

2.137 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.

```
ffplay -dumpgraph 1 -f lavfi "color=s=200x200:c=red[1];color=s=200x200:c=blue[r];nullsrc=s=400x200,zmq[bg];[bg]
[1]overlay[bg+1];[bg+1][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/>

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

Copy the files zmqsnd.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:

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

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:

```
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.

2.138 Changing an input image while FFmpeg is running

Let's assume you want to change an input image while FFmpeg is running. As a simple example, we want to show the image on the screen.

The trick is not to use the image as input, but instead a symbolic link (symlink) which is pointing to the image:

```
ffmpeg -framerate 1 -stream_loop -1 -f image2 -i symlink.lnk -window_x 0 -window_y 0 -f sdl2 -  
pause
```

The complicated part is how to create and change the symlink. You need administrator permission to do this. Type "cmd" in the Windows search window, then click on "Run as administrator". Go to your working directory by typing (for example)

```
cd C:\Users\astro\Desktop\test
```

Before you can start FFmpeg, you must create the symlink which is pointing to the first image:

```
mklink symlink.lnk image1.png
```

Then FFmpeg can be started and it should show the first image.

As the next step, we want to change the symlink to the second image, while FFmpeg is running. To do this, type the following three commands:

```
mklink temp.lnk image2.png  
copy /l /y temp.lnk symlink.lnk  
del temp.lnk
```

In the first command line a temporary symlink to the new image is created.

In the second command line the symlink is changed to the new target and FFmpeg should immediately show the new image.

In the third command line the temporary symlink is deleted. That must be done because a new symlink can't be created if one with the same name does already exist.

See also: <https://stackoverflow.com/questions/11296491/change-target-for-symbolic-link-in-windows>

Note: You can also create links by right-clicking in a folder (New --> Shortcut), but those links don't work for this purpose. They are somehow different from the symlinks which are used here.

Note: It was also suggested to use -readrate 1 (which is the same as -re). This didn't work when I tested it, but it might be useful in other cases.

2.139 FFmpeg and C# Programming

Start a FFplay process with console window and video window:

```
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:

```
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>

Stop the FFplay process:

```
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:

```
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:

```
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 \\"tests SRC2=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".

2.140 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
```

Two notes from Andrei B.:

- The "vidstab" filter has the drawback that it gets confused by rolling shutter from CMOS sensors.
- A very good (and free) tool that can do much better is VirtualDub with Deshaker 3.0 filter. This filter has a rolling shutter factor input and can greatly improve on reducing the wobbliness of a stabilized video. Its documentation includes instructions on how to measure your camera's rolling shutter factor.

Some notes from Steffen Richter:

- "shakiness" option seems to have no effect.
- "tripod" may lead to strong z rotations, if no relevant movements are detected.
- "relative": Different from the documentation, the default seems to be "1", which also makes sense.
- "zoomspeed": Values between 0.01 and 0.1 are useful.

This is a test for comparing the "deshake" filter with the "vidstabdetect/vidstabtransform" filters:

```
rem deshake
ffmpeg -i test.mov -lavfi split[a][b];[a]deshake=rx=64:ry=64:edge=0[c];[b][c]hstack -y deshake.mov

rem vidstabdetect
ffmpeg -i test.mov -lavfi vidstabdetect=shakiness=10:show=1 -y dummy.mov

rem vidstabtransform with optzoom=0 (which means no zoom, so that borders are visible)
ffmpeg -i test.mov -lavfi split[a][b];[a]vidstabtransform=smoothing=50:crop=black:optzoom=0[c];[b][c]hstack -y
vidstab.mov

rem vidstabtransform with optzoom=1 (which means optimized zoom, so that no borders are visible)
ffmpeg -i test.mov -lavfi split[a][b];[a]vidstabtransform=smoothing=50:crop=black:optzoom=1[c];[b][c]hstack -y
vidstab_zoom.mov

pause
```

By comparing the results, I found that the two-stage solution with "vidstabdetect/vidstabtransform" gives much better results than the one-stage "deshake" solution.

In "vidstabtransform" it's possible to set the "smoothing" parameter which defines the number of frames ($2 * \text{smoothing} + 1$) for low-pass filtering of the camera movements. The default "smoothing" value is 10 (which means 21 frames), but I found it useful to use higher values between 20 and 50.

"vidstabtransform" does correct x,y translations and also rotations.

The option "optzoom=1" does automatically choose a suitable zoom factor, so that there are no no-data areas at the borders visible.

Note: I think there is a bug in vidstabtransform, the option crop=black doesn't work as described. The no-data borders are filled with the colors of the edge of the image. Black borders appear only when the image shift becomes very large. But that doesn't matter, because with "optzoom=1" (which is the default) the no-data borders are anyway cropped away.

See also: <http://oioiioixiii.blogspot.com/2016/09/ffmpeg-video-stabilisation-using.html>

This is an example for removing a linear drift from a video. The first and the last image is extracted and the x,y coordinates of an object is measured in these two images. The diffence between the coordinates is the motion vector. Then the "crop" filter is used for cropping a window out of the input video, where the top left coordinates of the crop window are a function of time. The input size was 2048x2048, and the output size is reduced to 1856x1856 (which is the input size minus the larger of the two motion vectors).

```
rem Extract the first frame
ffmpeg -i M31-STATT-2020.mov -frames 1 -y first_frame.jpg

rem Extract the last frame
ffmpeg -sseof -0.2 -i M31-STATT-2020.mov -update 1 -y last_frame.jpg

rem Coordinates of object in first frame: 1026, 1091
rem Coordinates of object in lastframe: 1131, 1282
rem Motion vector: +105, +191
rem Duration: 17.23s

ffmpeg -i M31-STATT-2020.mov -lavfi crop=x='105/17.23*t':y='191/17.23*t':w=1856:h=1856 -y out1.mp4

pause
```

2.141 Stabilization of 360° Videos

Note: This stabilization method does no longer work because the behaviour of the v360 filter was changed 25-Oct-2020. The yaw, pitch and roll rotation angles are now interpreted as relative angles, if they are sent via sendcmd. See also ticket 9447. There was a new option "reset_rot" added which resets the rotation angles, so that absolute rotations can be simulated. I'm not sure if it could be used in this example. Recommended workaround: Use the spherical stabilizer in DaVinci Resolve.

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:

```
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
pause
```

Now use this image for creating a test video with rotations around different axes:

```
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
pause
```

The rotations are defined in the file "sendcmd.cmd": I don't really understand what reset_rot=-1 is doing.

```
0.0-9.0 [expr] v360 reset_rot '-1';

0.0-1.0 drawtext reinit 'text=';
0.0-1.0 [expr] v360 yaw '0';
0.0-1.0 [expr] v360 pitch '0';
0.0-1.0 [expr] v360 roll '0';
```

```

1.0-2.0 drawtext reinit 'text=PITCH';
1.0-2.0 [expr] v360 yaw '0';
1.0-2.0 [expr] v360 pitch 'lerp(0,45,TI)';
1.0-2.0 [expr] v360 roll '0';

2.0-3.0 drawtext reinit 'text=YAW';
2.0-3.0 [expr] v360 yaw 'lerp(0,45,TI)';
2.0-3.0 [expr] v360 pitch '45';
2.0-3.0 [expr] v360 roll '0';

3.0-4.0 drawtext reinit 'text=PITCH + YAW';
3.0-4.0 [expr] v360 yaw 'lerp(45,90,TI)';
3.0-4.0 [expr] v360 pitch 'lerp(45,90,TI)';
3.0-4.0 [expr] v360 roll '0';

4.0-5.0 drawtext reinit 'text=ROLL';
4.0-5.0 [expr] v360 yaw '90';
4.0-5.0 [expr] v360 pitch '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 yaw '90';
5.0-6.0 [expr] v360 pitch 'lerp(90,135,TI)';
5.0-6.0 [expr] v360 roll 'lerp(45,90,TI)';

6.0-7.0 drawtext reinit 'text=YAW + ROLL';
6.0-7.0 [expr] v360 yaw 'lerp(90,135,TI)';
6.0-7.0 [expr] v360 pitch '135';
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 yaw 'lerp(135,180,TI)';
7.0-8.0 [expr] v360 pitch '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 [expr] v360 yaw '180';
8.0-9.0 [expr] v360 pitch '180';
8.0-9.0 [expr] 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 video
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

pause
```

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:

Time [s]	Frames	Rotation	Movement in image A (center of equirectangular video)	Movement in image B (90° right of center of equirectangular video)
0-1	0-25	--	--	--
1-2	25-50	pitch	down	counter-clockwise rotation
2-3	50-75	yaw	left	down and left
3-4	75-100	pitch + yaw	down and left	first down and left, then down
4-5	100-125	roll	counter-clockwise rotation	up
5-6	125-150	pitch + roll	first down and right, then right	up and right
6-7	150-175	yaw + roll	first up, then up and left	first up, then up and left
7-8	175-200	pitch + yaw + roll	first up, then up and left	first up, then up and left
8-9	200-224	--	--	--

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:

`Yaw_angle_in_radians = -c * horizontal_shift_in_image_A`

`Pitch_angle_in_radians = c * vertical_shift_in_image_A`

`Roll_angle_in_radians = -c * vertical_shift_in_image_B`

where `c` is a constant, `c = Pi / 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:

```
set "IN=test.mp4"          :: Equirectangular input video
set "B=100"                :: Box size in pixels

ffmpeg -i %IN% -vf drawbox=x=iw/2-%B%/2:y=ih/2-%B%/2:w=%B%:h=%B%:color=red:thickness=5,drawbox=x=3/4*iw-%B%/2:y=ih/2-%B%
/%2:w=%B%:h=%B%:color=red:thickness=5 -y dummy.mp4

pause
```

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 a C# program, which produces a file "stabilize.cmd" which can later be used as input for FFmpeg's sendcmd filter.

(See the C# source code of the improved version in the next chapter)

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:

```
set "IN=test.mp4"          :: Equirectangular input video
set "T=10"                 :: 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
```

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.

2.142 Stabilization of 360° Videos, improved

Note: This stabilization method does no longer work because the behaviour of the v360 filter was changed 25-Oct-2020. The yaw, pitch and roll rotation angles are now interpreted as relative angles, if they are sent via sendcmd. See also ticket 9447. There was a new option "reset_rot" added which resets the rotation angles, so that absolute rotations can be simulated. I'm not sure if it could be used in this example. Recommended workaround: Use the spherical stabilizer in DaVinci Resolve.

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:

	Movement in images					
Rotation	XA	XB	XC	XD	XE	XF
pitch	down		up		down	down
yaw	left	left	left	left		
roll		up		down	left	right

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 video
```

```

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
ffmpeg -i %IN% -vf v360=input=e:output=e:yaw=180,crop=w=%B%:h=%B% -start_number 0 -t %T% -y XC%%04d.png
ffmpeg -i %IN% -vf v360=input=e:output=e:yaw=-90,crop=w=%B%:h=%B% -start_number 0 -t %T% -y XD%%04d.png
ffmpeg -i %IN% -vf v360=input=e:output=e:pitch=90,crop=w=%B%:h=%B% -start_number 0 -t %T% -y XE%%04d.png
ffmpeg -i %IN% -vf v360=input=e:output=e:pitch=-90,crop=w=%B%:h=%B% -start_number 0 -t %T% -y XF%%04d.png

pause

```

My C# source code can be downloaded here: <http://www.astro-electronic.de/source/v360stabilizer.zip>

This is the batch file for applying the rotations to the input video:

```

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

```

Note: There is a very good spherical stabilizer in DaVinci Resolve, highly recommended.

2.143 Remap Video-in-Video with perspective filter

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 x0,y0 (top left), x1,y1 (top right), x2,y2 (bottom left) and x3,y3 (bottom right) and must me 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:

$a = (x - x_0) / (x_1 - x_0)$ 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

$x_4 = x$

$y_4 = y_0 + a * (y_1 - y_0)$

We do the same thing for the lower edge:

$b = (x - x_2) / (x_3 - x_2)$

$x_5 = x$

$y_5 = y_2 + b * (y_3 - y_2)$

Parameter c describes where the point x,y lies on the line segment between points 4 and 5:

$c = (y - y_4) / (y_5 - y_4)$

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

$x_{map} = (a + c * (b - a))$

$y_{map} = c$

2.144 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 -f lavfi -i nullsrc=size=%W%x%H% -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=%W%x%H% -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=%W%x%H% -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=%W%x%H% -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:



me.jpg



bigger_nose.jpg



smaller_nose.jpg

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.

The `displace` filter isn't fully compatible with 10-bit data. Dithering is introduced.

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 -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)*(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:

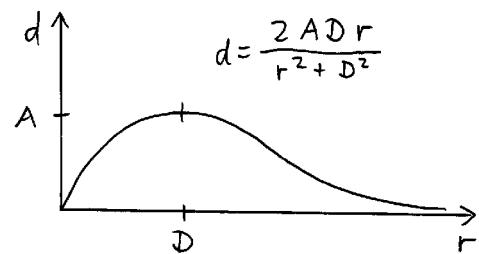


me.jpg



big_eyes.jpg

Mathematics for this distortion:



$$d = \frac{2ADr}{r^2 + D^2} \quad r = \sqrt{(x - cx)^2 + (y - cy)^2} \quad \frac{d}{r} = \frac{2AD}{(x - cx)^2 + (y - cy)^2 + D^2} \quad dx = (x - cx) \frac{d}{r} \quad dy = (y - cy) \frac{d}{r}$$

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

2.145 Noise reduction

FFmpeg has several filters for video noise reduction (denoising):

Filter	Description	Notes and Examples
atadenoise	Apply an Adaptive Temporal Averaging Denoiser to the video input	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
bm3d	Denoise frames using Block-Matching 3D algorithm	very very slow, currently implemented as spatial only, algorithm considered as one of the state of art denoisers; LGPL
chromanr	Reduce chrominance noise	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 "stepw" and "steph" options it's possible to use only a subset of the neighbour pixels for averaging.
dctdnoiz	Denoise frames using 2D DCT (frequency domain filtering)	very very slow: spatial only, blurs too much; LGPL
fftdenoiz	Denoise frames using 3D FFT (frequency domain filtering)	slow, spatial and limited temporal, using Fast Fourier Transform, may have introduce ringing with bad settings; LGPL
hqdn3d	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.	fast, both spatial and temporal, does basically lowpass by destroying high frequencies, blurs with extreme settings; GPL Example: hqdn3d=4:4:9:9
nlmeans	Denoise frames using Non-Local means algorithm	very slow, currently implemented as spatial only, algorithm considered as one of the state of art denoisers; LGPL
owdenoise	Apply Overcomplete Wavelet denoiser	very very very slow, spatial only, wavelet; GPL Example: owdenoise=ls=25
removegrain	Spatial denoiser for progressive video	fast, spatial only, limited usecase
vaguedenoiser	Apply a wavelet based denoiser	slow, spatial only, pretty good, wavelet; LGPL
tmix	Noise reduction by averaging up to 1024 successive frames	Not suitable for moving objects. Example: tmix=frames=20
tmedian	Noise reduction by calculating the median out of up to 127 successive frames	Example: tmedian=radius=20

Special thanks to Paul B Mahol who posted most of these notes in the FFmpeg-user list on October 27, 2019

2.146 -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.

2.147 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
```

```
pause
```

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
```

```
pause
```

2.148 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:

```
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 mpegs 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.

For informations about UDP (User Datagram Protocol) see https://en.wikipedia.org/wiki/User_Datagram_Protocol

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:

```
set "T=20"      :: Duration in seconds

ffmpeg -f mpegs -i udp://127.0.0.1:1234 -t %T% -y out.mp4
```

For testing the stream you can use this batch file:

```
ffmpeg -f mpegs -i udp://127.0.0.1:1234 -f sd12 -
```

or alternatively this batch file:

```
ffplay -f mpegs -i udp://127.0.0.1:1234
```

See also <https://trac.ffmpeg.org/wiki/Capture/Lightning> (that's a complicated solution with external software)

2.149 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

```
ffmpeg -i IMG_044x.jpg -pix_fmt yuvj422p -q 0 IMG_044.jpg  
pause
```

Set the pixel format of a video to 4:4:4 and scale the video to 1920x1080

```
ffmpeg -i input.MOV -pix_fmt yuv444p -s 1920x1080 out.mov  
pause
```

In a filter chain the format can be set as follows:

```
ffmpeg -i input.MOV -lavfi "format=pix_fmts=rgb24" -y out.mp4  
pause
```

Which is the same as:

```
ffmpeg -i input.MOV -lavfi "format=rgb24" -y out.mp4  
pause
```

See also: <https://trac.ffmpeg.org/wiki/Chroma%20Subsampling>

Chroma subsampling	8-bit format	10-bit format	Notes
4:4:4	yuv444p	yuv444p10le	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. Note that "4:4:4" may instead be referring to R'G'B' color space, which implicitly also does not have any chroma subsampling.
4:2:2	yuv422p	yuv422p10le	The horizontal chroma resolution is halved. This reduces the bandwidth of an uncompressed video signal by one-third with little to no visual difference.
4:2:0	yuv420p	yuv420p10le	Chroma is subsampled at a factor of 2 both horizontally and vertically.

Source: Wikipedia

RGB and gray pixel formats (this is only a subset of the available formats):

	NB_COMPONENTS	BITS_PER_PIXEL	Notes
rgb24, bgr24	3	24	packed
gbp	3	24	planar
gray	1	8	
argb, rgba, abgr, bgra	4	32	packed
gray16be, gray16le	1	16	
rgb48be, rgb48le, bgr48be, bgr48le	3	48	packed
brgp16be, brgp16le	3	48	planar
gray14be, gray14le	1	14	
grayf32be, grayf32le	1	32	float grayscale

Example for creating an image with float grayscale:

```
ffmpeg -f lavfi -i nullsrc=size=100x100 -vf format=pix_fmts=grayf32le,geq='(X-50)/25' -frames 1 -y test.pfm
```

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:

```
ffmpeg -pix_fmts  
pause
```

List of all pixel formats:

```
Pixel formats:  
I.... = Supported Input format for conversion  
.O... = Supported Output format for conversion  
.H.. = Hardware accelerated format  
.P. = Palettized format  
.B = Bitstream format  
FLAGS NAME          NB_COMPONENTS BITS_PER_PIXEL  
----  
IO... yuv420p        3            12  
IO... yuyv422         3            16  
IO... rgb24           3            24  
IO... bgr24           3            24  
IO... yuv422p         3            16  
IO... yuv444p         3            24  
IO... yuv410p         3            9  
IO... yuv411p         3            12  
IO... gray             1            8  
IO..B monow           1            1  
IO..B monob           1            1  
I..P. pa18             1            8  
IO... yuvj420p         3            12  
IO... yuvj422p         3            16  
IO... yuvj444p         3            24  
IO... uvyv422          3            16  
..... uyyvyy411         3            12  
IO... bgr8              3            8  
.O..B bgr4              3            4  
IO... bgr4_byte         3            4  
IO... rgb8              3            8  
.O..B rgb4              3            4  
IO... rgb4_byte         3            4  
IO... nv12              3            12  
IO... nv21              3            12  
IO... argb              4            32  
IO... rgba              4            32  
IO... abgr              4            32  
IO... bgra              4            32  
IO... gray16be          1            16  
IO... gray16le          1            16  
IO... yuv440p           3            16  
IO... yuvj440p          3            16
```

IO...	yuva420p	4	20
IO...	rgb48be	3	48
IO...	rgb48le	3	48
IO...	rgb565be	3	16
IO...	rgb565le	3	16
IO...	rgb555be	3	15
IO...	rgb555le	3	15
IO...	bgr565be	3	16
IO...	bgr565le	3	16
IO...	bgr555be	3	15
IO...	bgr555le	3	15
..H...	vaapi_moco	0	0
..H...	vaapi_idct	0	0
..H...	vaapi_vld	0	0
IO...	yuv420p16le	3	24
IO...	yuv420p16be	3	24
IO...	yuv422p16le	3	32
IO...	yuv422p16be	3	32
IO...	yuv444p16le	3	48
IO...	yuv444p16be	3	48
..H...	dxva2_vld	0	0
IO...	rgb444le	3	12
IO...	rgb444be	3	12
IO...	bgr444le	3	12
IO...	bgr444be	3	12
IO...	ya8	2	16
IO...	bgr48be	3	48
IO...	bgr48le	3	48
IO...	yuv420p9be	3	13
IO...	yuv420p9le	3	13
IO...	yuv420p10be	3	15
IO...	yuv420p10le	3	15
IO...	yuv422p10be	3	20
IO...	yuv422p10le	3	20
IO...	yuv444p9be	3	27
IO...	yuv444p9le	3	27
IO...	yuv444p10be	3	30
IO...	yuv444p10le	3	30
IO...	yuv422p9be	3	18
IO...	yuv422p9le	3	18
IO...	gbrp	3	24
IO...	gbrp9be	3	27
IO...	gbrp9le	3	27
IO...	gbrp10be	3	30
IO...	gbrp10le	3	30
IO...	gbrp16be	3	48
IO...	gbrp16le	3	48
IO...	yuva422p	4	24
IO...	yuva444p	4	32

IO...	yuva420p9be	4	22
IO...	yuva420p9le	4	22
IO...	yuva422p9be	4	27
IO...	yuva422p9le	4	27
IO...	yuva444p9be	4	36
IO...	yuva444p9le	4	36
IO...	yuva420p10be	4	25
IO...	yuva420p10le	4	25
IO...	yuva422p10be	4	30
IO...	yuva422p10le	4	30
IO...	yuva444p10be	4	40
IO...	yuva444p10le	4	40
IO...	yuva420p16be	4	40
IO...	yuva420p16le	4	40
IO...	yuva422p16be	4	48
IO...	yuva422p16le	4	48
IO...	yuva444p16be	4	64
IO...	yuva444p16le	4	64
..H..	vdpau	0	0
IO...	xyz12le	3	36
IO...	xyz12be	3	36
....	nv16	3	16
....	nv20le	3	20
....	nv20be	3	20
IO...	rgba64be	4	64
IO...	rgba64le	4	64
IO...	bgra64be	4	64
IO...	bgra64le	4	64
IO...	yvyu422	3	16
IO...	ya16be	2	32
IO...	ya16le	2	32
IO...	gbrap	4	32
IO...	gbrap16be	4	64
IO...	gbrap16le	4	64
..H..	qsv	0	0
..H..	mmal	0	0
..H..	d3d11va_vld	0	0
..H..	cuda	0	0
IO...	0rgb	3	24
IO...	rgb0	3	24
IO...	0bgr	3	24
IO...	bgr0	3	24
IO...	yuv420p12be	3	18
IO...	yuv420p12le	3	18
IO...	yuv420p14be	3	21
IO...	yuv420p14le	3	21
IO...	yuv422p12be	3	24
IO...	yuv422p12le	3	24
IO...	yuv422p14be	3	28

IO...	yuv422p14le	3	28
IO...	yuv444p12be	3	36
IO...	yuv444p12le	3	36
IO...	yuv444p14be	3	42
IO...	yuv444p14le	3	42
IO...	gbrp12be	3	36
IO...	gbrp12le	3	36
IO...	gbrp14be	3	42
IO...	gbrp14le	3	42
IO...	yuvj411p	3	12
I....	bayer_bggr8	3	8
I....	bayer_rggb8	3	8
I....	bayer_gbrg8	3	8
I....	bayer_grbg8	3	8
I....	bayer_bggr16le	3	16
I....	bayer_bggr16be	3	16
I....	bayer_rggb16le	3	16
I....	bayer_rggb16be	3	16
I....	bayer_gbrg16le	3	16
I....	bayer_gbrg16be	3	16
I....	bayer_grbg16le	3	16
I....	bayer_grbg16be	3	16
..H..	xvmc	0	0
IO...	yuv440p10le	3	20
IO...	yuv440p10be	3	20
IO...	yuv440p12le	3	24
IO...	yuv440p12be	3	24
IO...	ayuv64le	4	64
.....	ayuv64be	4	64
..H..	videotoolbox_vld	0	0
IO...	p010le	3	15
IO...	p010be	3	15
IO...	gbrap12be	4	48
IO...	gbrap12le	4	48
IO...	gbrap10be	4	40
IO...	gbrap10le	4	40
..H..	mediacodec	0	0
IO...	gray12be	1	12
IO...	gray12le	1	12
IO...	gray10be	1	10
IO...	gray10le	1	10
IO...	p016le	3	24
IO...	p016be	3	24
..H..	d3d11	0	0
IO...	gray9be	1	9
IO...	gray9le	1	9
IO...	gbrpf32be	3	96
IO...	gbrpf32le	3	96
IO...	gbrapf32be	4	128

IO...	gbraphf32le	4	128
..H..	drm_prime	0	0
..H..	opencl	0	0
IO...	gray14be	1	14
IO...	gray14le	1	14
IO...	grayf32be	1	32
IO...	grayf32le	1	32
IO...	yuva422p12be	4	36
IO...	yuva422p12le	4	36
IO...	yuva444p12be	4	48
IO...	yuva444p12le	4	48
IO...	nv24	3	24
IO...	nv42	3	24
..H..	vulkan	0	0
.....	y210be	3	20
I....	y210le	3	20
IO...	x2rgb10le	3	30
.....	x2rgb10be	3	30

2.150 Automatic format conversions

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
pause
```

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
pause
```

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."

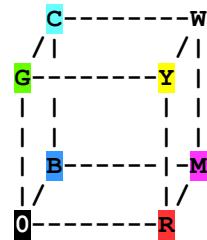
For more details about the scale filter see also: <https://trac.ffmpeg.org/wiki/Scaling>

Some important notes from the above website:

- When going from BGR (not RGB) to yuv420p the conversion is broken (off-by-one). Use -vf scale=flags=accurate_rnd to fix that.
- yuvjxxxp pixel formats are deprecated. Yet for x265 the workaround was implemented, but not for jpeg2000 and AV1. For those -vf scale=out_range=pc should be used.
- Conversion from YCbCr limited to RGB 16 bit is broken, use zscale instead of swscale
- Limited range RGB is not supported at all.
- Dither can be turned off using -vf scale=sws_dither=none
- One should always remember that YCbCr 4:4:4 8 bit is not enough to preserve RGB 8 bit, YCbCr 4:4:4 10 bit is required.
- The default for matrix in untagged input and output is always limited BT.601

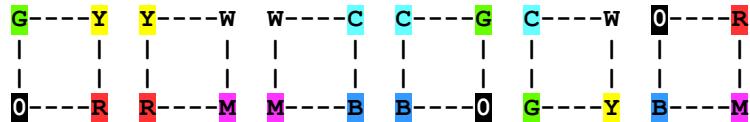
2.151 RGB Color Cube

This is the RGB color cube:

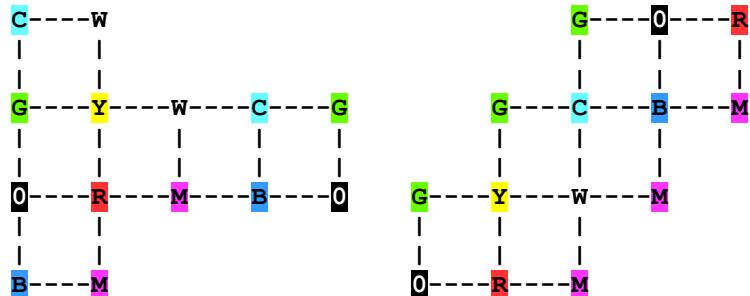


0 = black, R = red, G = green, B = blue, Y = yellow, C = cyan, M = magenta, W = white

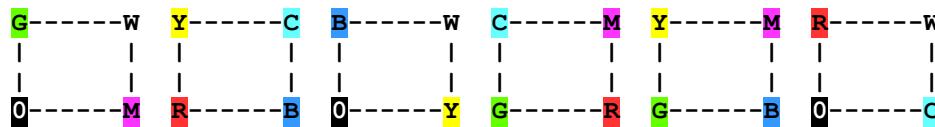
The cube has 6 faces:



There are several ways how to connect the faces to a net. Here are two examples:

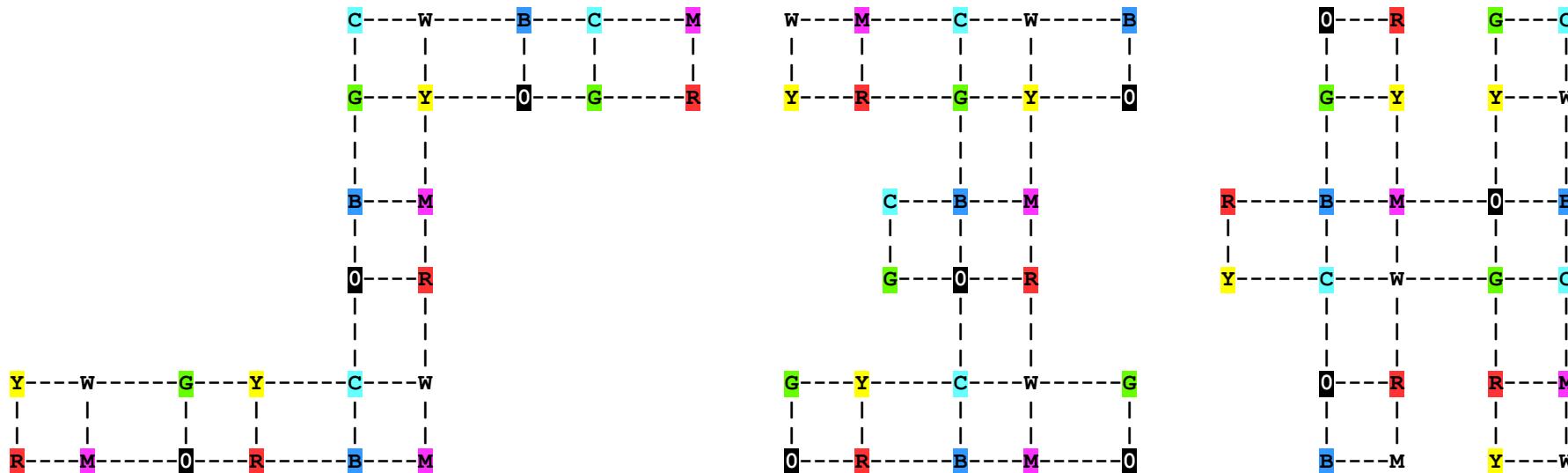


The cube does also contain 6 internal diagonal planes:



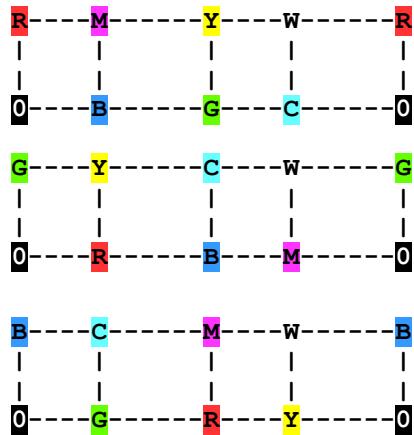
It's not possible to connect these 6 diagonal planes to a net, because they don't have any common edges.

But it's possible to find nets which contain all 6 faces and all 6 diagonal planes:



If you find a more compact net than the middle or right one, please let me know.

If you want to show all 6 sides and all 6 diagonal planes in a single image, it's more efficient to split the net in 3 parts:

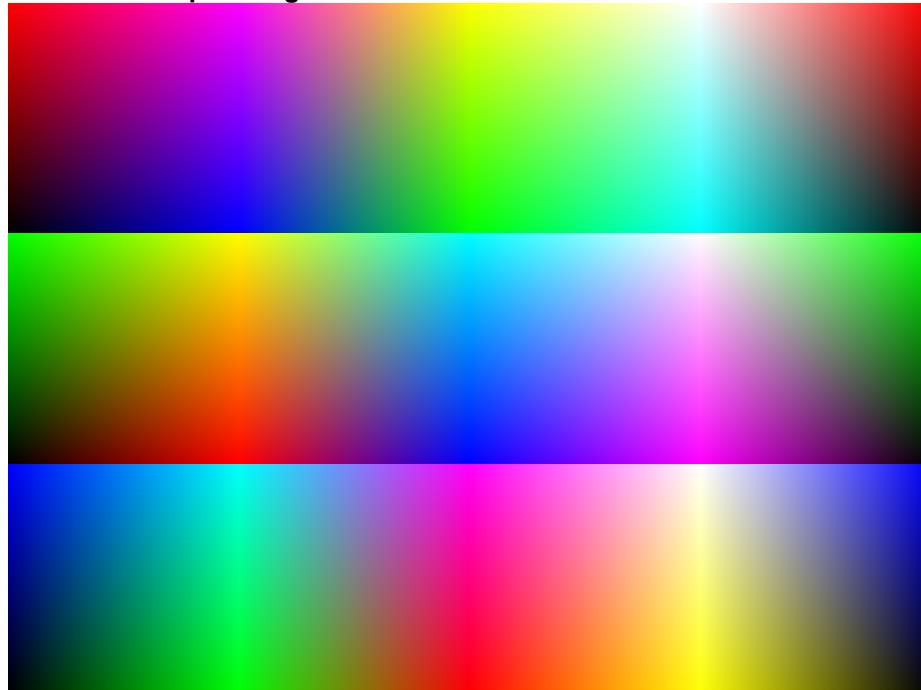


This is an example for creating an image which contains all 6 faces and all 6 diagonal planes of the RGB cube:

```
ffmpeg -f lavfi -i color=black:size=256x256 -lavfi split=4[a][b][c][d];^
[a]geq=r='lerp(255,0,Y/W)':g='0':b='lerp(0,255,X/W)':[aa];^
[b]geq=r='lerp(255,0,Y/W)':g='lerp(0,255,X/W)':b='lerp(255,0,X/W)':[bb];^
[c]geq=r='lerp(255,0,Y/W)':g='255':b='lerp(0,255,X/W)':[cc];^
[d]geq=r='lerp(255,0,Y/W)':g='lerp(255,0,X/W)':b='lerp(255,0,X/W)':[dd];^
[aa][bb][cc][dd]hstack=4,split=3[e1][e2][e3];^
[e2]colorchannelmixer=0:0:1:0:0:0:0:0:1:0:0:f2;^
[e3]colorchannelmixer=0:1:0:0:0:0:1:0:1:0:0:f3;^
[e1][f2][f3]vstack=3 -frames 1 -y out.png

pause
```

This is the output image:



2.152 Convert RGB to HSV or HSL

```

MAX := max(R,G,B), MIN := min(R,G,B)

H := 0                                if MAX = MIN
H := 60° * (0 + (G - B) / (MAX - MIN)) if MAX = R
H := 60° * (2 + (B - R) / (MAX - MIN)) if MAX = G
H := 60° * (4 + (R - G) / (MAX - MIN)) if MAX = B

if H < 0 then H := H + 360°

SHSV := 0          if MAX = 0
SHSV := (MAX - MIN) / MAX if MAX != 0

V := MAX

SHSL := 0          if MAX = 0
SHSL := 0          if MIN = 0
SHSL := (MAX - MIN) / (1 - |MAX + MIN - 1| ) if MAX != 0 and MIN != 0
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".

L := (MAX + MIN) / 2

```

Color table:

Color	HUE	R	G	B
red	0°	255	0	0
orange	30°	255	128	0
yellow	60°	255	255	0
green-yellow	90°	128	255	0
green	120°	0	255	0
blue-green	150°	0	255	128
cyan	180°	0	255	255
green-blue	210°	0	128	255
blue	240°	0	0	255
violet	270°	128	0	255
magenta	300°	255	0	255
blue-red	330°	255	0	128
white		255	255	255
black		0	0	0

This is a different method for calculating hue and chroma:

```

HUE := atan2(sqrt(3) * (G - B), 2 * R - G - B)
CHROMA := sqrt(3 * (G - B)^2 + (2 * R - G - B)^2)

```

2.153 Convert HSV to RGB

Given is HUE [0..360°] and S,V [0..1]

```
hi := |HUE / 60°|
f := HUE / 60° - hi
p := V * (1 - S)
q := V * (1 - S * f)
t := V * (1 - S * (1 - f))

(R,G,B) := (V,t,p)    if hi = 0    (from red to yellow)
(R,G,B) := (q,V,p)    if hi = 1    (from yellow to green)
(R,G,B) := (p,V,t)    if hi = 2    (from green to cyan)
(R,G,B) := (p,q,V)    if hi = 3    (from cyan to blue)
(R,G,B) := (t,p,V)    if hi = 4    (from blue to magenta)
(R,G,B) := (V,p,q)    if hi = 5    (from magenta to red)
```

Note: HSV = Hue-Saturation-Value

2.154 Video Codecs

-c:v mpeg4	This is the older MP4 codec, which is poorly documented. See also https://trac.ffmpeg.org/wiki/Encode/MPEG-4
-c:v libxvid	This MP4 codec is using the external library "libxvid". Search for "libxvid" in the documentation. See also http://www.ffmpeg.org/ffmpeg-codecs.html#libxvid
-c:v libx264	Newer H264 codec with better compression than mpeg4, but it's possible that the videos don't play on older computers. It's also possible to create lossless videos, as described in the next link: See also https://trac.ffmpeg.org/wiki/Encode/H.264 See also http://www.ffmpeg.org/ffmpeg-codecs.html#libx264_002c-libx264rgb
-c:v libx265	H265 is newer than H264, it has better compression than H264 and about the same quality. It's possible that the videos don't play on older computers. See also https://trac.ffmpeg.org/wiki/Encode/H.265 See also http://www.ffmpeg.org/ffmpeg-codecs.html#libx265
-c:v prores_ks	Apple ProRes encoder, example: <code>ffmpeg -i input.MOV -vcodec prores_ks -pix_fmt yuva444p10le -profile:v 4444 -bits_per_mb 8000 -s 1920x1080 out.mov</code> See also http://www.ffmpeg.org/ffmpeg-codecs.html#ProRes See also 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 DaVinci Resolve software. There isn't much documentation available for this codec and its options. Example: <code>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</code>
-c:v rawvideo	This means the output format is uncompressed raw video. It's good for lossless intermediate video files. Example: <code>ffmpeg -i in.mp4 -c:v rawvideo -f rawvideo -an -y out.raw</code> (This file can't contain audio) The drawback of this format is that you have to know the framerate, size and pixel format when you read such a file. This problem can be avoided by using the *.nut file format. This format can also contain audio. Example: <code>ffmpeg -i in.mp4 -c:v rawvideo -y out.nut</code>
-c:v ffv1 -c:v ffv1 -level 3	FFV1 is a lossless video codec which comes in two versions, 1 and 3. See also https://trac.ffmpeg.org/wiki/Encode/FFV1

For the "mpeg4" and "libxvid" codecs you can select a video quality level with `-q:v n`, where *n* is a number from 1-31, with 1 being highest

quality/largest filesize and 31 being the lowest quality/smallest filesize. This is a variable bit rate mode.

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 (for 8-bit, but not for 10-bit), 23 is the default, and 51 is worst quality possible. For lossless 10 bit -qp 0 is required.

This is in more detail explained here: 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:

film	use for high quality movie content; lowers deblocking
animation	good for cartoons; uses higher deblocking and more reference frames
grain	preserves the grain structure in old, grainy film material
stillimage	good for slideshow-like content
fastdecode	allows faster decoding by disabling certain filters
zerolatency	good for fast encoding and low-latency streaming

List all possible internal presets and tunes:

```
ffmpeg -hide_banner -f lavfi -i nullsrc -c:v libx264 -preset help -f mp4 -  
pause
```

2.155 Bitrate

See also <https://trac.ffmpeg.org/wiki/Limiting%20the%20output%20bitrate>

-b:v -minrate -maxrate -bufsize

2.156 Keyframes

-g 1 Use every frame as a keyframe

2.157 Open GOP and closed GOP

The difference between open and closed GOP (Group of Pictures) is explained here:

<https://streaminglearningcenter.com/blogs/open-and-closed-gops-all-you-need-to-know.html>

2.158 Video codecs with alpha channel

These are a few examples for exporting videos with alpha channel:

```
rem PNG images (lossless compression, output pixel format is rgba)
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=0.2 -pix_fmt rgba -y test_png_8bit_%%5d.png

rem PNG images (lossless compression, output pixel format is rgba64be)
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=0.2 -pix_fmt rgba64be -y test_png_16bit_%%5d.png

rem Apple ProRes (in all four cases the output pixel format is yuva444p12le)
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt rgba -c:v prores_ks -y test_prores1.mov
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt rgba64le -c:v prores_ks -y test_prores2.mov
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt yuva444p10le -c:v prores_ks -y test_prores3.mov
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt yuva444p12le -c:v prores_ks -y test_prores4.mov

rem Rawvideo (uncompressed, output pixel format is yuva444p10le)
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt yuva444p10le -c:v rawvideo -y test_rawvideo1.nut

rem Rawvideo (uncompressed, output pixel format is yuva444p12le)
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt yuva444p12le -c:v rawvideo -y test_rawvideo2.nut

rem Rawvideo (uncompressed, output pixel format is rgba)
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt rgba -c:v rawvideo -y test_rawvideo3.nut

rem Rawvideo (uncompressed, output pixel format is rgba64le)
ffmpeg -f lavfi -i testsrc2=s=1920x1080:d=4 -pix_fmt rgba64le -c:v rawvideo -y test_rawvideo4.nut

pause
```

Note: The *.nut format is unique for FFmpeg.

Note: alpha=0 means transparent, alpha=255 means opaque

See also https://www.digitalrebellion.com/blog/posts/list_of_video_formats_supporting_alpha_channels

2.159 Metadata

Global metadata can be saved in a text file as follows:

```
ffmpeg -i input.mp4 -f ffmetadata metadata.txt  
pause
```

If you also need the metadata from the video and audio streams (which may contain more informations), use this command line:

```
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  
pause
```

The metadata can be re-inserted into a video as follows:

```
ffmpeg -i input.mp4 -i metadata.txt -map_metadata 1 -codec copy output.mp4  
pause
```

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  
pause
```

Unfortunately FFmpeg can't insert the metadata that is required for a spherical 360° video.

This website describes which metadata tags are actually written to the output files: https://wiki.multimedia.cx/index.php/FFmpeg_Metadata

There is also a list on this page: <https://trac.ffmpeg.org/wiki/FilteringGuide>

FFmpeg can't write EXIF metadata to *.jpg images (February 2021).

Note: By default, "-filter_complex" or "-lavfi" don't copy the metadata from the input to the output.

Also it's not possible to copy the metadata from the input to the output with "-map_metadata 0".

The metadata map must be explicitly specified. For example, this would be -map_metadata:s:v:0 0:s:v:0. For videos with multiple streams however, all output streams (regardless of whether or not they are filtered) would have to be specified this way to preserve all stream metadata (since a single map disables all mappings). (Source: <https://trac.ffmpeg.org/ticket/9649>)

2.160 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.

For more details about "null" filter see also: <https://trac.ffmpeg.org/wiki/Null>

2.161 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 it 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`.

2.162 Filenames for images

Image filenames can contain variables. Please note that in a Windows batch file the % character must be replaced by two %% characters.

Variable	Description	Notes
%nd	A sequential n-digit number	The start number can be specified with the -start_number option. The default value is 1.
%0nd	A sequential number that is 0-padded to n digits	
%Y	Year	Only available if the "-strftime 1" option is used. Example for embedding date and time in the filename: <code>Screenshot-%Y-%m-%d-%H-%M-%S.jpg</code>
%m	Month	
%d	Day	
%H	Hours	
%M	Minutes	
%S	Seconds	Only available if the "-frame_pts 1" option is used. Example for embedding the PTS in the filename: <code>Screenshot-%d.jpg</code>
%d	PTS of the frame (Presentation time stamp)	

2.163 Create two-dimensional gradients

This is an example for making a two-dimensional gradient by defining the colors in the four corners:

```
set "SIZE=256x256"    :: Size
set "R1=255"          :: Red   \
set "G1=0"            :: Green  top left corner
set "B1=0"            :: Blue   /
set "R2=255"          :: Red   \
set "G2=255"          :: Green  top right corner
set "B2=0"            :: Blue   /
set "R3=0"            :: Red   \
set "G3=0"            :: Green  bottom left corner
set "B3=255"          :: Blue   /
set "R4=255"          :: Red   \
set "G4=0"            :: Green  bottom right corner
set "B4=255"          :: Blue   /


ffmpeg -f lavfi -i color=black:size=%SIZE% -lavfi geq=r='lerp(lerp(%R1%,%R2%,X/W),lerp(%R3%,
%R4%,X/W),Y/W)':g='lerp(lerp(%G1%,%G2%,X/W),lerp(%G3%,%G4%,X/W),Y/W)':b='lerp(lerp(%B1%,%B2%,X/W),lerp(%B3%,
%B4%,X/W),Y/W)' -frames 1 -y out.png

pause
```

This is the output image:

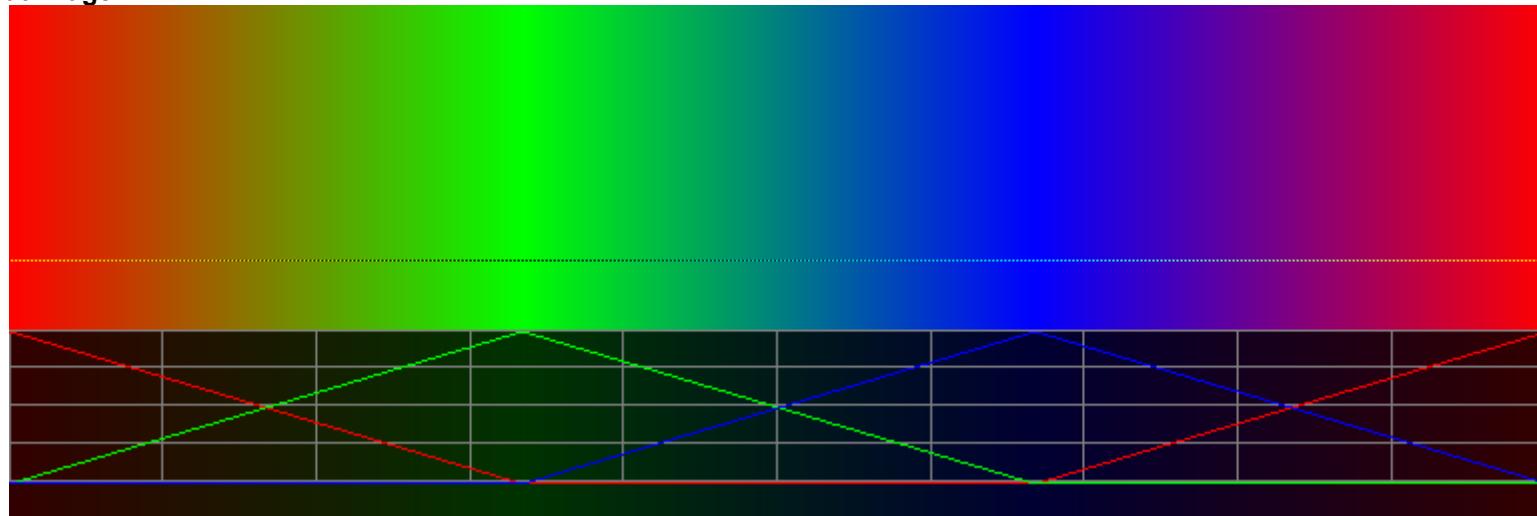


2.164 Make spectrum images

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.

```
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  
pause
```

This is the output image:



The same thing can also be done with the "gradients" video source:

```
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  
pause
```

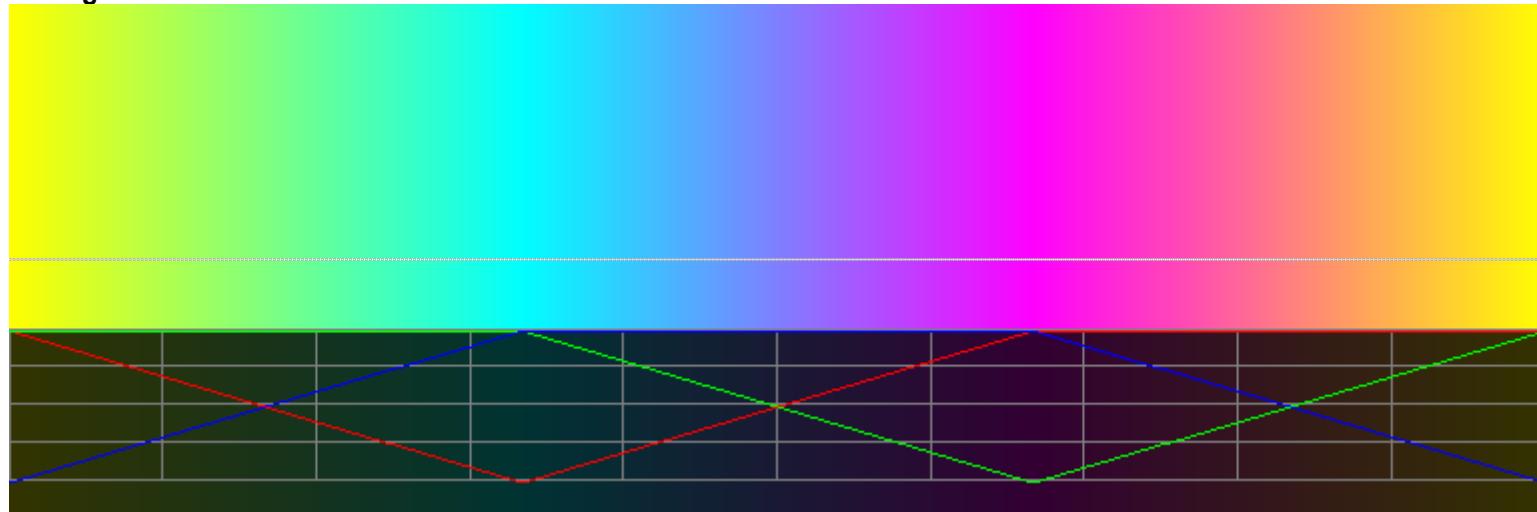
Note: "green" is 0x008000, "lime" is 0x00FF00

This command line creates an image with a spectrum yellow - cyan - magenta - yellow:

```
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
```

```
pause
```

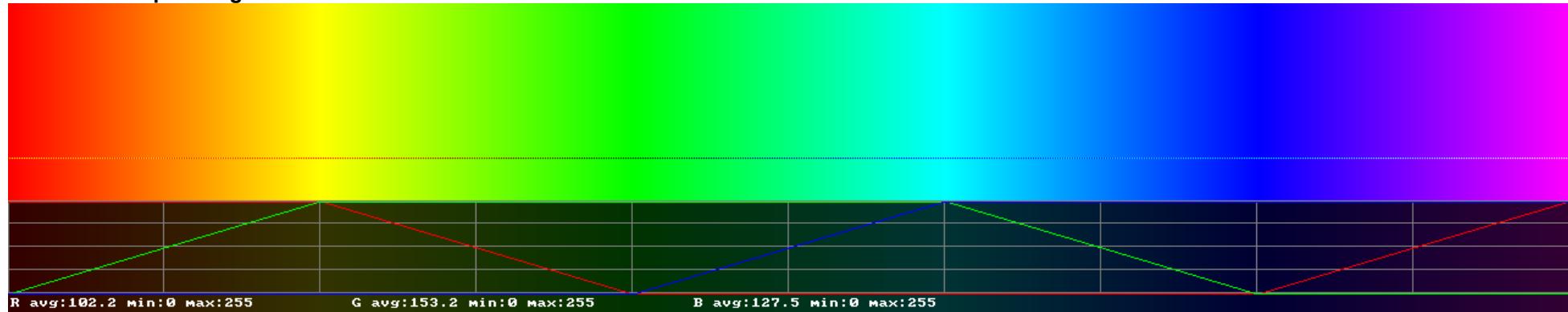
This is the output image:



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.

```
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-512,0,255)+gte(X,1024)*clip(1536-X,0,255)',oscilloscope=tw=1:s=1 -frames 1 -y spectrum2.png  
pause
```

This is the output image:



The same thing can also be done with the "gradients" video source:

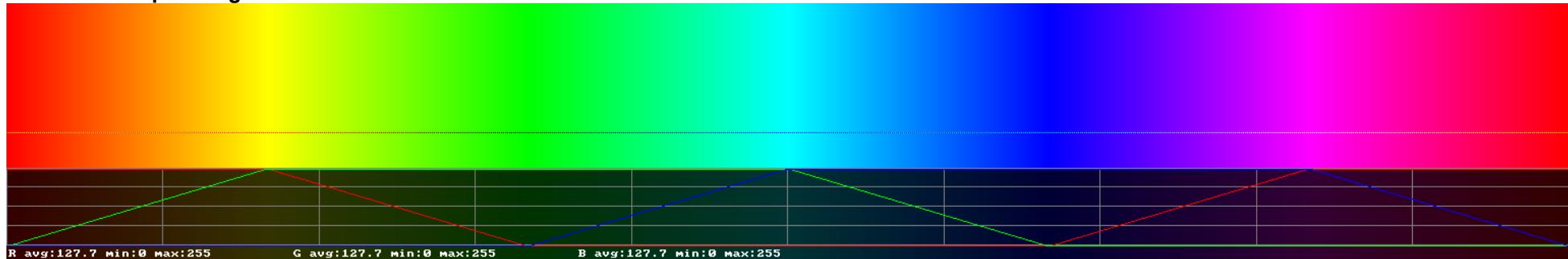
```
ffmpeg -f lavfi -i  
gradients=s=1280x256:c0=red:c1=yellow:c2=lime:c3=cyan:c4=blue:c5=magenta:nb_colors=6:x0=0:y0=0:x1=1279:y1=0 -vf  
oscilloscope=tw=1:s=1 -frames 1 -y spectrum2.png  
pause
```

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.

```
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  
pause
```

This is the output image:



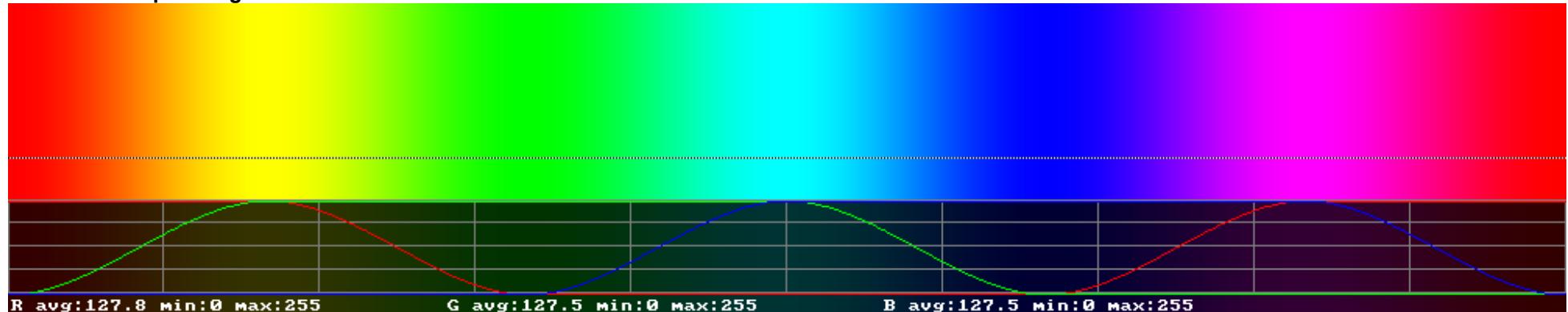
The same spectrum can also be created with the "gradients" video source:

```
ffmpeg -f lavfi -i  
gradients=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 spectrum3.png  
pause
```

A similar spectrum can be created with the "colorspectrum" video source. The difference is that the RGB channels have sine shapes, giving a better result:

```
ffmpeg -f lavfi -i colorspectrum=size=1000x2 -vf format=rgb24,crop=iw:1,scale=iw:50 -frames 1 -y out.png  
pause
```

This is the output image:



Create a full saturation spectrum at the bottom with a gradient to black at the top:

```
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  
pause
```

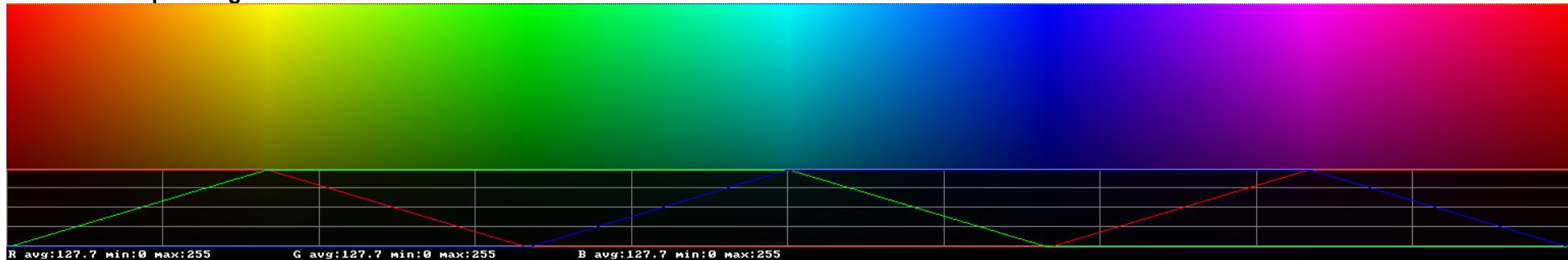
This is the output image:



Create a full saturation spectrum at the top with a gradient to black at the bottom:

```
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':g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));(255-Y)*ld(0)/255':b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));(255-Y)*ld(0)/255', oscilloscope=tw=1:y=0:s=1 -frames 1 -y spectrum5.png  
pause
```

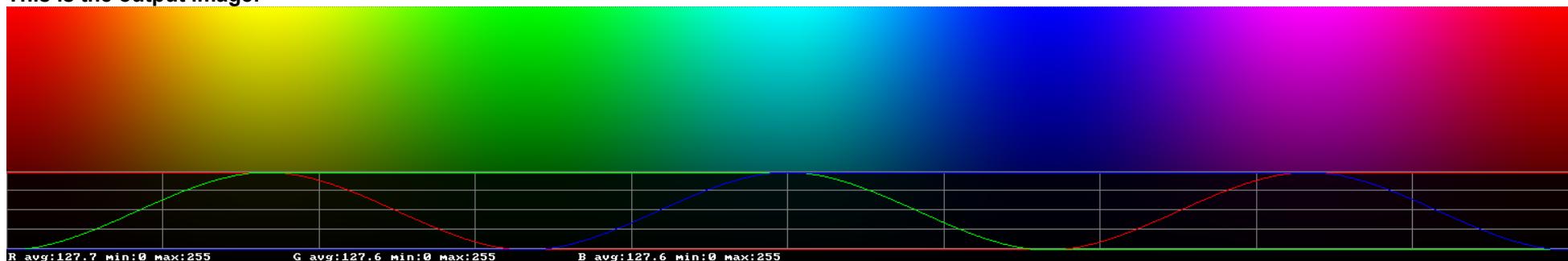
This is the output image:



A similar spectrum can be created with the "colorspectrum" source. The difference is that the RGB channels have sine shapes, giving a better result:

```
ffmpeg -f lavfi -i colorspectrum=size=1536x256:type=black -vf oscilloscope=tw=1:y=0:s=1 -frames 1 -y spectrum5a.png
```

This is the output image:



Create a full saturation spectrum at the top with a gradient to white at the bottom:

```
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)*(1-Y/255)':g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));Y+ld(0)*(1-Y/255)':b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));Y+ld(0)*(1-Y/255)' -frames 1 -y spectrum6.png  
pause
```

This is the output image:



Create a full saturation spectrum at the bottom with a gradient to white at the top:

```
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)/255-1)':g='st(0,lt(X,512)*clip(X,0,255)+gte(X,512)*clip(1024-X,0,255));255+Y*(1d(0)/255-1)':b='st(0,lt(X,1024)*clip(X-512,0,255)+gte(X,1024)*clip(1536-X,0,255));255+Y*(1d(0)/255-1)' -frames 1 -y spectrum7.png  
pause
```

This is the output image:



A similar spectrum can be created with the "colorspectrum" source. The difference is that the RGB channels have sine shapes, giving a better result:

```
ffmpeg -f lavfi -i colorspectrum=size=1536x256:type=white -frames 1 -y spectrum7a.png
```

This is the output image:



Create a full saturation spectrum in the middle with a gradient to white at the top and a gradient to black at the bottom:

```
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 spectrum8.png  
pause
```

This is the output image:



A similar spectrum can be created with the "colorspectrum" source. The difference is that the RGB channels have sine shapes, giving a better result:

```
ffmpeg -f lavfi -i colorspectrum=size=1536x512:type=all -frames 1 -y spectrum8a.png
```

This is the output image:



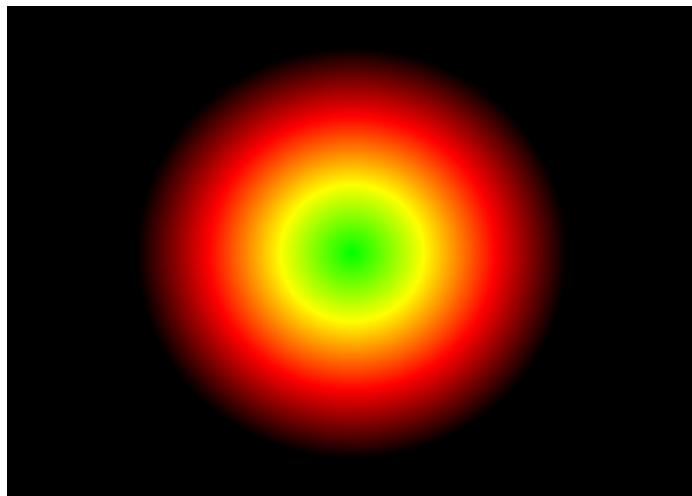
2.165 Radial Gradients

Example:

```
set "N=4"      :: Number of colors
set "C0=lime"   :: Color 0
set "C1=yellow" :: Color 1
set "C2=red"    :: Color 2
set "C3=black"  :: Color 3
set "X0=320"    :: Start X
set "Y0=240"    :: Start Y
set "X1=160"    :: End X
set "Y1=120"    :: End Y

ffmpeg -f lavfi -i gradients=n=%N%:c0=%C0%:c1=%C1%:c2=%C2%:c3=%C3%:type=radial:x0=%X0%:y0=%Y0%:x1=%X1%:y1=%Y1% -frames 1
-y out.png

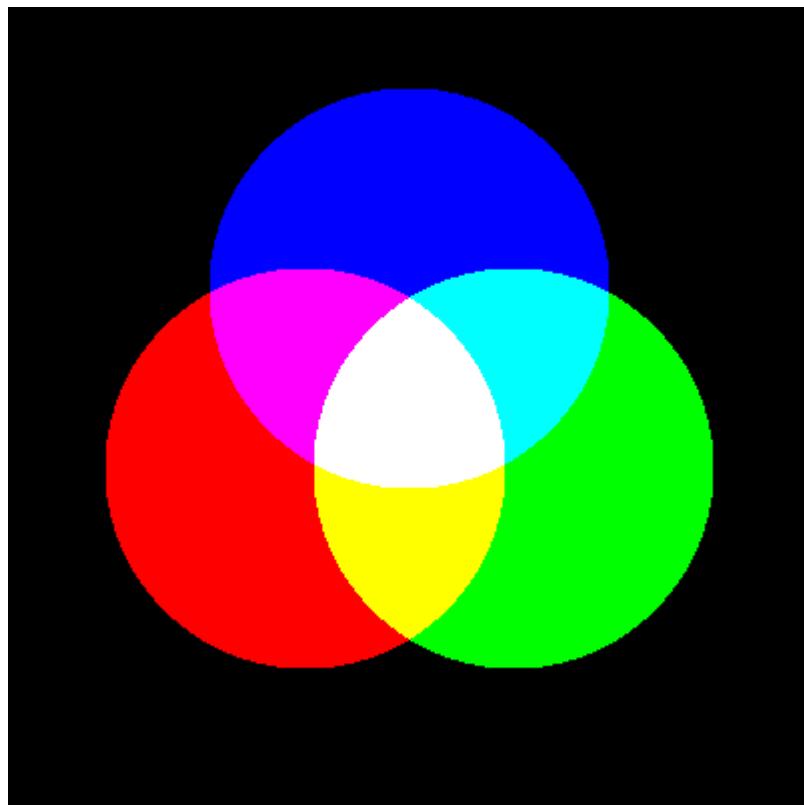
pause
```



2.166 Make an additive color mixing image

```
ffmpeg -f lavfi -i color=black:s=400x400 -lavfi geq=r='255*lt(hypot(X-148,Y-230),100)':g='255*lt(hypot(X-252,Y-230),100)':b='255*lt(hypot(X-200,Y-140),100)' -frames 1 -y test.png  
pause
```

This is the output image:



2.167 Make a test image with shifted RGB channels

```
ffmpeg -f lavfi -i color=black:s=400x160,format=rgb24 -lavfi drawtext=text="Test":font="arial  
black":fontcolor=white:fontsize=140:x=20:y=25,rgbashift=gh=20:bh=10:bv=10 -frames 1 -y rgb_text.png  
  
pause
```

This is the output image:

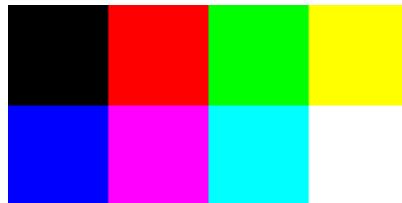


2.168 Make a 8-color test image

This test image can be used for spherical transformations of equirectangular images.

```
ffmpeg -f lavfi -i color=black:s=200x100 -lavfi  
geq=r='255*mod(trunc(4*x/W),2)':g='255*trunc(2*x/W)':b='255*trunc(2*y/H)' -frames 1 -y test.png  
pause
```

This is the output image:



2.169 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

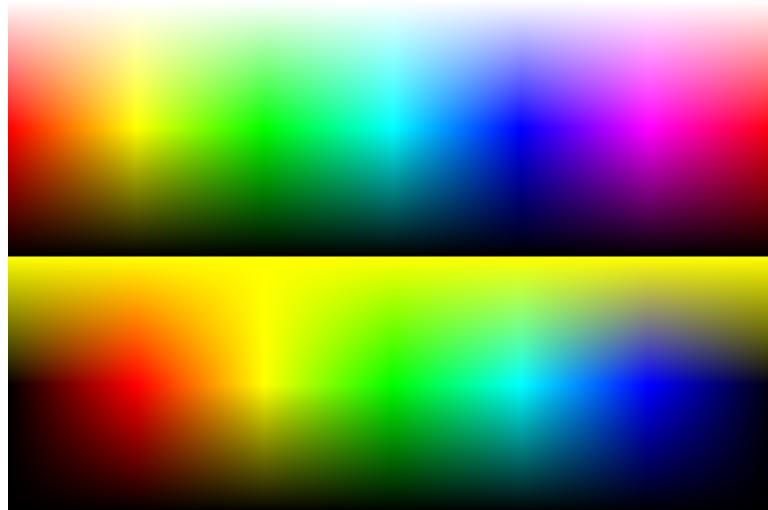
```
LUT_3D_SIZE 2
# black remains black
0 0 0
# red becomes black
0 0 0
# green becomes yellow
1 1 0
# yellow becomes red
1 0 0
# blue becomes cyan
0 1 1
# magenta becomes blue
0 0 1
# cyan becomes green
0 1 0
# white becomes yellow
1 1 0
```

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
```

This is the output image:



Some notes for redshift in astronomy:

$$\Delta \lambda = \lambda * v / c \quad z = (\lambda_{\text{obs}} / \lambda_{\text{emission}}) - 1 \quad z = v / c \quad (\text{approximation for } v \ll c)$$

Note: Redshift could also be realized with the "colormap" filter.

2.170 SMPTE Color Bars

Example:

```
ffmpeg -f lavfi -i smptebars=duration=10 -frames 1 -y out.png  
pause
```

See also: https://en.wikipedia.org/wiki/SMPTE_color_bars

2.171 Make many JPG test images

```
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  
pause
```

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.

2.172 Make a grid video

```
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
pause
```

Note: If thickness is almost as large as grid size, the result becomes a grid of dots.

2.173 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
pause
```

2.174 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
pause
```

2.175 Make a noise video

This is an example for a noise generator:

```
ffmpeg -f lavfi -i color=gray:size=vga -vf noise=c0s=100:allf=t+u -t 10 -y out.mp4  
pause
```

2.176 Make a grayscale test image

```
rem Make an image with 16x16=256 gray levels:  
ffmpeg -f lavfi -i nullsrc -vf geq=lum='X/W*16+16*floor(Y/H*16)':cr=128:cb=128 -frames 1 -y grayscale.png  
pause
```

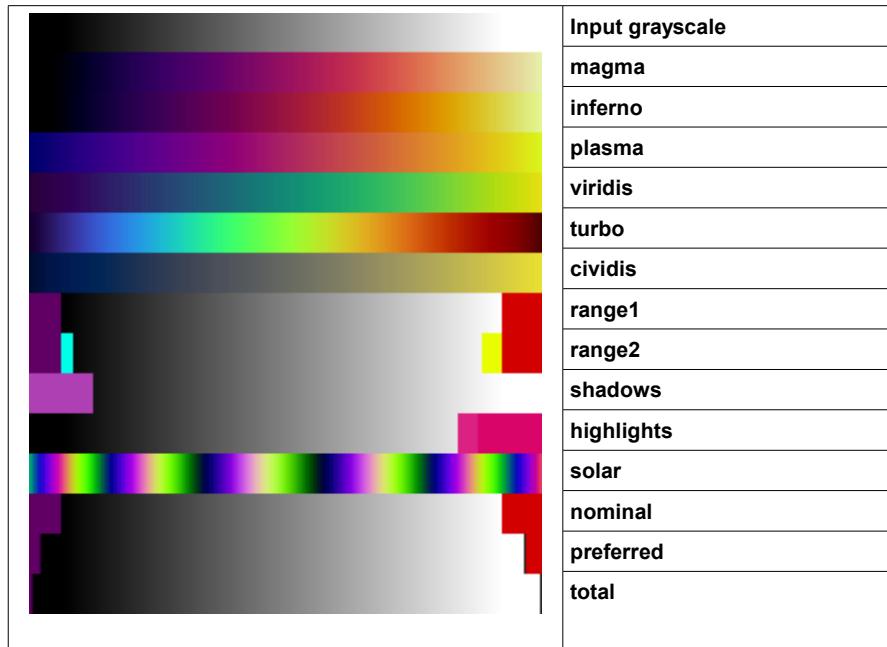
2.177 Pseudocolor filter

This filter can be used for colorizing of black and white images or videos. It has 15 default palettes:

```
ffmpeg -f lavfi -i color=black:s=256x20 -lavfi geq=lum=X:cr=128,split=15[a][b][c][d][e][f][g][h][i][j][k][l][m][n][o];  
[b]pseudocolor=p=magma[B];[c]pseudocolor=p=inferno[C];[d]pseudocolor=p=plasma[D];[e]pseudocolor=p=viridis[E];  
[f]pseudocolor=p=turbo[F];[g]pseudocolor=p=cividis[G];[h]pseudocolor=p=range1[H];[i]pseudocolor=p=range2[I];  
[j]pseudocolor=p=shadows[J];[k]pseudocolor=p=highlights[K];[l]pseudocolor=p=solar[L];[m]pseudocolor=p=nominal[M];  
[n]pseudocolor=p=preferred[N];[o]pseudocolor=p=total[O];[a][B][C][D][E][F][G][H][I][J][K][L][M][N][O]vstack=15 -frames 1  
-y out.png
```

```
pause
```

This is the output image:



2.178 Test a filtergraph for 10-bit compatibility

Sometimes it's important to know if a filtergraph has 10-bit accuracy, or if there is a hidden 8-bit conversion somewhere in the filtergraph.

This can be tested with an image which contains 1024 levels of gray in a 32x32 grid:

```
rem Step 1: Make a 10-bit image with 32*32=1024 levels of gray:  
  
ffmpeg -f lavfi -i nullsrc=s=svga,format=yuv444p10le -vf geq=lum='X/W*32+32*floor(Y/H*32)':cr=512:cb=512,format=rgb48be  
-frames 1 -y 10bit.png  
  
rem Step 2: Apply the filter under test (replace "null" by the filtergraph that you want to test):  
ffmpeg -i 10bit.png -lavfi null -y test.png  
  
rem Step 3: Use strong contrast enhancement by a factor 32 to make the least significant bits visible:  
ffmpeg -i test.png -vf format=yuv444p10le,geq=lum='clip(32*(lum(X,Y)-512),0,1023)':cr='cr(X,Y)':cb='cb(X,Y)' -y 32x.png  
pause
```

Now check if the output image has 32 levels of gray in X direction. If there was a hidden 8-bit conversion, then only 8 levels of gray are visible.

Note: The strong contrast enhancement in the 3rd step is done with the "geq" filter, because the "eq" filter isn't 10-bit compatible.

If you want to do the same test with a video instead of an image, use this batch file:

```
rem Step 1: Make a 10-bit video with 32*32=1024 levels of gray:  
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 0 -t 5 -y  
10bit.mov  
  
rem Step 2: Apply the filter under test (replace "null" by the filtergraph that you want to test):  
ffmpeg -i 10bit.mov -lavfi null -crf 0 -y test.mov  
  
rem Step 3: Use strong contrast enhancement by a factor 32 to make the least significant bits visible:  
ffmpeg -i test.mov -vf geq=lum='clip(32*(lum(X,Y)-512),0,1023)':cr='cr(X,Y)':cb='cb(X,Y)' -crf 0 -y 32x.mov  
pause
```

Here are the results of the 10-bit compatibility test (tested 2022-01-01, sorted alphabetically):

Filter	Is it 10-bit compatible?
atadenoise	yes
blend	yes
bm3d	yes
chromahold	yes
chromakey	yes
chromanr	yes
chromashift	yes
colorbalance	yes
colorchannelmixer	yes
colorcontrast	yes
colorcorrect	yes

colorhold	no
colorkey	no
colortemperature	yes
crop	yes
dcf dnaiz	no
deflicker	yes
deshake	no (dithering is introduced)
despill	no
dilation	yes
displace	no (dithering is introduced)
drawbox	no (dithering is introduced)
eq	no (dithering is introduced)
extractplanes	yes
fftdnaiz	yes
fftfilt	yes
geq	yes
gradfun	no
hqdn3d	yes
hstack	yes
hsvhold	yes
hsvkey	yes
hue	yes
huesaturation	yes
lagfun	yes
lenscorrection	yes
lumakey	yes

monochrome	yes
negate	yes
nlmeans	no
null	yes
owdenoise	yes
perspective	no
remap	yes
removegrain	no
rgbashift	yes
rotate	no
scale	yes
selectivecolor	yes
smartblur	no
swapuv	yes
tmedian	yes
tmix	yes
unsharp	yes
v360	yes
vaguedenoiser	yes
vibrance	yes
vignette	no
vstack	yes

2.179 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  
pause
```

2.180 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.

```
ffmpeg -i test1.png -vf find_rect=test2.pgm,cover_rect out.png  
pause
```

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" and "cover_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.
- If you want to pixelize the region, use the "pixelize" filter.

2.181 Feedback filter

The feedback filter is useful if another filter shall be applied only to a small region of the frame.

```
ffmpeg -i test.jpg -lavfi "[0][fb_in]feedback=x=2000:y=1000:w=2000:h=2000[out][fb_out];[fb_out]gblur=50[fb_in]" -map "[out]" -y out.jpg  
pause
```

2.182 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 and the score to a file. This works only with FFprobe and not with FFmpeg:

```
set "IN=in.mp4"          :: Input video
set "OBJ=needle.pgm"     :: Image of the object, must be gray8
set "TH=0.04"             :: 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,lavfi.rect.score -of csv 1> log.csv

pause
```

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,0.041667,1633,1000,0.032153
frame,0.083333,1634,1000,0.033357
frame,0.125000
frame,0.166667,1634,1000,0.035687
frame,0.208333,1634,1000,0.038733
frame,0.250000
frame,0.291667,1634,1000,0.019203
frame,0.333333,1634,1000,0.007542
...
```

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

Note: Tracking of objects works better in DaVinci resolve.

2.183 Detect black frames and replace them by previous frame

See <https://video.stackexchange.com/q/23589/>

See "blackframe" and "blackdetect" filters.

2.184 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>

Image Format	compressed?	lossless?	16-Bit?	Notes
BMP	no	yes	no	very big filesize
DNG	yes	yes	yes	"Adobe Digital Negative" format, recommended for saving RAW images from cameras. Use Adobe DNG converter to make these files. Warning: FFmpeg's DNG decoder doesn't work correctly with most images.
FITS	yes	yes	yes	Flexible Image Transport System, a popular image format in astronomy
GIF	yes	yes	no	this is obsolete, use PNG instead
JPG	yes	no	no	recommended for 8-bit images if small file size is required
PAM	no	?	?	"Portable Arbitrary Map"
PFM	no	yes	no, 32-bit float	"Portable Float Map" Pixel format is "grayf32"
PGM	no	yes	yes	"Portable Graymap", 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.
PHM	no	yes, but limited precision	yes, float	"Portable Half Float Map" (1 sign bit, 5 exponent bits, 10 fraction bits), see https://en.wikipedia.org/wiki/Half-precision_floating-point_format
PGMYUV	no	yes	?	This is a FFmpeg variant of the binary PGM format
PNG	yes	yes	yes	recommended for lossless saving of 8-bit or 16-bit images
PPM	no	yes	yes	"Portable Pixmap", FFmpeg can read binary PPM (P6) or ASCII PPM (P3) files, but for output only binary PPM (P6) is supported.
TGA	yes	yes	no	this is obsolete, use PNG instead
TIFF	no	yes	yes	

Show all supported pixel formats of the PNG encoder:

```
ffmpeg -h encoder=png  
pause
```

2.185 Image size limits

In FFmpeg the image size is limited to $((Width * 8) + 1024) * (Height + 128) < \text{INT_MAX} = 2147483647$

That means 16384 x 16384 is allowed but 32768 x 16384 is not.

See also <https://video.stackexchange.com/questions/28408/how-to-fix-ffmpeg-picture-size-32768x16384-is-invalid>

2.186 CR2 and DNG Images

FFmpeg's DNG decoder has many problems and fails with most DNG images. Also FFmpeg can't import CR2 (Canon RAW) images.

This is the best workaround I could find so far, for FFmpeg DNG to JPG conversion. The DNG image comes from a Canon 6D via Adobe DNG converter V12.4: **Update December 2021: This command line doesn't work any more.**

```
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  
  
pause
```

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.

Note: This workaround doesn't work with DNG images that were exported from Adobe Lightroom.

Recommended workarounds for importing DNG images in FFmpeg:

- If 8-bit is sufficient, use IrfanView to convert the DNG images to PNG or JPG. Batch processing is possible for groups of images.
- If 16-bit is required, use GIMP with Darktable plugin to convert DNG images to 16-bit PNG. Unfortunately batch processing for groups of images isn't possible, because the BIMP plugin doesn't allow DNG as import format.
- To convert groups of CR2 or DNG images to 16-bit PNG, you can use Darktable (as a stand-alone program without GIMP).
- Don't try to import DNG images in FFmpeg. It might theoretically be possible, however there are too many problems and it's not worth the trouble.

Highly recommended workaround for importing CR2 or DNG images (or groups of numbered images) into a video:

- It's very easy to do this with DaVinci Resolve 17.

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 hints were given in the FFmpeg user list:

- "Use zscale filter."

Sources:

<http://ffmpeg.org/pipermail/ffmpeg-user/2020-October/050340.html>

<http://ffmpeg.org/pipermail/ffmpeg-user/2020-October/050415.html>

<http://ffmpeg.org/pipermail/ffmpeg-user/2020-November/050701.html>

It's unclear how zscale must be used. Which options, with which other filters and/or format conversions, in which order? Unfortunately the documentation of FFmpeg's zscale filter is a useless mess.

The "zimg" (zscale) library can be found here, but don't expect any useful documentation: <https://github.com/sekrit-twc/zimg>

An example for zscale can be found here: <https://stackoverflow.com/questions/69251960/how-can-i-encode-rgb-images-into-hdr10-videos-in-ffmpeg-command-line>

- "Use the right input pixel format prior to calling zscale as it does not support bayer formats."

Source: <http://ffmpeg.org/pipermail/ffmpeg-user/2020-October/050428.html>

It's unclear 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."

Source: <http://ffmpeg.org/pipermail/ffmpeg-user/2020-October/050428.html>

"trc" means "transformation characteristics". 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?

- The following threads in the ffmpeg-user list are about importing DNG images in FFmpeg, but none of them contains a working example:

- <http://ffmpeg.org/pipermail/ffmpeg-user/2013-February/013312.html>
- <http://ffmpeg.org/pipermail/ffmpeg-user/2020-August/049681.html>
- <http://ffmpeg.org/pipermail/ffmpeg-user/2020-October/050323.html>
- <http://ffmpeg.org/pipermail/ffmpeg-user/2020-October/050412.html>
- <http://ffmpeg.org/pipermail/ffmpeg-user/2020-November/050700.html>
- <http://ffmpeg.org/pipermail/ffmpeg-user/2021-February/051999.html>

I gave 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. If anybody has found a working example, please let me know.

I used these DNG images for testing:

http://www.astro-electronic.de/IMG_3459.dng This image was taken with a Canon 6D camera and converted to DNG by Adobe DNG converter V12.4

http://www.astro-electronic.de/Canon_5D.dng This image was taken with a Canon 5D-MK4 camera and converted to DNG by Adobe DNG converter V12.4

http://www.astro-electronic.de/Pentax_K5.DNG This image was directly generated by a Pentax K5 camera.

Some other DNG images can be found here: <http://ffmpeg.org/pipermail/ffmpeg-user/2020-August/049681.html>

A DNG test image can be downloaded near the bottom of this website: <https://www.rawdigger.com/howtouse/rawdigger-histograms-display-modes>

2.187 Colorspace

For more infos about colorspace, colormatrix and zscale see <https://trac.ffmpeg.org/wiki/colorspace>

This is copied from the above wiki page:

FFmpeg stores all these properties in the AVFrame struct:

- The format (type and bit-depth), in AVFrame->format
- The signal range, in AVFrame->color_range
- The YUV/RGB transformation matrix, in AVFrame->colorspace
- The linearization function (a.k.a. transformation characteristics), in AVFrame->color_trc
- The RGB/XYZ matrix, in AVFrame->color_primaries

See also <https://stackoverflow.com/questions/61834623/how-to-use-ffmpeg-colorspace-options>

See also <https://medium.com/invideo-io/talking-about-colorspaces-and-ffmpeg-f6d0b037cc2f>

2.188 Video sizes

This list contains only the most important video sizes. The full list is in the FFmpeg documentation.

	Size	Aspect ratio	Notes
ntsc	720x480	3:2	
pal	720x576	5:4	
vga	640x480	4:3	
svga	800x600	4:3	
xga	1024x768	4:3	
hd720	1280x720	16:9	
uxga	1600x1200	4:3	
hd1080	1920x1080	16:9	
wuxga	1920x1200	8:5	Beamer in the planetarium in the St. Andreasberg observatory
2kflat	1998x1080	37:20 = 16.65:9	Cinema advertising
2k, 2kdci	2048x1080	256:135 = 17.066:9	
uhd2160	3840x2160	16:9	
4kflat	3996x2160	37:20 = 16.65:9	
4k, 4kdci	4096x2160	256:135 = 17.066:9	

	128	160	176	240	320	352	400	432	480	640	704	720	768	800	852	960
96	sqcif															
120		qqvga														
144			qcif													
160				hqvga												
200					cga											
240					qvga	qntsc, film, ntsc-film	wqvga	fwqvga								
288						qpal, cif										
320									hvga							
350										ega						
360										nhd						
480										sntsc, vga	ntsc			wvga, hd480		
540																qhd
576										4cif	pal	spal				
600													svga			

	1024	1280	1366	1408	1600	1920	1998	2048	2560	3200	3840	3996	4096	5120	6400	7680
720		hd720														
768	xga		wxga													
858								2kscope								
1152				16cif												
1024		sxga			wsxga											
1080						hd1080	2kflat	2k, 2kdci								
1200					uxga	wuxga										
1536								qxga								
1600									woxga							
1716												4kscope				
2048									gsxga	wqsxga						
2160											uhd2160	4kflat	4k, 4kdci			
2400											wquxga					
4096													hsxga	whsxga		
4320															uhd4320	
4800																whuxga

Note: WQXGA = 2716x1528

2.189 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=3"                :: MP4 quality level, 1 is best quality, 3 is normal, 31 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)
set "CR=2280:2280:88:88"      :: 180° field of view: 2104:2104:176:176
                                :: 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.

2.190 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.

```
rem Manipulate a video from KODAK SP360_4K camera (size 2880x2880)

set "SIZE=1200"          :: Video size (square)
set "QU=3"                :: MP4 quality level, 1 is best quality, 3 is normal, 31 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)
set "CROP=2728:2728:74:74"  :: 180° field of view: 2456:2456:210:210
                           :: 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
```

2.191 Postprocessing of real time videos of the night sky

```
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

rem 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

rem 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=all_mode=subtract,eq=brightness=%BR%,split=2[b][c];[b]atadenoise=0a=%D%:1a=%D%:2a=%D%:0b=%D%:1b=%D%:2b=%D%[d];[c]tmix=%TMIX%[e];[d][e]blend=all_expr='max(%NR%*A,B)'" -q:v 1 -y %OUT%
pause
```

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
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=all_mode=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=all_mode=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:

```
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

set AUD=""
if %AUDIO%==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=all_mode=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=all_mode=lighten:shortest=1" -q:v 1 -c:v mpeg4 %AUD% -y %~n1.mp4

exit /b

:the_end

pause
```

2.192 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:

```
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 "[0]format=pix_fmts=rgb48be[a];[1]format=pix_fmts=rgb48be[b];[b][a]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:

```
set "IN=P1000128.mov"          :: Input video
set "BR=0.04"                   :: Small brightness adjustment before applying the CLUT
set "OUT=128.mp4"              :: Output video

ffmpeg -i processed_image_with_clut.png -vf crop=512:512:0:0 -y clut.png

ffmpeg -i %IN% -i clut.png -filter_complex "[0]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
```

2.193 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.

```
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=3"                     :: MP4 Quality, 1 is best Quality, 3 is normal, 31 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
```

2.194 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:

```
set "IN=IMG_%%4d.jpg"      :: Input pictures
set "SN=3551"               :: Number of first picture
set "SIZE=1200"              :: Video size (square)
set "QU=3"                   :: MP4 quality level, 1 is best quality, 3 is normal, 31 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%, ^
overlay,zoompan=s=%SIZE%x%SIZE%:d=%D%+%E%*`eq(in,1)`+%E%*`eq(in,%L%)`:fps=%F%, ^
framerate=%FPS%:interp_start=0:interp_end=255:scene=100,rotate=%ROT%,deflicker=size=%DEF%, ^
fade=in:d=%FADE%,fade=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
```

2.195 Slow motion with Panasonic GH5S at 240fps

Set "Rec Format" to "MOV" and "Rec Quality" to one of these:

System Frequency	Rec Quality	Available Framerates
59.94Hz (NTSC)	[FHD/8bit/100M/60p]	2 30 56 58 60 62 64 90 120 150 180 210 240
	[FHD/8bit/100M/30p]	2 15 26 28 30 32 34 45 60 75 90 105 120 135 150 165 180 195 210 225 240
	[FHD/8bit/100M/24p]	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
50.00Hz (PAL)	[FHD/8bit/100M/50p]	2 25 46 48 50 52 54 75 100 125 150 200 240
	[FHD/8bit/100M/25p]	2 12 21 23 25 27 30 37 50 62 75 87 100 112 125 137 150 175 200 225 240
24.00Hz (CINEMA)	[FHD/8bit/100M/24p]	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

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:

Speed_Factor = Framerate / 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:

```
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 = 50p, VFRRatio = 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.

Table for setpts value (for videos taken at 240fps):

Base Framerate	Desired Slow Motion Factor													
	2x	2.4x	2.5x	3x	4x	4.8x	5x	6x	8x	9.6x	10x	12x	16x	19.2x
24	0.2	0.24	0.25	0.3	0.4	0.48	0.5	0.6	0.8	0.96	1	1.2	1.6	1.92
25	0.20833	0.25	0.26042	0.3125	0.41667	0.5	0.52083	0.625	0.8333	1	1.0417	1.25	1.6667	2
30	0.25	0.3	0.3125	0.375	0.5	0.6	0.625	0.75	1	1.2	1.25	1.5	2	2.4
50	0.41667	0.5	0.52208	0.625	0.83333	1	1.0417	1.3021	1.6667	2	2.0833	2.5	3.3333	4
60	0.5	0.6	0.625	0.75	1	1.2	1.25	1.5	2	2.4	2.5	3	4	4.8
Output fps	120	100	96	80	60	50	48	40	30	25	24	20	15	12.5

Base Framerate	Desired Slow Motion Factor													
	20x	24x	25x	30x	40x	48x	50x	60x	80x	96x	100x	120x	160x	192x
24	2	2.4	2.5	3	0.4	4.8	5	6	8	9.6	10	12	16	19.2
25	2.0833	2.5	2.6042	3.125	4.1667	5	5.2083	6.25	8.3333	10	10.417	12.5	16.667	20
30	2.5	3	3.125	3.75	5	6	6.25	7.5	10	12	12.5	15	20	24
50	4.1667	5	5.2083	6.25	8.3333	10	10.417	13.021	16.667	20	20.833	25	33.333	40
60	5	6	6.25	7.5	10	12	12.5	15	20	24	25	30	40	48
Output fps	12	10	9.6	8	6	5	4.8	4	3	2.5	2.4	2	1.5	1.25

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.

2.196 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%

pause

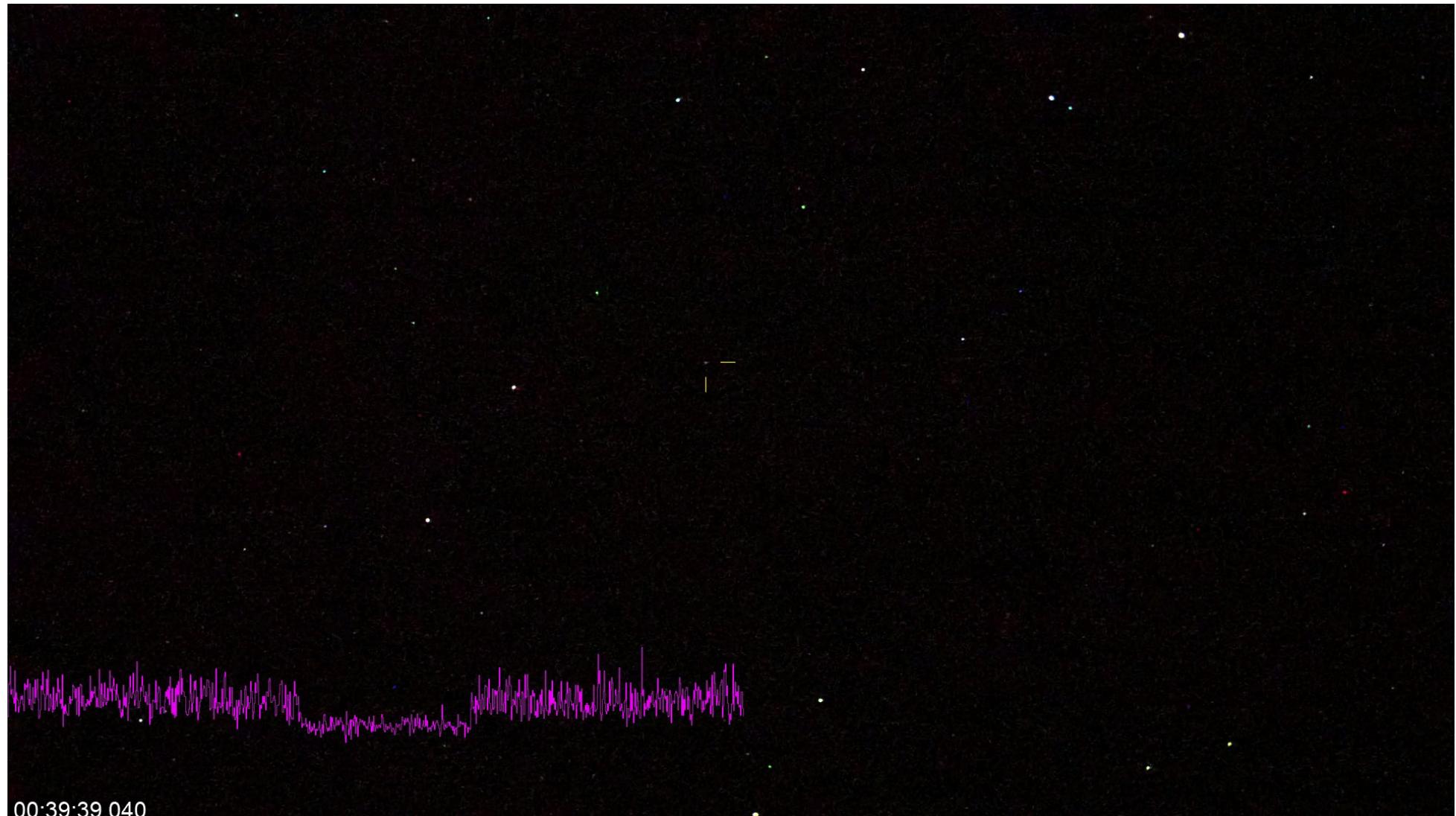
```

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 0xAABBGGRR, 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:

```
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=0xfffff00ff"          :: 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%" -frames 1 -y frame1.png

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:
```

```
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%  
  
pause
```

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.

```
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 "C1=0xffffffffffff"     :: Color for star curve, in format 0xAABBGGRR  
set "C2=0xfffff00fff"       :: 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]asetnsamples=n=1920,astats=metadata=1:reset=1,adrawgraph=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=1:h=2*B[4];[4][3]overlay=y=750,%CLOCK% -pix_fmt yuv420p -t %T% -y %OUT%

pause

```

The value 1920 for asetnsamples is the number of audio samples per video frame, in this case $48000 / 25 = 1920$.

2.197 Oscilloscope

The oscilloscope filter can show the brightness of a video line:

```
set "IN=P1000479.mov"      :: Input video
set "OUT=out.mp4"          :: Output video
::
set "LINE=495"              :: The shown line
set "H=1080"                :: Height of the video

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.

2.198 Vectorscope

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

ffplay %IN% -vf "split[a][b],[b]vectorscope=g=green[c],[a][c]overlay=x=W-w:y=H-h"
```

2.199 Capture a video from a webcam

List all supported, connected capture devices:

```
ffmpeg -list_devices 1 -f dshow -i dummy  
pause
```

List the possible video sizes, frame rates and pixel formats for one capture device:

```
ffmpeg -list_options 1 -f dshow -i video="HD WebCam"  
pause
```

Capture video from the webcam:

```
ffmpeg -y -f dshow -video_size 1280x720 -framerate 10 -pixel_format yuyv422 -i video="HD WebCam" -t 5 out.mp4  
pause
```

See also: <https://trac.ffmpeg.org/wiki/Capture/Webcam>

See also: <https://trac.ffmpeg.org/wiki/DirectShow>

A possible capture device is the "OBS Virtual Camera", example:

```
ffmpeg -rtbufsize 500M -f dshow -i video="OBS Virtual Camera" -t 2 -y out.mp4
```

Built-in help for "dshow" input device:

```
ffmpeg -h demuxer=dshow  
pause
```

2.200 Capture video from the desktop or from a window

Capture the entire desktop without audio:

```
set "FR=10"          :: Framerate  
  
ffmpeg -f gdigrab -framerate %FR% -i desktop -y out.mp4  
  
pause
```

Capture the entire desktop with audio:

Go to the windows device manager and choose "Audio Devices". Select the "Recording" tab. Enable the "Stereomix" device. Click on "Properties". Under the "Listen" tab, check the box "Listen to this device". Under the "Levels" tab, disable the speaker.
Then run this command to find out the name of the "Stereomix" device:

```
ffmpeg -list_devices true -f dshow -i dummy  
  
pause
```

Enter the name in the following command line, in this case "Stereomix (Realtek(R) Audio)". If the audio volume is too low, you can increase it with the "volume" filter.

```
ffmpeg -f gdigrab -i desktop -f dshow -i audio="Stereomix (Realtek(R) Audio)" -af volume=3 -y grabbed_video.mp4  
  
pause
```

Or alternatively with Nvidia driver:

```
ffmpeg -f gdigrab -i desktop -f dshow -i audio="Stereomix (Realtek(R) Audio)" -af volume=3 -c:v h264_nvenc -y  
grabbed_video.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 change dynamically. For example, the title of an editor changes as soon as you begin to enter something (a * is inserted at the beginning).

2.201 Capture video and audio from a HDMI to USB converter

It's also possible 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:

```
ffmpeg -list_options 1 -f dshow -i video="USB Video"  
pause
```

This is the output:

```
[dshow @ 00000000000184800] DirectShow video device options (from video devices)  
[dshow @ 00000000000184800] Pin "Capture" (alternative pin name "0")  
[dshow @ 00000000000184800] vcodec=mjpeg min s=1920x1080 fps=5 max s=1920x1080 fps=30  
[dshow @ 00000000000184800] vcodec=mjpeg min s=1600x1200 fps=5 max s=1600x1200 fps=30  
[dshow @ 00000000000184800] vcodec=mjpeg min s=1360x768 fps=5 max s=1360x768 fps=30  
[dshow @ 00000000000184800] vcodec=mjpeg min s=1280x1024 fps=5 max s=1280x1024 fps=30  
[dshow @ 00000000000184800] vcodec=mjpeg min s=1280x960 fps=5 max s=1280x960 fps=50  
[dshow @ 00000000000184800] 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.

The available audio options of the HDMI to USB converter can be shown with this command:

```
ffmpeg -list_options 1 -f dshow -i audio="Digitale Audioschnittstelle (US"
pause
```

This is the output:

```
[dshow @ 00000000003c48c0] DirectShow audio only device options (from audio devices)
[dshow @ 00000000003c48c0] Pin "Capture" (alternative pin name "Capture")
[dshow @ 00000000003c48c0] 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):

```
ffmpeg -f dshow -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
pause
```

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:

```
ffmpeg -f dshow -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  
  
pause
```

The same can also be done without filtering, by using the "-sar" option:

```
ffmpeg -f dshow -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  
  
pause
```

Note: If the size 1600x1200 is used, the sample aspect ratio is 4/3. That means the pixels are wider than high and the input size is 16:9.

Case study: We want to capture a square 1:1 region from the center of the field (for example a fisheye), coming from GH5S camera. Is it better to set the camera to FHD (16:9) or anamorphic (4:3) format?

FHD or 4K (16:9)	Anamorphic (4:3)
Output pixels have 1:1 aspect ratio: <pre>ffmpeg -f dshow -video_size 1920x1080 -framerate 5 -pixel_format yuyv422 -i video="USB Video" -lavfi "crop=1200:1200,format=rgb24" -f sdl2 -</pre>	Output pixels have 1:1 aspect ratio: <pre>ffmpeg -f dshow -video_size 1600x1200 -framerate 5 -pixel_format yuyv422 -i video="USB Video" -lavfi "crop=900:1200,format=rgb24,scale=1200:1200" -f sdl2 -</pre> Output pixels have 4:3 aspect ratio: <pre>ffmpeg -f dshow -video_size 1600x1200 -framerate 5 -pixel_format yuyv422 -i video="USB Video" -lavfi "setsar=4/3,format=rgb24,crop=900:1200" -f sdl2 -</pre>
Horizontal resolution: 1080	Horizontal resolution: 900
Vertical resolution: 1080	Vertical resolution: 1200

This is the batch file for capturing video and audio from the HDMI to USB converter, using mjpeg compression which allows higher framerates (tested with GH5S):

```
ffmpeg -f dshow -rtbufsize 500M -video_size 1920x1080 -framerate 30 -vcodec mjpeg -i video="USB Video":audio="Digitale  
Audioschnittstelle (US" -q:v 2 -c:v mpeg4 -t 10 -y out.mp4  
  
pause
```

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:

```
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  
  
pause
```

This is the same without filtering, with "-sar" option instead:

```
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  
  
pause
```

If mjpeg compression is used, it's possible to avoid re-encoding by using the "-c:v copy" option:

```
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  
  
pause
```

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.

2.202 Adding *.srt subtitles to a video

Create a "SubRip" *.srt subtitle file. The format is as follows and the text must begin in the first line:

```
1
00:00:00,000 --> 00:00:02,000
This is the title text

2
00:00:02,500 --> 00:00:04,000
<b>This is bold text</b>

3
00:00:04,500 --> 00:00:06,000 X1:100 X2:100 Y1:100 Y2:100
<u>This is underlined at position 100,100</u>

4
00:00:06,500 --> 00:00:08,000
<i>italic</i>

5
00:00:08,500 --> 00:00:10,000
<font color="#ffff00">The End (yellow text)</font>
```

Note: I didn't fully understand how the text coordinates are defined (third subtitle in the above example). What's the difference between X1 and X2? It's unclear to which frame size the coordinates are referring to.

For more info about the SubRip file format, see also: <https://en.wikipedia.org/wiki/SubRip>

Method 1, burning the subtitles directly into the video frames:

```
ffmpeg -i subtitle.srt -y subtitle.ass
ffmpeg -i test.mp4 -vf ass=subtitle.ass -y out.mp4
pause
```

Method 2, the output seems to be the same as with method 1:

```
ffmpeg -i test.mp4 -vf subtitles=subtitle.srt -y out.mp4  
pause
```

Same thing with specifying the fontsize and yellow color:

```
ffmpeg -i test.mp4 -vf subtitles=subtitle.srt:force_style='Fontsize=26,PrimaryColour=&H00ffff&' -y out.mp4  
pause
```

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  
pause
```

Note: MP4 containers support subtitles only in "MPEG-4 Timed Text" (MP4TT) format. You can force FFmpeg to write this format with the `-c:s mov_text` option.

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>

Convert YouTube SBV to SRT: <https://trac.ffmpeg.org/wiki/HowToConvertYouTubeSBVtoSRT>

If you want to create many subtitles, consider using a subtitler , for example "aegisub": <http://www.aegisub.org/>

2.203 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]
Format: Name, Fontname, Fontsize, PrimaryColour, SecondaryColour, OutlineColour, BackColour, Bold, Italic, Underline, StrikeOut,
ScaleX, ScaleY, Spacing, Angle, BorderStyle, Outline, Shadow, Alignment,
MarginL, MarginR, MarginV, Encoding
Style: testStyle,Verdana,96,&H00A0A0A0,&H00000000,&Hc0000000,-1,0,0,0,100,100,0,0.00,1,1,0,2,0,0,0,0

[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 &HAABBGGRR, 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>

The *.ass file format is explained in detail here: 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%
pause
```

2.204 Removing subtitles

Subtitles can be removed with the **-sn** option, for inputs or for outputs.

2.205 frei0r

This seems to be an external library for video effects.

"Frei0r is mostly adopted on GNU/Linux and OSX platforms, counts more than 100 plugins and is used by several video software applications ..."

See also <https://frei0r.dyne.org/>

See also <https://en.wikipedia.org/wiki/Frei0r>

Some documentation can be found here: https://mltframework.github.io/mlt_web/plugins/PluginsFilters/

2.206 Mapping streams to the output (-map)

For details about the "-map" option, see also: <https://trac.ffmpeg.org/wiki/Map>

2.207 Creating multiple outputs

This is described on this Wiki page: <https://trac.ffmpeg.org/wiki/Creating%20multiple%20outputs>

2.208 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:v:1 copy -c:v:1 png -disposition:v:1 attached_pic -y %OUT%
pause
```

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:v:1 copy -c:v:1 png -disposition:v:1 attached_pic -y %OUT%
pause
```

It's also possible to write all in one command line:

```
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:

```
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
```

Note: When you upload a video with embeded preview image to a Wordpress website, then you have two files in the media pool: The video and the preview image. But the preview image isn't automatically used when you insert a "video" block. You have to edit the properties of the video block and select the preview image. In this case it makes sense to give tge preview image the same size as the video (don't reduce it to width 320).

2.209 Attach Cover Art to a Video

Example:

```
ffmpeg -i "input.mkv" -attach "cover.jpg" -map 0 -c copy -metadata:s:t mimetype="image/jpg" -metadata:s:t:0
filename="cover.jpg" "output_cover.mkv"
```

Source: <http://ffmpeg.org/pipermail/ffmpeg-user/2022-March/054398.html>

2.210 Timeline support

Some filters have timeline support, which means they can be enabled with the "enable=" option.

Example:

```
delogo=200:100:100:50:enable='between(t,10,20)'
```

It's missing in the documentation. You have to use this command to see which filters have timeline support:

```
ffmpeg -filters
pause
```

Here is the result:

```
Filters:
T.. = Timeline support
.S. = Slice threading
..C = Command support
A = Audio input/output
V = Video input/output
N = Dynamic number and/or type of input/output
| = Source or sink filter
... abench          A->A    Benchmark part of a filtergraph.
..C acompressor     A->A    Audio compressor.
... acontrast        A->A    Simple audio dynamic range compression/expansion filter.
... acopy            A->A    Copy the input audio unchanged to the output.
... acue             A->A    Delay filtering to match a cue.
... acrossfade       AA->A   Cross fade two input audio streams.
..S. crossover      A->N    Split audio into per-bands streams.
..C acrusher        A->A    Reduce audio bit resolution.
TS. adeclick        A->A    Remove impulsive noise from input audio.
TS. adeclip         A->A    Remove clipping from input audio.
T.. adelay          A->A    Delay one or more audio channels.
TSC adenorm        A->A    Remedy denormals by adding extremely low-level noise.
... aderivative     A->A    Compute derivative of input audio.
... aecho            A->A    Add echoing to the audio.
TSC aemphasis       A->A    Audio emphasis.
T.. aeval           A->A    Filter audio signal according to a specified expression.
```

T.C aexciter	A->A	Enhance high frequency part of audio.
T.C afade	A->A	Fade in/out input audio.
TSC afftdn	A->A	Denoise audio samples using FFT.
... afftfilt	A->A	Apply arbitrary expressions to samples in frequency domain.
.SC afir	N->N	Apply Finite Impulse Response filter with supplied coefficients in additional stream(s).
... aformat	A->A	Convert the input audio to one of the specified formats.
TSC afreqshift	A->A	Apply frequency shifting to input audio.
T.C agate	A->A	Audio gate.
.S. aiir	A->N	Apply Infinite Impulse Response filter with supplied coefficients.
... aintegral	A->A	Compute integral of input audio.
... ainterleave	N->A	Temporally interleave audio inputs.
... alimiter	A->A	Audio lookahead limiter.
TSC allpass	A->A	Apply a two-pole all-pass filter.
... aloop	A->A	Loop audio samples.
... amerge	N->A	Merge two or more audio streams into a single multi-channel stream.
T.. ametadata	A->A	Manipulate audio frame metadata.
.C amix	N->A	Audio mixing.
... amultiply	AA->A	Multiply two audio streams.
TSC anequalizer	A->N	Apply high-order audio parametric multi band equalizer.
TSC anlmdn	A->A	Reduce broadband noise from stream using Non-Local Means.
.SC anlms	AA->A	Apply Normalized Least-Mean-Squares algorithm to first audio stream.
... anull	A->A	Pass the source unchanged to the output.
T.. apad	A->A	Pad audio with silence.
T.. aperms	A->A	Set permissions for the output audio frame.
... aphaser	A->A	Add a phasing effect to the audio.
TSC aphaseshift	A->A	Apply phase shifting to input audio.
... apulsator	A->A	Audio pulsator.
... arealtime	A->A	Slow down filtering to match realtime.
... aresample	A->A	Resample audio data.
... areverse	A->A	Reverse an audio clip.
TSC arnndn	A->A	Reduce noise from speech using Recurrent Neural Networks.
... aselect	A->N	Select audio frames to pass in output.
... asendcmd	A->A	Send commands to filters.
... asetnsamples	A->A	Set the number of samples for each output audio frames.
... asetpts	A->A	Set PTS for the output audio frame.
... asetrate	A->A	Change the sample rate without altering the data.
... asettbt	A->A	Set timebase for the audio output link.
... ashawinfo	A->A	Show textual information for each audio frame.
T.. asidedata	A->A	Manipulate audio frame side data.
TSC asoftclip	A->A	Audio Soft Clipper.
... asplit	A->N	Pass on the audio input to N audio outputs.
.S. astats	A->A	Show time domain statistics about audio frames.
..C astreamselect	N->N	Select audio streams
TSC asubboost	A->A	Boost subwoofer frequencies.

TSC asubcut	A->A	Cut subwoofer frequencies.
TSC asupercut	A->A	Cut super frequencies.
TSC asuperpass	A->A	Apply high order Butterworth band-pass filter.
TSC asuperstop	A->A	Apply high order Butterworth band-stop filter.
..C atempo	A->A	Adjust audio tempo.
... atrim	A->A	Pick one continuous section from the input, drop the rest.
... axcorrelate	AA->A	Cross-correlate two audio streams.
... azmq	A->A	Receive commands through ZMQ and broker them to filters.
TSC bandpass	A->A	Apply a two-pole Butterworth band-pass filter.
TSC bandreject	A->A	Apply a two-pole Butterworth band-reject filter.
TSC bass	A->A	Boost or cut lower frequencies.
TSC biquad	A->A	Apply a biquad IIR filter with the given coefficients.
... channelmap	A->A	Remap audio channels.
... channelsplit	A->N	Split audio into per-channel streams.
... chorus	A->A	Add a chorus effect to the audio.
... compand	A->A	Compress or expand audio dynamic range.
... compensationdelay	A->A	Audio Compensation Delay Line.
T.C crossfeed	A->A	Apply headphone crossfeed filter.
TSC crystalizer	A->A	Simple audio noise sharpening filter.
T.. dcshift	A->A	Apply a DC shift to the audio.
T.. deesser	A->A	Apply de-essing to the audio.
... drmeter	A->A	Measure audio dynamic range.
T.C dynaudnorm	A->A	Dynamic Audio Normalizer.
... earwax	A->A	Widen the stereo image.
... ebur128	A->N	EBU R128 scanner.
TSC equalizer	A->A	Apply two-pole peaking equalization (EQ) filter.
T.C extrastereo	A->A	Increase difference between stereo audio channels.
..C firequalizer	A->A	Finite Impulse Response Equalizer.
... flanger	A->A	Apply a flanging effect to the audio.
... haas	A->A	Apply Haas Stereo Enhancer.
... hdcd	A->A	Apply High Definition Compatible Digital (HDCD) decoding.
.S. headphone	N->A	Apply headphone binaural spatialization with HRTFs in additional streams.
TSC highpass	A->A	Apply a high-pass filter with 3dB point frequency.
TSC highshelf	A->A	Apply a high shelf filter.
... join	N->A	Join multiple audio streams into multi-channel output.
... loudnorm	A->A	EBU R128 loudness normalization
TSC lowpass	A->A	Apply a low-pass filter with 3dB point frequency.
TSC lowshelf	A->A	Apply a low shelf filter.
... mcompand	A->A	Multiband Compress or expand audio dynamic range.
... pan	A->A	Remix channels with coefficients (panning).
... replaygain	A->A	ReplayGain scanner.
..C rubberband	A->A	Apply time-stretching and pitch-shifting.
..C sidechaincompress	AA->A	Sidechain compressor.
T.C sidechaingate	AA->A	Audio sidechain gate.

... silencedetect	A->A	Detect silence.
... silenceremove	A->A	Remove silence.
.C speechnorm	A->A	Speech Normalizer.
T.C stereotools	A->A	Apply various stereo tools.
T.C stereowiden	A->A	Apply stereo widening effect.
... superequalizer	A->A	Apply 18 band equalization filter.
.S. surround	A->A	Apply audio surround upmix filter.
TSC treble	A->A	Boost or cut upper frequencies.
... tremolo	A->A	Apply tremolo effect.
... vibrato	A->A	Apply vibrato effect.
T.C volume	A->A	Change input volume.
... volumedetect	A->A	Detect audio volume.
... aevalsrc	I->A	Generate an audio signal generated by an expression.
... afirsrc	I->A	Generate a FIR coefficients audio stream.
... anoisefsrc	I->A	Generate a noise audio signal.
... anullsrc	I->A	Null audio source, return empty audio frames.
... hilbert	I->A	Generate a Hilbert transform FIR coefficients.
... sinc	I->A	Generate a sinc kaiser-windowed low-pass, high-pass, band-pass, or band-reject FIR coefficients.
... sine	I->A	Generate sine wave audio signal.
... anullsink	A->I	Do absolutely nothing with the input audio.
... addroi	V->V	Add region of interest to frame.
... alphaextract	V->N	Extract an alpha channel as a grayscale image component.
T.. alphamerge	VV->V	Copy the luma value of the second input into the alpha channel of the first input.
TSC amplify	V->V	Amplify changes between successive video frames.
... ass	V->V	Render ASS subtitles onto input video using the libass library.
TSC atadenoise	V->V	Apply an Adaptive Temporal Averaging Denoiser.
TSC avgblur	V->V	Apply Average Blur filter.
T.C bbox	V->V	Compute bounding box for each frame.
... bench	V->V	Benchmark part of a filtergraph.
T.C bilateral	V->V	Apply Bilateral filter.
T.. bitplanenoise	V->V	Measure bit plane noise.
.S. blackdetect	V->V	Detect video intervals that are (almost) black.
... blackframe	V->V	Detect frames that are (almost) black.
TSC blend	VV->V	Blend two video frames into each other.
TS. bm3d	N->V	Block-Matching 3D denoiser.
T.. boxblur	V->V	Blur the input.
TS. bwdif	V->V	Deinterlace the input image.
TSC cas	V->V	Contrast Adaptive Sharpen.
TSC chromahold	V->V	Turns a certain color range into gray.
TSC chromakey	V->V	Turns a certain color into transparency. Operates on YUV colors.
TSC chromanr	V->V	Reduce chrominance noise.
TSC chromashift	V->V	Shift chroma.
... ciescope	V->V	Video CIE scope.
T.. codecview	V->V	Visualize information about some codecs.

TSC colorbalance	V->V	Adjust the color balance.
TSC colorchannelmixer	V->V	Adjust colors by mixing color channels.
TSC colorcontrast	V->V	Adjust color contrast between RGB components.
TSC colorcorrect	V->V	Adjust color white balance selectively for blacks and whites.
TSC colorize	V->V	Overlay a solid color on the video stream.
TSC colorkey	V->V	Turns a certain color into transparency. Operates on RGB colors.
TSC colorhold	V->V	Turns a certain color range into gray. Operates on RGB colors.
TSC colorlevels	V->V	Adjust the color levels.
TS. colormatrix	V->V	Convert color matrix.
TS. colorspace	V->V	Convert between colorspaces.
TSC colortemperature	V->V	Adjust color temperature of video.
TSC convolution	V->V	Apply convolution filter.
TS. convolve	VV->V	Convolve first video stream with second video stream.
... copy	V->V	Copy the input video unchanged to the output.
... cover_rect	V->V	Find and cover a user specified object.
..C crop	V->V	Crop the input video.
T.. cropdetect	V->V	Auto-detect crop size.
... cue	V->V	Delay filtering to match a cue.
TSC curves	V->V	Adjust components curves.
.SC datascope	V->V	Video data analysis.
T.C dblur	V->V	Apply Directional Blur filter.
TS. dctdnoiz	V->V	Denoise frames using 2D DCT.
TSC deband	V->V	Debands video.
T.C deblock	V->V	Deblock video.
... decimate	N->V	Decimate frames (post field matching filter).
TS. deconvolve	VV->V	Deconvolve first video stream with second video stream.
TS. dedot	V->V	Reduce cross-luminance and cross-color.
TSC deflate	V->V	Apply deflate effect.
... deflicker	V->V	Remove temporal frame luminance variations.
... deinterlace_qsv	V->V	QuickSync video deinterlacing
... dejudder	V->V	Remove judder produced by pullup.
T.. delogo	V->V	Remove logo from input video.
T.. derain	V->V	Apply derain filter to the input.
... deshake	V->V	Stabilize shaky video.
TSC despill	V->V	Despill video.
... detelecine	V->V	Apply an inverse telecine pattern.
TSC dilation	V->V	Apply dilation effect.
T.. displace	VVV->V	Displace pixels.
... dnn_processing	V->V	Apply DNN processing filter to the input.
.S. doubleweave	V->V	Weave input video fields into double number of frames.
T.C drawbox	V->V	Draw a colored box on the input video.
... drawgraph	V->V	Draw a graph using input video metadata.
T.C drawgrid	V->V	Draw a colored grid on the input video.
T.C drawtext	V->V	Draw text on top of video frames using libfreetype library.

T.. edgedetect	V->V	Detect and draw edge.
... elbg	V->V	Apply posterize effect, using the ELBG algorithm.
T.. entropy	V->V	Measure video frames entropy.
.S. epx	V->V	Scale the input using EPX algorithm.
T.C eq	V->V	Adjust brightness, contrast, gamma, and saturation.
TSC erosion	V->V	Apply erosion effect.
TSC estdif	V->V	Apply Edge Slope Tracing deinterlace.
TSC exposure	V->V	Adjust exposure of the video stream.
... extractplanes	V->N	Extract planes as grayscale frames.
TS. fade	V->V	Fade in/out input video.
T.. fftdnoiz	V->V	Denoise frames using 3D FFT.
T.. fftfilt	V->V	Apply arbitrary expressions to pixels in frequency domain.
... field	V->V	Extract a field from the input video.
... fieldhint	V->V	Field matching using hints.
... fieldmatch	N->V	Field matching for inverse telecine.
T.. fieldorder	V->V	Set the field order.
T.C fillborders	V->V	Fill borders of the input video.
... find_rect	V->V	Find a user specified object.
T.. floodfill	V->V	Fill area with same color with another color.
... format	V->V	Convert the input video to one of the specified pixel formats.
... fps	V->V	Force constant framerate.
... framepack	VV->V	Generate a frame packed stereoscopic video.
.S. framerate	V->V	Upsamples or downsamples progressive source between specified frame rates.
T.. framestep	V->V	Select one frame every N frames.
... freezedetect	V->V	Detects frozen video input.
... freezeframes	VV->V	Freeze video frames.
T.. fspp	V->V	Apply Fast Simple Post-processing filter.
TSC gblur	V->V	Apply Gaussian Blur filter.
TS. geq	V->V	Apply generic equation to each pixel.
T.. gradfun	V->V	Debands video quickly using gradients.
... graphmonitor	V->V	Show various filtergraph stats.
TS. greyedge	V->V	Estimates scene illumination by grey edge assumption.
TSC haldclut	VV->V	Adjust colors using a Hald CLUT.
TS. hflip	V->V	Horizontally flip the input video.
T.. histeq	V->V	Apply global color histogram equalization.
... histogram	V->V	Compute and draw a histogram.
TSC hqdn3d	V->V	Apply a High Quality 3D Denoiser.
.S. hqx	V->V	Scale the input by 2, 3 or 4 using the hq*x magnification algorithm.
.S. hstack	N->V	Stack video inputs horizontally.
T.C hue	V->V	Adjust the hue and saturation of the input video.
... hwdownload	V->V	Download a hardware frame to a normal frame
... hwmap	V->V	Map hardware frames
... hwupload	V->V	Upload a normal frame to a hardware frame
... hwupload_cuda	V->V	Upload a system memory frame to a CUDA device.

T.. hysteresis	VV->V	Grow first stream into second stream by connecting components.
TS. identity	VV->V	Calculate the Identity between two video streams.
... idet	V->V	Interlace detect Filter.
T.C il	V->V	Deinterleave or interleave fields.
TSC inflate	V->V	Apply inflate effect.
... interlace	V->V	Convert progressive video into interlaced.
... interleave	N->V	Temporally interleave video inputs.
... kerndeint	V->V	Apply kernel deinterlacing to the input.
TSC kirsch	V->V	Apply kirsch operator.
TSC lagfun	V->V	Slowly update darker pixels.
TSC lenscorrection	V->V	Rectify the image by correcting for lens distortion.
... libvmaf	VV->V	Calculate the VMAF between two video streams.
TSC limiter	V->V	Limit pixels components to the specified range.
... loop	V->V	Loop video frames.
TSC lumakey	V->V	Turns a certain luma into transparency.
TSC lut	V->V	Compute and apply a lookup table to the RGB/YUV input video.
TSC lut1d	V->V	Adjust colors using a 1D LUT.
TSC lut2	VV->V	Compute and apply a lookup table from two video inputs.
TSC lut3d	V->V	Adjust colors using a 3D LUT.
TSC lutrgb	V->V	Compute and apply a lookup table to the RGB input video.
TSC lutyuv	V->V	Compute and apply a lookup table to the YUV input video.
TSC maskedclamp	VVV->V	Clamp first stream with second stream and third stream.
TSC maskedmax	VVV->V	Apply filtering with maximum difference of two streams.
TSC maskedmerge	VVV->V	Merge first stream with second stream using third stream as mask.
TSC maskedmin	VVV->V	Apply filtering with minimum difference of two streams.
TSC maskedthreshold	VV->V	Pick pixels comparing absolute difference of two streams with threshold.
TSC maskfun	V->V	Create Mask.
... mcdeint	V->V	Apply motion compensating deinterlacing.
TSC median	V->V	Apply Median filter.
... mergeplanes	N->V	Merge planes.
... mestimate	V->V	Generate motion vectors.
T.. metadata	V->V	Manipulate video frame metadata.
T.. midequalizer	VV->V	Apply Midway Equalization.
... minterpolate	V->V	Frame rate conversion using Motion Interpolation.
TSC mix	N->V	Mix video inputs.
TSC monochrome	V->V	Convert video to gray using custom color filter.
... mpdecimate	V->V	Remove near-duplicate frames.
TS. msad	VV->V	Calculate the MSAD between two video streams.
TSC negate	V->V	Negate input video.
TS. nlmeans	V->V	Non-local means denoiser.
TSC nnedi	V->V	Apply neural network edge directed interpolation intra-only deinterlacer.
... noformat	V->V	Force libavfilter not to use any of the specified pixel formats for the input to the next filter.
TS. noise	V->V	Add noise.
T.C normalize	V->V	Normalize RGB video.

... null	V->V	Pass the source unchanged to the output.
T.C oscilloscope	V->V	2D Video Oscilloscope.
TSC overlay	VV->V	Overlay a video source on top of the input.
... overlay_qsv	VV->V	Quick Sync Video overlay.
... overlay_cuda	VV->V	Overlay one video on top of another using CUDA
T.. owdenoise	V->V	Denoise using wavelets.
... pad	V->V	Pad the input video.
... palettegen	V->V	Find the optimal palette for a given stream.
... paletteuse	VV->V	Use a palette to downsample an input video stream.
T.. perms	V->V	Set permissions for the output video frame.
TS. perspective	V->V	Correct the perspective of video.
T.C phase	V->V	Phase shift fields.
... photosensitivity	V->V	Filter out photosensitive epilepsy seizure-inducing flashes.
... pixdescstest	V->V	Test pixel format definitions.
T.C pixscope	V->V	Pixel data analysis.
T.C pp	V->V	Filter video using libpostproc.
T.. pp7	V->V	Apply Postprocessing 7 filter.
TS. premultiply	N->V	PreMultiply first stream with first plane of second stream.
TSC prewitt	V->V	Apply prewitt operator.
TSC pseudocolor	V->V	Make pseudocolored video frames.
TS. psnr	VV->V	Calculate the PSNR between two video streams.
... pullup	V->V	Pullup from field sequence to frames.
T.. qp	V->V	Change video quantization parameters.
... random	V->V	Return random frames.
TSC readeia608	V->V	Read EIA-608 Closed Caption codes from input video and write them to frame metadata.
... readvitic	V->V	Read vertical interval timecode and write it to frame metadata.
... realtime	V->V	Slow down filtering to match realtime.
TS. remap	VVV->V	Remap pixels.
TS. removegrain	V->V	Remove grain.
T.. removelogo	V->V	Remove a TV logo based on a mask image.
... repeatfields	V->V	Hard repeat fields based on MPEG repeat field flag.
... reverse	V->V	Reverse a clip.
TSC rgbshift	V->V	Shift RGBA.
TSC roberts	V->V	Apply roberts cross operator.
TSC rotate	V->V	Rotate the input image.
T.. sab	V->V	Apply shape adaptive blur.
..C scale	V->V	Scale the input video size and/or convert the image format.
... scale_cuda	V->V	GPU accelerated video resizer
... scale_qsv	V->V	QuickSync video scaling and format conversion
..C scale2ref	VV->VV	Scale the input video size and/or convert the image format to the given reference.
... scdet	V->V	Detect video scene change
TSC scroll	V->V	Scroll input video.
... select	V->N	Select video frames to pass in output.
TS. selectivecolor	V->V	Apply CMYK adjustments to specific color ranges.

... sendcmd	V->V	Send commands to filters.
... separatefields	V->V	Split input video frames into fields.
... setdar	V->V	Set the frame display aspect ratio.
... setfield	V->V	Force field for the output video frame.
... setparams	V->V	Force field, or color property for the output video frame.
... setpts	V->V	Set PTS for the output video frame.
... setrange	V->V	Force color range for the output video frame.
... setsar	V->V	Set the pixel sample aspect ratio.
... settb	V->V	Set timebase for the video output link.
TSC shear	V->V	Shear transform the input image.
... showinfo	V->V	Show textual information for each video frame.
... showpalette	V->V	Display frame palette.
T.. shuffleframes	V->V	Shuffle video frames.
TS. shufflepixels	V->V	Shuffle video pixels.
T.. shuffleplanes	V->V	Shuffle video planes.
T.. sidedata	V->V	Manipulate video frame side data.
.S. signalstats	V->V	Generate statistics from video analysis.
... signature	N->V	Calculate the MPEG-7 video signature
T.. smartblur	V->V	Blur the input video without impacting the outlines.
TSC sobel	V->V	Apply sobel operator.
... split	V->N	Pass on the input to N video outputs.
T.C spp	V->V	Apply a simple post processing filter.
... sr	V->V	Apply DNN-based image super resolution to the input.
TS. ssim	VV->V	Calculate the SSIM between two video streams.
.S. stereo3d	V->V	Convert video stereoscopic 3D view.
..C streamselect	N->N	Select video streams
... subtitles	V->V	Render text subtitles onto input video using the libass library.
.S. super2xsai	V->V	Scale the input by 2x using the Super2xSaI pixel art algorithm.
T.C swaprect	V->V	Swap 2 rectangular objects in video.
T.. swapuv	V->V	Swap U and V components.
TSC tblend	V->V	Blend successive frames.
... telecine	V->V	Apply a telecine pattern.
... thistogram	V->V	Compute and draw a temporal histogram.
TS. threshold	VVVV->V	Threshold first video stream using other video streams.
T.. thumbnail	V->V	Select the most representative frame in a given sequence of consecutive frames.
... thumbnail_cuda	V->V	Select the most representative frame in a given sequence of consecutive frames.
... tile	V->V	Tile several successive frames together.
... tinterlace	V->V	Perform temporal field interlacing.
TSC tlut2	V->V	Compute and apply a lookup table from two successive frames.
TSC tmedian	V->V	Pick median pixels from successive frames.
T.. tmidequalizer	V->V	Apply Temporal Midway Equalization.
TSC tmix	V->V	Mix successive video frames.
.S. tonemap	V->V	Conversion to/from different dynamic ranges.
... tpad	V->V	Temporarily pad video frames.

.S. transpose	V->V	Transpose input video.
... trim	V->V	Pick one continuous section from the input, drop the rest.
TS. unpremultiply	N->V	UnPreMultiply first stream with first plane of second stream.
TS. unsharp	V->V	Sharpen or blur the input video.
... until	V->V	Until a frame into a sequence of frames.
T.. uspp	V->V	Apply Ultra Simple / Slow Post-processing filter.
.SC v360	V->V	Convert 360 projection of video.
T.. vaguedenoiser	V->V	Apply a Wavelet based Denoiser.
... vectorscope	V->V	Video vectorscope.
T.. vflip	V->V	Flip the input video vertically.
... vfrdet	V->V	Variable frame rate detect filter.
TSC vibrance	V->V	Boost or alter saturation.
... vidstabdetect	V->V	Extract relative transformations, pass 1 of 2 for stabilization (see vidstabtransform for pass 2).
... vidstabtransform	V->V	Transform the frames, pass 2 of 2 for stabilization (see vidstabdetect for pass 1).
TS. vif	VV->V	Calculate the VIF between two video streams.
T.. vignette	V->V	Make or reverse a vignette effect.
... vmafmotion	V->V	Calculate the VMAF Motion score.
... vpp_qsv	V->V	Quick Sync Video VPP.
.S. vstack	N->V	Stack video inputs vertically.
TSC w3fdif	V->V	Apply Martin Weston three field deinterlace.
.S. waveform	V->V	Video waveform monitor.
.S. weave	V->V	Weave input video fields into frames.
.S. xbr	V->V	Scale the input using xBR algorithm.
.S. xfade	VV->V	Cross fade one video with another video.
TSC xmedian	N->V	Pick median pixels from several video inputs.
.S. xstack	N->V	Stack video inputs into custom layout.
TS. yadif	V->V	Deinterlace the input image.
T.. yadif_cuda	V->V	Deinterlace CUDA frames
TSC yaepblur	V->V	Yet another edge preserving blur filter.
... zmq	V->V	Receive commands through ZMQ and broker them to filters.
... zoompan	V->V	Apply Zoom & Pan effect.
..C zscale	V->V	Apply resizing, colorspace and bit depth conversion.
... allrgb	I->V	Generate all RGB colors.
... allyuv	I->V	Generate all yuv colors.
... cellauto	I->V	Create pattern generated by an elementary cellular automaton.
..C color	I->V	Provide an uniformly colored input.
.S. gradients	I->V	Draw a gradients.
... haldclutsrc	I->V	Provide an identity Hald CLUT.
... life	I->V	Create life.
... mandelbrot	I->V	Render a Mandelbrot fractal.
... mptestsrc	I->V	Generate various test pattern.
... nullsrc	I->V	Null video source, return unprocessed video frames.
... pal75bars	I->V	Generate PAL 75% color bars.
... pal100bars	I->V	Generate PAL 100% color bars.

... rgbsrc	->V	Generate RGB test pattern.
.S. sierpinskifractal	->V	Render a Sierpinski fractal.
... smptebars	->V	Generate SMPTE color bars.
... smptehdbars	->V	Generate SMPTE HD color bars.
... testsrc	->V	Generate test pattern.
... testsrc2	->V	Generate another test pattern.
... yuvtestsink	->V	Generate YUV test pattern.
... nullsink	V->	Do absolutely nothing with the input video.
... abitscope	A->V	Convert input audio to audio bit scope video output.
... adrawgraph	A->V	Draw a graph using input audio metadata.
... agraphmonitor	A->V	Show various filtergraph stats.
... ahistogram	A->V	Convert input audio to histogram video output.
... aphasemeter	A->N	Convert input audio to phase meter video output.
... avectorscope	A->V	Convert input audio to vectorscope video output.
.C concat	N->N	Concatenate audio and video streams.
... showcqt	A->V	Convert input audio to a CQT (Constant/Clamped Q Transform) spectrum video output.
... showfreqs	A->V	Convert input audio to a frequencies video output.
.S. showspatial	A->V	Convert input audio to a spatial video output.
.S. showspectrum	A->V	Convert input audio to a spectrum video output.
.S. showspectrumpic	A->V	Convert input audio to a spectrum video output single picture.
... showvolume	A->V	Convert input audio volume to video output.
... showwaves	A->V	Convert input audio to a video output.
... showwavespic	A->V	Convert input audio to a video output single picture.
... spectrumsynth	VV->A	Convert input spectrum videos to audio output.
.C amovie	->N	Read audio from a movie source.
.C movie	->N	Read from a movie source.
... afifo	A->A	Buffer input frames and send them when they are requested.
... fifo	V->V	Buffer input images and send them when they are requested.
... abuffer	->A	Buffer audio frames, and make them accessible to the filterchain.
... buffer	->V	Buffer video frames, and make them accessible to the filterchain.
... abuffersink	A->	Buffer audio frames, and make them available to the end of the filter graph.
... buffersink	V->	Buffer video frames, and make them available to the end of the filter graph.

2.211 Expression evaluation

This is a subset of the available expressions:

<code>abs(x)</code>	Return absolute value of <i>x</i> .
<code>acos(x)</code>	Return arccosine of <i>x</i> .
<code>asin(x)</code>	Return arcsine of <i>x</i> .
<code>atan(x)</code>	Return arctangent of <i>x</i> .
<code>atan2(x,y)</code>	Return the arc tangent of <i>y/x</i> in the range -Pi to Pi.
<code>between(x,min,max)</code>	Return 1 if <i>x</i> is greater than or equal to <i>min</i> and less than or equal to <i>max</i> , 0 otherwise. Use this workaround if the <i>max</i> value shall be excluded from the intervall: <code>bitand(gte(t,min),lt(t,max))</code>
<code>bitand(x,y)</code>	Compute bitwise and operation on <i>x</i> and <i>y</i> .
<code>bitor(x,y)</code>	Compute bitwise or operation on <i>x</i> and <i>y</i> .
<code>ceil(expr)</code>	Round the value of expression <i>expr</i> upwards to the nearest integer. For example, "ceil(1.5)" is "2.0".
<code>clip(x,min,max)</code>	Return the value of <i>x</i> clipped between <i>min</i> and <i>max</i> .
<code>cos(x)</code>	Compute cosine of <i>x</i> .
<code>eq(x,y)</code>	Return 1 if <i>x</i> and <i>y</i> are equivalent, 0 otherwise.
<code>exp(x)</code>	Compute exponential of <i>x</i> .
<code>floor(expr)</code>	Round the value of expression <i>expr</i> downwards to the nearest integer. For example, "floor(-1.5)" is "-2.0".
<code>gt(x,y)</code>	Return 1 if <i>x</i> is greater than <i>y</i> , 0 otherwise.
<code>gte(x,y)</code>	Return 1 if <i>x</i> is greater than or equal to <i>y</i> , 0 otherwise.
<code>hypot(x,y)</code>	Return $\sqrt{x^*x + y^*y}$
<code>if(x,y)</code>	Evaluate <i>x</i> , and if the result is non-zero return the result of the evaluation of <i>y</i> , return 0 otherwise.
<code>if(x,y,z)</code>	Evaluate <i>x</i> , and if the result is non-zero return the evaluation result of <i>y</i> , otherwise the evaluation result of <i>z</i> .
<code>ifnot(x,y)</code>	Evaluate <i>x</i> , and if the result is zero return the result of the evaluation of <i>y</i> , return 0 otherwise.
<code>ifnot(x,y,z)</code>	Evaluate <i>x</i> , and if the result is zero return the evaluation result of <i>y</i> , otherwise the evaluation result of <i>z</i> .

<code>ld(var)</code>	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 initialized with 0 at the beginning. Warning: The <code>random(x)</code> function uses the same variables!
<code>lerp(x,y,z)</code>	Return <code>x</code> if <code>z = 0</code> , <code>y</code> if <code>z = 1</code> and a linear interpolation for any value of <code>z</code> . There is no clipping for <code>z < 0</code> or <code>z > 1</code> . The return value is: $x + z * (y - x)$
<code>log(x)</code>	Compute natural logarithm of <code>x</code> .
<code>lt(x,y)</code>	Return 1 if <code>x</code> is less than <code>y</code> , 0 otherwise.
<code>lte(x,y)</code>	Return 1 if <code>x</code> is less than or equal to <code>y</code> , 0 otherwise.
<code>max(x,y)</code>	Return the maximum between <code>x</code> and <code>y</code> .
<code>min(x,y)</code>	Return the minimum between <code>x</code> and <code>y</code> .
<code>mod(x,y)</code>	Return the remainder of division of <code>x</code> by <code>y</code> .
<code>pow(x,y)</code>	Return the power of <code>x</code> elevated <code>y</code> , it is equivalent to " <code>(x)^y</code> ".
<code>print(t)</code>	Print the value of expression <code>t</code> and returns the value of the expression printed.
<code>random(x)</code>	Return a pseudo random value between 0.0 and 1.0. <code>x</code> is the index of the internal variable which will be used to save the seed/state. Warning: The <code>Id(var)</code> and <code>st(var, expr)</code> functions use the same variables! Note: <code>random(0)</code> uses the variable 0 as a seed value. If you want to set the seed value, you must use the <code>st(0, expr)</code> function.
<code>root(expr,max)</code>	Find an input value for which the function represented by <code>expr</code> with argument <code>Id(0)</code> is 0. The input value must be in the interval <code>[0..max]</code> . The expression in <code>expr</code> must denote a continuous function or the result is undefined. <code>Id(0)</code> 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 <code>Id(0)</code> . When the expression evaluates to 0 then the corresponding input value will be returned. Warning: If there is no input value in the <code>[0..max]</code> interval for which the result of the expression becomes 0, then the <code>root()</code> function returns a wrong result!
<code>round(expr)</code>	Round the value of expression <code>expr</code> to the nearest integer. For example, " <code>round(1.5)</code> " is "2.0".
<code>sgn(x)</code>	Return the sign of <code>x</code> (-1, 0 or +1)
<code>sin(x)</code>	Return sine of <code>x</code> .
<code>sqrt(x)</code>	Return the square root ix <code>x</code> .
<code>squish(x)</code>	Compute expression $1/(1+\exp(4*x))$ $\text{squish}(-1) = 0.982, \text{squish}(0) = 0.5, \text{squish}(1) = 0.0179$

<code>st(var,expr)</code>	Store the value of the expression <code>expr</code> in an internal variable. <code>var</code> 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 <code>random(x)</code> function uses the same variables!
<code>tan(x)</code>	Compute tangent of <code>x</code> .
<code>trunc(expr)</code>	Round the value of expression <code>expr</code> towards zero to the nearest integer. For example, " <code>trunc(-1.5)</code> " is "-1.0".
<code>while(cond,expr)</code>	Evaluate expression <code>expr</code> while the expression <code>cond</code> is non-zero, and returns the value of the last <code>expr</code> evaluation, or <code>NAN</code> if <code>cond</code> was always false.
<code>PI</code>	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:

```
between(t,0,1) * lerp(v0,v1,t) + between(t,1,2) * lerp(v1,v2,t-1) + between(t,2,3) * lerp(v2,v3,t-2) + ...
```

Workaround for segment-wise linear interpolation, two segments per second:

```
between(t,0,0.5) * lerp(v00,v05,2*t) + between(t,0.5,1.0) * lerp(v05,v10,2*(t-0.5)) + between(t,1.0,1.5) * lerp(v10,v15,2*(t-1.0)) + ...
```

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`.

Workaround for `between(t,min,max)` if the maximum shall be excluded from the intervall: `bitand(gte(t,min),lt(t,max))`

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.

This is a trick for the "geq" filter if you want to use the same expression for the R, G and B channels. Of course it's a bad idea to use the same expression three times, because that's slow. The trick is to set the format to `gray8`, then apply the `geq` filter with only one expression, and then set the format to `rgb24`, as in this example:

```
format=gray8,geq='128+64*sin(0.2*x)',format=rgb24
```

2.212 Evaluation of mathematical functions

```
set "START=-1"      :: Start value
set "STEP=0.1"       :: Step value
set "N=21"           :: Number of values

rem  In this example the input is in variable (0) and has the values -1, -0.9, -0.8, ... 0.9, 1.0
ffmpeg -loglevel repeat -f lavfi -i nullsrc=size=%N%x1 -vf format=gray,geq='st(0,%START%+X*%STEP%
');print(ld(0));print(asin(ld(0)))' -frames 1 -f null NUL

pause
```

Note: Use "-loglevel repeat" to suppress the "Last message repeated n times" messages in the console output.

Note: For "format=gray" the values are evaluated one time, however for "format=rgb24" they would be evaluated three times.

2.213 Error handling in expressions

It's a known problem that there is no error message thrown if an expression has an invalid result. Unfortunately the results "nan", "inf" and "-inf" are all silently converted to zero in the output file:

```
ffmpeg -f lavfi -i nullsrc=size=1x1 -vf format=gray16le,geq='print(sqrt(-1))' -frames 1 -y test.pgm  
pause
```

Expression	Result	Value in *.pgm output file	isnan()	isinf()
<code>sqrt(-1)</code>	<code>nan</code>	<code>0</code>	<code>1 = true</code>	<code>0 = false</code>
<code>1/0</code>	<code>inf</code>	<code>0</code>	<code>0 = false</code>	<code>1 = true</code>
<code>-1/0</code>	<code>-inf</code>	<code>0</code>	<code>0 = false</code>	<code>1 = true</code>
<code>asin(2)</code>	<code>nan</code>	<code>0</code>	<code>1 = true</code>	<code>0 = false</code>
<code>log(0)</code>	<code>-inf</code>	<code>0</code>	<code>0 = false</code>	<code>1 = true</code>
<code>log(-1)</code>	<code>nan</code>	<code>0</code>	<code>1 = true</code>	<code>0 = false</code>

2.214 Debugging of expressions

Infinite loop, can be used after printing a variable so that the result is visible at the end of the console output:

```
while(1,0);
```

Check if variable (0) is in the [0.99 ... 1.01] range, if yes then print "11111111", otherwise print nothing:

```
if(between(ld(0),0.99,1.01),print(11111111));
```

Check if variable (0) is in the [0.99 ... 1.01] range, if yes then print nothing, otherwise print "11111111":

```
ifnot(between(ld(0),0.99,1.01),print(11111111));
```

Same as before, but enter an infinite loop after the message was printed, so that you can't overlook it in the console output:

```
ifnot(between(ld(0),0.99,1.01),print(11111111);while(1,0));
```

Check if variable (0) is "nan" (not a number), if yes then print "22222222" and enter an infinite loop, otherwise print nothing:

```
if(isnan(ld(0)),print(22222222);while(1,0));
```

Check if variable (0) is "inf" (plus or minus infinite), if yes then print "33333333" and enter an infinite loop, otherwise print nothing:

```
if(isinf(ld(0)),print(33333333);while(1,0));
```

Check if variable (0) is "nan", if yes then print "44444444" and also print the geq filter's input variables X and Y, then enter the infinite loop, otherwise print nothing:

```
if(isnan(ld(0)),print(44444444);print(X);print(Y);while(1,0));
```

Check if variable (0) is "nan" or "inf", if yes then print "55555555" and also the geq filter's input variables X and Y, then enter the infinite loop, otherwise print nothing. Both command lines are equivalent because logical "or" can also be realized by the "+" operator:

```
if(bitor(isnan(ld(0)),isinf(ld(0))),print(55555555);print(X);print(Y);while(1,0));
if(isnan(ld(0))+isinf(ld(0)),print(55555555);print(X);print(Y);while(1,0));
```

Check if the 3-dimensional vector in variables (4), (5) and (6) is normalized. If yes, then do nothing, otherwise print "66666666" and print the geq filter's input variables X and Y and the content of the variables (4), (5) and (6), before entering an infinite loop.

```
if(gt(abs(1-sqrt(ld(4)*ld(4)+ld(5)*ld(5)+ld(6)*ld(6))),1e-6),^
print(66666666);print(X);print(Y);print(ld(4));print(ld(5));print(ld(6));while(1,0));
```

Note: Use "-loglevel repeat" to suppress the "Last message repeated n times" messages in the console output.

Note: FFmpeg uses multithreading by default. It may be useful to use the option "-threads 1" so that only one thread is used, because otherwise you get multiple print outputs, one from each thread, before each thread has run into its own endless loop. "-threads 1" must be written after the input file(s) and before the output file. In some cases it may also be useful to use "-slices 1".

2.215 List the R,G,B values as ASCII text file

List the R,G,B values of a (small) image as *.txt ASCII text file. Normally the output of the print commands would go to the console output. If you want to send it to a file instead, you can use this trick:

```
ffmpeg -f lavfi -i haldclutsrc=2 -frames 1 -y clut.png  
ffmpeg -loglevel repeat+error -i clut.png -vf geq=b='print(r(X,Y),16);print(g(X,Y),16);print(b(X,Y),16)' -threads 1 -f null NUL 2> out.txt  
  
pause
```

Note: "loglevel error" means that only error messages are logged, and nothing else. But the outputs of the print commands are also logged, because for them the loglevel is explicitly set to 16, which means "error".

Note: It's important to use "-threads 1", as otherwise multiple threads are used and the output values aren't printed in the correct order.

Note: 2> does redirect the StdErr output to a file. 1> would be used for redirecting StdOut to a file (but FFmpeg doesn't use StdOut).

Note: If "-report" is added, the console output is also written to a file named *ffmpeg-YYYYMMDD-HHMMSS.log*

Note: This technique for writing the pixel values fails if the height of the input image is 1. This seems to be a bug in the "geq" filter.

As a workaround for the height=1 problem you can do the same thing with the blend filter:

```
ffmpeg -f lavfi -i haldclutsrc=2 -vf crop=iw:1 -frames 1 -y test.png  
ffmpeg -loglevel repeat+error -i test.png -lavfi split,blend=c2_expr='print(A,16)' -threads 1 -f null NUL 2> out.txt  
  
pause
```

Note: "c2_expr" is for the red channel.

Example for printing the pixel values of a brightness profile:

```
rem Make a brightness profile:  
ffmpeg -f lavfi -i color=white -vf format=gray8,vignette=PI/4:x0=(w-1)/2:y0=(h-1)/2,crop=iw:1 -frames 1 -y test.png  
  
rem Print the pixel values to a *.txt file:  
ffmpeg -loglevel repeat+error -i test.png -lavfi split,blend=c0_expr='print(A,16)' -threads 1 -f null NUL 2> out.txt  
pause
```

Note: In this case "c0_expr" is used, because the monochrome image has only one channel.

Another workaround for the height=1 problem is to use the "geq" filter, but select one row in the expression, instead of cropping the frame to height=1.

```
ffmpeg -f lavfi -i color=white -vf format=gray8,vignette=PI/4:x0=(w-1)/2:y0=(h-1)/2 -frames 1 -y test.png  
ffmpeg -loglevel repeat+error -i test.png -lavfi geq=r='if(eq(Y,120),print(r(X,Y),16))' -threads 1 -f null NUL 2> out.txt  
pause
```

2.216 Downloading videos from Facebook

Downloading a video from Facebook is possible, but a little bit complicated.

In Facebook, click on the three dots and then select "copy link". Now go to <https://www.fbdown.net/>, paste the link and click on "Download". Now click on "Download Video in Normal Quality" or "Download Video in HD Quality". This will open the video in a browser, where you can make a right click and save the video.

See also (german) <https://www.heise.de/tipps-tricks/Facebook-Videos-herunterladen-so-geht-s-3904709.html>

2.217 Uploading spherical 360 images to Facebook

A spherical 360 image is basically a normal equirectangular *.jpg image, but additionally a few properties must be set:

The property "make" must be set to "Ricoh" and "model" must be set to "Ricoh THETA S". These two changes are sufficient so that Facebook recognizes the image as spherical 360 image. Please note that width:height must be 2:1 and the image size must not exceed 6000x3000 pixels.

You can edit these properties in the Windows Explorer, or you can make the changes with "Exiftool" as in the following example:

```
set "INPUT=in.jpg"          :: Input and output image
exiftool -overwrite_original -make="Ricoh" -model="Ricoh THETA S" %INPUT%
pause
```

See also (in german): <https://www.allerstorfer.at/photosphere-xmp-metadaten-zu-360-panorama-hinzufuegen/>

See also: https://evanwill.github.io/_drafts/notes/photosphere.html

2.218 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.

See also <https://trac.ffmpeg.org/wiki/Encode/YouTube>

2.219 Downloading videos from Youtube

Downloading a video from Youtube is possible with the help of external websites:

Go to Youtube and copy the URL of the video. Now go (for example) to <https://www.downvids.net/>, paste the URL, select the file type and video quality, then click on "DOWNLOAD".

See also (german) <https://www.heise.de/tipps-tricks/Youtube-Video-herunterladen-am-Computer-so-geht-s-3931676.html>

2K and 4K video downloads are easy with the free "4K Video Downloader". Use the mkv format for high resolution videos:

<https://www.4kdownload.com/de/>

Notes from Andrei B.:

You can also use some browser extensions:

Firefox - addon "YouTube Video and Audio Downloader (Dev Edt.)" (uses ffmpeg as companion, plus a native companion for assembling streams)

Chrome - addon "YouTube Video and MP3 Downloader" <https://addoncrop.com/youtube-video-downloader/>

2.220 Youtube recommended settings

```
ffmpeg \
-i input.mp4 \
-r 30000/1001 \
-vf scale="1920:1080" \
-codec:v libx264 \
-crf 21 \
-bf 2 \
-flags +cgop \
-pix_fmt yuv420p \
-c:a aac \
-b:a 128k \
-ac 2 \
-r:a 44100 \
-map 0:v:0 \
-map 0:a:0 \
-movflags faststart \
output.mp4

pause
```

Found on Rodrigo Polo's github site: <https://github.com/rodrigopolو/cheatsheets/blob/master/ffmpeg.md>

See also <https://trac.ffmpeg.org/wiki/Encode/YouTube>

See also (in german): <https://support.google.com/youtube/answer/1722171>

2.221 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>

See also: Encoding for streaming sites <https://trac.ffmpeg.org/wiki/EncodingForStreamingSites>

See also: Streaming guide <https://trac.ffmpeg.org/wiki/StreamingGuide>

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>

2.222 Limiting FFmpeg's encoding speed to realtime

There are two possible methods:

- Use the "-re" option (not possible with all input stream types)
- Use the "realtime" filter at the beginning of the filter chain

2.223 Point-to-point streaming

This is an example for streaming from one computer to another computer over LAN or WLAN:

	Sender side	Receiver side
Find out the IP addresses by typing ipconfig in the console window. The sender and receiver IPv4 addresses are different and the subnet masks must be equal. If sender and receiver are on the same computer (for testing), then IPv4 addresses are equal.	Windows-IP-Konfiguration Ethernet adapter Ethernet 2: Autoconfiguration IPv4 Address . . . : 169.254.255.43 Subnet Mask : 255.255.0.0 Wireless LAN adapter WLAN: Connection-specific DNS Suffix . : fritz.box IPv4 Address. : 192.168.178.45 Subnet Mask : 255.255.255.0	Windows-IP-Konfiguration Ethernet-Adapter LAN-Verbindung 3: IPv4-Adresse (Auto. Konfiguration): 169.254.23.82 Subnetzmase : 255.255.0.0 Drahtlos-LAN-Adapter Drahtlosnetzwerkverbindung: Verbindungsspezifisches DNS-Suffix: fritz.box IPv4-Adresse : 192.168.178.25 Subnetzmase : 255.255.255.0
Optional checking the LAN connection	ping -t 169.254.23.82	ping -t 169.254.255.43
Batch files for point-to-point streaming over LAN	ffmpeg -f lavfi -i sine -re -f lavfi -i testsrc2=s=vga -b:v 2M -f mpegts udp:// 169.254.23.82:1235	ffmpeg -i udp:// 169.254.255.43:1235 -f sdl2 - or ffplay udp:// 169.254.255.43:1235
Optional checking the WLAN connection	ping -t 192.168.178.25	ping -t 192.168.178.45
Batch files for point-to-point streaming over WLAN	ffmpeg -f lavfi -i sine -re -f lavfi -i testsrc2=s=vga -b:v 2M -f mpegts udp:// 192.168.178.25:1235	ffmpeg -i udp:// 192.168.178.45:1235 -f sdl2 - or ffplay 192.168.178.45:1235

Note: Windows firewall is a possible error source. At the receiver side, FFplay is better than FFmpeg (with sdl output device) because it also plays audio.

Note: Additional options for FFplay are -noborder -x 960 -y 960 -left 0 -top 0

There is a (possibly undocumented?) feature for adding packet size and input buffer size to the UDP output:

```
ffmpeg -f lavfi -i sine=frequency=1000:sample_rate=48000 -framerate 10 -loop 1 -i 4096.png -vf rotate='2*PI*t/240' -strict -1 -g 1  
-b:v 40M -f mpegs udp://169.254.23.82:1234?pkt_size=1316&buffer_size=65535
```

Note: `pkt_size` must be a multiple of 188.

Note: The option `-g 1` means that all frames are transferred as keyframes. The advantage is that transmission errors are only visible in one frame.

Note: If there are gaps in the audio transmission, try to reduce the framerate with the `-framerate` option.

Note: `-strict -1` is only required if the width or height of the frame is a multiple of 4096.

2.224 Streaming a video to a multicast address

See also <https://trac.ffmpeg.org/wiki/StreamingGuide> (But that guide isn't useful, as none of the examples did work when I tested them)

This is an example for sending the udp stream:

```
ffmpeg -re -f lavfi -i testsrc2=s=vga -b:v 2M -f mpegts udp://239.0.0.1:1234
```

Note: the -re flag is important and means that the stream is slowed down to realtime.

Alternatively you can use this test source which contains a horizontal bar that's scrolling vertically. This is good for checking if two receivers are running exactly synchronously:

```
ffmpeg -re -f lavfi -i color=white:s=vga -lavfi drawbox=h=20:c=red:t=fill,scroll=v=1/25 -b:v 2M -f mpegts udp://239.0.0.1:1234
```

This is an example for receiving the stream with FFplay:

```
ffplay udp://239.0.0.1:1234
```

Alternatively you can also receive the stream with FFmpeg:

```
ffmpeg -i udp://239.0.0.1:1234 -f sdl2 -
```

Note: In both cases it's possible to run multiple instances of the receiver.

Note: The full range of multicast addresses is from 224.0.0.0 to 239.255.255.255, but a lot of that range is restricted. If you want to use multicast for something private, better use 239.0.0.0 to 239.255.255.255.

See also: <https://superuser.com/questions/307130/ffmpeg-command-to-stream-video-to-a-multicast-address>

This is an example for starting two instances of FFplay:

```
start ffplay -noborder -x 640 -y 480 -left 0 -top 200 udp://239.0.0.1:1234
start ffplay -noborder -x 640 -y 480 -left 640 -top 200 udp://239.0.0.1:1234

pause
```

This is an example for starting two instances of FFmpeg:

```
start ffmpeg -i udp://239.0.0.1:1234 -window_borderless 1 -window_x 0 -window_y 200 -f sdl2 -
start ffmpeg -i udp://239.0.0.1:1234 -window_borderless 1 -window_x 640 -window_y 200 -f sdl2 -

pause
```

Note: If two or more monitors are connected to the computer, it's possible to show the videos on different monitors. For example if both monitors are 1920x1080, set -window_x to 1920 for showing the video at the left edge of the second monitor.

Example for multicast streaming with rtp protocol:

```
ffmpeg -re -f lavfi -i testsrc2=s=vga -b:v 2M -c:v h264 -f rtp -sdp_file video.sdp "rtp://239.0.0.1:1234"
```

Note: The *.sdp file contains important informations about the stream, which the receiver wants to know.

Note: The rtp protocol can only be used for one stream. Not for audio and video simultaneously.

Example for receiving the rtp stream with two instances of FFplay:

```
start ffplay -protocol_whitelist file,rtp,udp -noborder -x 640 -y 480 -left 0 -top 200 video.sdp
start ffplay -protocol_whitelist file,rtp,udp -noborder -x 640 -y 480 -left 640 -top 200 video.sdp
```

Example for receiving the rtp stream with two instances of FFmpeg:

```
start ffmpeg -protocol_whitelist file,rtp,udp -i video.sdp -window_borderless 1 -window_x 0 -window_y 600 -f sdl2 -
start ffmpeg -protocol_whitelist file,rtp,udp -i video.sdp -window_borderless 1 -window_x 640 -window_y 600 -f sdl2 -
```

Example for receiving the stream with VLC Player:

```
"C:\Program Files\VideoLAN\VLC\vlc.exe" video.sdp
```

Note: The VLC path must be written in "" because it contains a space character.

Note: It takes about 10 seconds until the video appears.

Note: It seems to be impossible to specify the position and size of the VLC window. If you have a working example, please let me know.

See also this website where I found a lot of useful infos about rtp: <https://www.kurento.org/blog/rtp-ii-streaming-ffmpeg>

2.225 RTMP Streaming

See also: <https://sonnati.wordpress.com/2011/08/30/ffmpeg-%E2%80%93-the-swiss-army-knife-of-internet-streaming-%E2%80%93-part-iv/>

2.226 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>

Hardware accelerated h.264 encoding on machines with Nvidia GPUs:

```
ffmpeg -i in.mp4 -c:v h264_nvenc out.mp4  
pause
```

Note: If you get the error message "Driver does not support the required nvenc API version. Required: 11.1 Found: 11.0" then you should update your Nvidia driver to the latest version.

Note: Hardware acceleration with -c:v h264_nvenc doesn't work with my laptop, if the width exceeds 4096.

Hardware accelerated h.265/HEVC encoding on machines with Nvidia GPUs:

```
ffmpeg -i in.mp4 -c:v hevc_nvenc out.mp4  
pause
```

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.

2.227 OpenCL Hardware Acceleration

It might be possible to accelerate the "remap" filter by using the "remap_opencl" filter. But that's complicated stuff and the documentation is incomplete. Google finds absolutely no examples for this filter. I found the following working example by a lot of try and error:

```
ffmpeg -init_hw_device opencl.gpu:1.0 -filter_hw_device gpu -i input.png -i xmap.pgm -i ymap.pgm -lavfi [0]hwupload[a];[1]hwupload[b];[2]hwupload[c];[a][b][c]remap_opencl,hwdownload,format=rgba,format=yuv420p" -y out.png
```

Note: Don't ask me why after hwdownload there are two format conversions required. I don't understand it. If the first conversion is removed, there comes the error message "invalid output format". If the second conversion is removed, FFmpeg does run forever.

Note: For the second format conversion, rgb24 does also work.

Note: In my test remap_opencl was significantly slower than the remap filter running on CPU. Even when the time of hwupload, hwdownload and the format conversions is excluded from the benchmark, remap_opencl is still significantly slower than remap. Conclusion: remap_opencl is useless.

See also:

<file:///C:/ffmpeg/ffmpeg-all.html#OpenCL-Video-Filters>

<https://trac.ffmpeg.org/wiki/HWAccelIntro>

<https://docs.nvidia.com/video-technologies/video-codec-sdk/ffmpeg-with-nvidia-gpu/>

2.228 Benchmarks

Same example as before, but with benchmarks for measuring the execution time of some filters:

```
ffmpeg -i poles2.png -i xmap.pgm -i ymap.pgm -lavfi "[0]bench=start[a];[a][1][2]remap,bench=stop" -frames 1 -y out.png  
ffmpeg -init_hw_device opencl.gpu:1.0 -filter_hw_device gpu -i poles2.png -i xmap.pgm -i ymap.pgm -lavfi  
"[0]bench=start,hwupload[a];[1]hwupload[b];[2]hwupload[c];[a][b]  
[c]remap_opencl,hwdownload,format=rgba,format=rgb24,bench=stop" -frames 1 -y out.png  
  
pause
```

2.229 FFmpeg console output

	Explanation:
SAR	Sample Aspect Ratio
DAR	Display Aspect Ratio
fps	frames per second
tbr	This is guessed from the video stream and is the value users want to see when they look for the video frame rate, except sometimes it is twice what one would expect because of field rate versus frame rate.
tbn	The time base in AVStream that has come from the container. It is used for all AVStream time stamps. (time base number?)
tbc	The time base in AVCodecContext for the codec used for a particular stream. It is used for all AVCodecContext and related time stamps.

2.230 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/>
- Click on "Source Code" and download the file "ffmpeg-snapshot.tar.bz2"
- 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_"

2.231 FFmpeg: Suggestions for improvement

Listed in alphabetic order:

"adecorrelate" filter:

-- Documentation: More info required about what's the purpose of this filter, with examples. I thought it could be used for echo cancelling, but this doesn't seem to be the case. I did try this:

```
ffmpeg -f lavfi -i "sine=500:b=2,channelmap=0|0" -lavfi asplit[a][b];[a]adelay=500[c];[b][c]amix=weights="1 0.2" -t 10 -y echo.wav  
ffmpeg -i echo.wav -lavfi adecorrelate=seed=22050 -y out.wav
```

"afffilt" filter:

-- Documentation: For the overlap parameter, what are the default values if set to 1?

"afreqshift" filter:

-- Documentation: The unit of the frequency shift is unclear.

All filters:

-- Documentation: Add which filters support "enable=" timeline support. Currently it's only available via "ffmpeg -filters".

-- 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 "testsrt2" source in the same command line, you have to specify the size for at least one of them, because the default sizes are different.

"amix" filter:

-- Documentation: What are the default weights? Are they 1 or 1/n or 1/sqrt(n)?

"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.

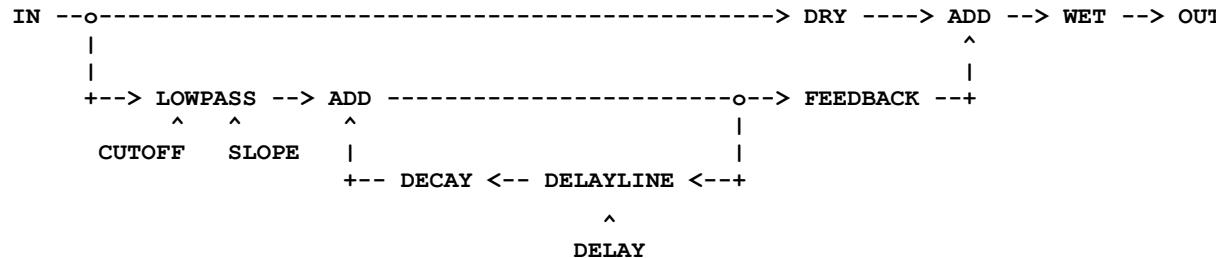
-- Documentation: Does this filter have a built-in limiter?

"aphaseshift" filter:

-- Documentation: The units of the phase shift and level are unclear.

"asubboost" filter:

-- The documentation of this filter is a total mess. After reverse engineering from the filter's output signal, I found that the block diagram looks like this:



The "feedback" option has a very misleading name. This isn't a feedback, however it's going straight forward to the output. It should be renamed "wet". What's now "wet" could be renamed "gain", or it could be removed because it's unnecessary. The explanations of the "wet" and "feedback" options are clearly wrong. The explanation of the "slope" option is unclear (I think the attenuation per octave is meant). The "delay" option is missing the information

that the unit is milliseconds.

"atilt" audio filter:

Documentation: I didn't understand what this filter is supposed to do.

"avsyncntest" multimedia source:

I don't understand how this source is expected to work.

ffmpeg -f lavfi -i avsyncntest=d=5 -y out.mp4

A working example is required.

"blend" filter:

-- Documentation: Please add documentation what all the modes do. A few modes are self-explaining, but most are not. A link to the wiki page would be helpful: <https://trac.ffmpeg.org/wiki/Blend>

-- Documentation: It's written that the "opacity" options are "Only used in combination with pixel component blend modes." But it's unclear what a "pixel component blend mode" is. It seems the "opacity" options aren't used if a user-defined expression is used. They are only used if one of the "mode" options is used. The default is "all_opacity=1", which means the full blend effect is applied. A smaller opacity value means the output is mixed with the first input stream. "all_opacity=0" means the effect is disabled and the first input stream is returned. But there is one exception from this rule: If the mode is "normal", then "all_opacity=1" returns the first stream and "all_opacity=0" returns the second stream.

"chromanr" filter:

-- Documentation of the threshold option is unclear.

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.

"colorchannelmixer" filter:

-- Documentation: This filter isn't listed in the correct alphabetic order (should be before colorcontrast).

"colorcorrect" filter:

-- Documentation: I didn't understand the "analyze" option. Better description and examples required.

"colorhold" filter:

-- Documentation: This filter isn't listed in the correct alphabetic order (should be before colorkey).

"colorlevels" filter:

-- Feature request: Add two new options "imin" and "imax" which set all three color channels simultaneously. Example:

```
colorlevels=rimin=0.1:gimin=0.1:bimin=0.1:rimax=0.3:gimax=0.3:bimax=0.3
```

could be written as:

```
colorlevels=imin=0.1:imax=0.3
```

"colormap" filter:

-- Documentation: Only the center pixels of the "source" and "target" fields are used. There is no averaging done in this filter. Examples are required. Default patch size is 64x64.

"colorspectrum" video source:

- Documentation: type=black is the default
- Feature request: type=none for disabling both black and white

"convolve" filter:

- Documentation: I know what a convolution is, but I don't understand what this filter is doing. Example required.

Cross correlation

- 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, and delay one of them by this amount. Can be used for synchronizing in-camera audio with sound from external recorder.

"cue" filter:

- please add a few 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: Which is the default mode?

-- 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? I think it's contrast (multiplicative), but I'm not 100% sure.

-- Feature request: Please allow to choose between additive and multiplicative deflicker. An example for additive deflicker is when you make a timelapse with a flash, and you know that the intensity of the flash is always the same. But you may have additive flicker if the setup isn't shielded from ambient light.

-- Feature request: Allow to deflicker the RGB channels separately, because fluorescent light does also have color flicker.

-- Feature request: Allow to split the video into many horizontal stripes and deflicker all stripes separately. Useful for fluorescent light and rolling shutter flicker. See my workaround in this book.

"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 (see the download links in chapter "CR2 and DNG images"), 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.

-- Feature request: Use LibRaw for importing RAW or DNG images, and throw away the DNG import stuff that doesn't work.

"drawbox" filter:

-- Documentation: Using a thickness greater than 1 doesn't increase the outer dimensions of the box. The number of pixels inside the box is (width-2*thickness) or (height-2*thickness).

"drawellipse" filter:

-- Feature request: A new filter for drawing circles or ellipses with or without filling.

"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.

Encoder and decoder for *.ser files

It's an uncompressed video format for astronomical videos. For infos about the SER video file format, see <https://github.com/cgarry/ser-player> and <https://sites.google.com/site/astropipp/ser-player>

"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.

-- Feature request: Add an option for specifying in which order the corrections are made.

-- 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.

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.

"exposure" filter:

-- Documentation: The order is first "black level" and then "exposure", which means first add then multiply. It works internally with floats and has no built-in limiter.

Expression evaluation:

-- 'between(x, min, max)' Feature request: Create an additional non-overlapping 'between2' function, with a slightly different definition: Return 1 if x is greater than or equal to *min* and less than *max*, 0 otherwise. This is useful for segment-wise linear interpolation, as in this example: `between(t, 0, 1) * lerp(2, 5, t) + between(t, 1, 2) * lerp(5, 6, t-1) + between(t, 2, 3) * lerp(6, 9, t-2) + ...`.

The problem in this example is that at the ends of the intervals both 'between' functions become true simultaneously.

-- '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)' Feature request: 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: $\text{between}(T+0.001,0,1)*\text{lerp}(200,400,T)+\text{between}(T+0.001,1,2)*\text{lerp}(400,500,T-1)+\text{between}(T+0.001,2,3)*\text{lerp}(500,550,T-2)$

-- Documentation: At the end of chapter 8 in ffmpeg-all.html is the list of prefixes. The last four entries (P, E, Z and Y) have a wrong power of two (10 must be added). See ticket 9707.

"-f image2pipe":

-- Documentation is missing in ffmpeg-all.html

"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.

"ffplay-all.htm"

-- Documentation: Why are the values of the options "-left" and "-top" called "title"? That should be integers.

-- Documentation: Please make clear that `-video_size` is used to manually tell ffplay the size for videos that do not contain a header (such as raw video), however it's not used to resize videos.

"ffmpeg-all.html"

-- Please add more details to chapter 34 "*Changing options at runtime with a command*", as was already described here:

<https://stackoverflow.com/questions/56058909/ffmpeg-drawtext-and-live-coordinates-with-sendcmd-zmq>

It should also be mentioned that two other methods exist: "sendcmd" and "zmq".

-- Add a chapter for accepted keys while FFmpeg is running, see function `check_keyboard_interaction()` in ffmpeg.c

-- Chapter 5.2 generic options: `-h muxer=muxer_name` Please add that this isn't only for muxers, but also for output devices. For example `ffmpeg -h muxer=sdl`

"fftfilt" filter:

-- Documentation: If the weight_U and weight_V expressions aren't set, they are by default copied from weight_Y. In most cases this doesn't make sense and produces a greenish tint in the output image. To solve this problem, you should always set weight_U and weight_V to 1.

-- All four examples: Add :weight_U=1:weight_V=1

-- Documentation: "Adjust the dc value (gain) of the ... plane of the image." This is a very misleading formulation, because dc value and gain are two completely different things. A dc value is an additive offset. However gain is a factor for multiplication. It should be made clear what's meant.

"fillborders" filter:

-- Documentation: Please add that the width of the borders is limited to half of the image width or height.

"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. (This is another reason why I don't use filter_complex_script)

"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 in this context.

-- 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.pgm:threshold=0.3 -show_entries  
frame=pkt_pts_time:frame_tags=lavfi.rect.x,lavfi.rect.y,lavfi.rect.score -of csv 1> log.csv
```

-- 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 a user-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.

"firequalizer" audio filter:

Documentation: Many things are unclear.

"floodfill" filter:

-- Documentation: I don't understand what this filter is doing. Especially the "source" and "destination" options are unclear.

"gblur" filter:

-- Documentation: The exact meaning of the options is unclear. Over which radius does the filter work? What does "steps" mean?

"gdigrab"

-- Feature request: 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 in the window's title line. Another problem is that the window's title may change dynamically. I suggest that the title mustn't be an exact match, however it should be sufficient if the window title contains the specified sub-string.

"geq" filter:

-- Error: If a function in the geq filter evaluates to 'nan', this result is silently changed to zero when writing to a *.pgm file. That's not correct. Either there should be an error message, or the maximum possible value (65535) could be written to indicate that there was a problem.

Can be reproduced as follows:

```
ffmpeg -f lavfi -i nullsrc=size=2x2 -vf format=pix_fmts=gray16le,geq='sqrt(-1)' -frames 1 -y test.pgm
```

Check the output file with a hex editor. It contains '00 00' for the pixel values.

See also <https://trac.ffmpeg.org/ticket/9705>

-- Error: The geq filter doesn't work correctly if the height of the frame is 1. See <https://trac.ffmpeg.org/ticket/9740>

- Documentation: "If one of the `lum_expr`, `cb_expr`, or `cr_expr` options is specified, the filter will automatically select a YCbCr colorspace." I think this sentence isn't fully correct because there is one exception. If the input format is `gray8` and if `geq=lum_expr` is used, the output format is still `gray8`, which isn't a YCbCr colorspace.
- 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.
- Feature request: Alternatively allow to specify a pixel in HSV (Hue-Saturation-Value) range.
- Feature request: I often have the case where the R, G and B channels must be set to the same value, but the expression is long and complicated. It's slow to calculate the same expression three times. Please add a mode which calculates the expression only one time and then sets the R, G and B channels to this value. A possible workaround is to calculate only the R channel with `geq`, and then use the `colorchannelmixer` filter to copy the R channel to the G and B channels.
- 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`"
- Feature request: Create two-dimensional gradients by defining the colors in the four corners of a rectangle. See also my workaround in this book.

"hsvhold" filter:

- Feature request: Please add three "similarity" parameters, one for hue, one for saturation and one for value. For example if a small hue range is required, but with a large range in saturation and value.

"hsvkey" filter:

- Feature request: Same as for `hsvhold`.

"hue" filter:

- Documentation: The range for the "`h`" and "`H`" options should be specified.
- Documentation: The "`b`" option isn't described correctly. It doesn't set the brightness, but instead the change of the brightness. Setting this option to 0

doesn't make the output dark. It leaves the brightness as it is.

"hysteresis" filter:

-- Documentation: It's totally unclear what this filter is doing. Better explanation and examples required.

"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. Better explanation and examples required.

"mix" filter:

-- Documentation: Replace "If number of weights is smaller than number of *frames* ..." by "If number of weights is smaller than number of inputs ..." . The same error is also in the "tmix" filter.

-- Documentation: The last sentence is misleading and should be changed to "By default scale is set to (1 / sum_of_weights)". The same error is also in the "tmix" filter.

"monochrome" filter:

-- Documentation: Better explanation and examples required.

"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>

New filters:

-- Feature request: A video filter that replaces transparent pixels by a non-transparent color. Usable for example after v360 filter with alpha_mask=1

option. The workaround is to overlay a second stream which contains a uniform color. This is complicated because when creating the uniform color stream, you must know the correct size in advance.

"nullsrc" video source:

-- Feature request: Please allow expressions for "size" option.

"overlay" filter

Documentation: Make clear that the variable "n" begins at 1.,

"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.

"PGM files":

-- Feature request: PGM files (required for example for "remap" filter) 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 when reading such a file.

"remap" filter:

-- Feature request: Add interpolation, use mapping files with floating point

-- Feature request: Add two user-defined colors for unmapped pixels. One color if xmap is out of range, and the other color if ymap is out of range.

"remap_opengl" filter:

-- Examples are missing. Explain why multiple format conversions may be required after this filter.

"rotate" filter:

-- The image seems to be shifted 0.5 or 1 pixel before the rotation is applied. This is only visible with "bilinear=1" which is the default setting. Not visible

with "bilinear=0". See ticket 9767. It's possible to use "v360=flat:flat:roll=..." as workaround.

"scale" filter:

-- Feature request: An option for strongly 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 loosing faint stars.

"scale2ref" filter:

-- Documentation: The first example is incomplete because the input labels are missing. Should be '[b][a]scale2ref[b][a];[a][b]overlay'

"scale_qsv" filter:

-- Documentation is missing

"sdl2" output device:

-- Documentation: The "-window_borderless" option is missing.

"selectivecolor" filter:

-- Documentation: I don't understand the difference between "absolute" and "relative" in this context. Please add a better explanation and an example to the documentation.

-- Documentation: Obviously there is no difference between absolute and relative modes, if the correction value is negative.

"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.

"showspectrum" filter:

-- Documentation: It's unclear what an "intensity color value" is.

"showwaves" filter:

-- Documentation: If all channels shall have the same color, it's sufficient to specify the color for the first channel.

-- Documentation: I suggest the following better documentation for the "draw" option:

Set the draw mode.

Available values are:

'scale' Use blending, that means the samples are drawn darker where the slope is larger. This is the default.

'full' Draw all samples with full intensity. This is useful for higher values of n .

"showwavespic" filter:

-- Feature request: Add a mode where for each audio sample only a pixel is drawn, instead of a line from +level to -level. So that the output looks like in an oscilloscope. Like the "point" and "p2p" values of the "mode" option in the "showwaves" filter.

"shufflepixels" filter:

-- Documentation: I tested this filter, but don't understand for what purpose it might be useful. Example required.

"shuffleplanes" filter:

-- Documentation: Please add that the input must have a planar pixel format. If a non-planar pixel format is used (for example rgb24), a format conversion to gbrp will be inserted automatically, which means the colors are in a different order and you would get an unexpected result.

"speechnorm" filter

-- Documentation: I didn't understand how this filter works. A clear definition for "local half cycle" is required. An example with a graphical waveform would be helpful.

"streamselect" filter:

-- Documentation: The examples are misleading. In the 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 the workaround with "sendcmd".

"superequalizer" audio filter:

-- Documentation: The unit of the options 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.

-- Undocumented feature missing in documentation: "`tmix=frames=1`" works and is a bypass mode.

-- Documentation: Replace "If number of weights is smaller than number of `frames` ..." by "If number of weights is smaller than number of inputs ...". The same error is also in the "mix" filter.

-- Documentation: The last sentence is misleading and should be changed to "By default scale is set to `(1 / sum_of_weights)`". The same error is also in the "mix" filter.

"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:

-- Revert request: After this patch <http://ffmpeg.org/pipermail/ffmpeg-cvslog/2020-October/124887.html> the yaw, pitch and roll angles are interpreted as relative angles if they are sent via sendcmd. This should be reverted, because it breaks previously correct behaviour. It's now impossible to send angles to the v360 filter with sendcmd and lerp. See also ticket 9447. It's very bad practice that the yaw, pitch and roll parameters have different meanings in the command line / if sent via sendcmd. Also this behaviour is undocumented. In my opinion "relative angle per frame" is the same as "rotational velocity" and it would be better not to re-use the same options (yaw, pitch and roll) for two different things.

-- The implementation of "h_offset" and "v_offset" options is wrong. Simply adding the offsets in x,y,z space, followed by a normalization doesn't give the correct result. The inverse algorithm is required. The output is given and the input has to be found. First the normalization must be undone by a suitable unknown factor, so that after subtracting the offsets the result is already normalized. I have successfully tested it with a remap simulation, see chapter "How to replace the v360 filter by the remap filter".

-- There are some small rounding errors in v360 if the input is "e", "flat" or "cylindrical". In most cases these errors can be neglected. See <https://trac.ffmpeg.org/ticket/9617>

-- Documentation: If specified in the command line, the v360 rotation angles are absolute angles. However if sent via sendcmd, they become relative angles. The "reset_rot" option can be used to reset the rotations to zero before new relative rotations are applied via sendcmd. "reset_rot=-1" is unclear and undocumented.

-- Documentation: For "perspective" projection, please describe the exact meaning of the option "v_fov". I'm quite sure that the current implementation of the "v_fov" option is wrong.

-- Documentation: For most options the default values aren't documented.

-- Feature request: Support 4th order polynomial fisheye projection, as described on Paul Bourke's website: <http://paulbourke.net/dome/fisheyecorrect/>

-- Feature request: "perspective" projection also for input. Could be used for making "impossible" images of the moon, as it looks from the side.

-- Feature request: For fisheye and dfisheye output, there should be an option to make the output circular and fill the outer area transparent. In most cases the outer area of the circle is not required, and it might become faster. The workaround is to overlay a circular mask, but it's uneffective to calculate the outer area and then mask it away.

"vibrance" filter:

-- Documentation: The purpose of the rbal, gbal, bbal, rlum, glum and blum options is unclear. The rlum, glum and blum values must be in the [0..1] range, this info is missing. Better explanation and/or examples required.

"vidstabtransform" filter:

-- The option "crop=black" doesn't work. See ticket 9410.

"vignette" filter:

-- Error: The effect isn't centered in the frame, if the default values for x0, y0 are used. Better use (W-1)/2 and (h-1)/2 as default values. See <https://trac.ffmpeg.org/ticket/9742>

-- Feature request: Automatically set the "aspect" option to the SAR of the input stream, if "aspect" is set to -1.

-- Feature request: Allow different functions, for example quadratic, cubic, or a user-defined exponent.

-- Documentation: Please describe which mathematical function is used for the vignette effect (is it quadratic or cubic, or something else?)

"xfade" filter:

-- Feature request: It would be nice if it could be used similar as "crossfade", 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 crossfades aren't longer than 1-2 seconds.

-- Documentation: Please add an example for the "expr" option.

-- Documentation: It's unclear if the second input stream is shifted towards the end or not. Example: Both input streams have duration 10, and offset is 5. Which frame from the second stream is used at t=7? Is it 2 or 7?

-- Documentation: What's the range of the "P" variable? From 0 to 1?

"zmq" filter:

Please add this to the documentation:

"It's possible to send zmq messages to all those filters and options that support commands. Don't send messages to other options, as this may lead to malfunction. However, 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 the output of "scale" can be used as the second input of "overlay"), but in most other cases it fails."

-- Feature request: 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_ video stream 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, so that it could be placed anywhere in the filter chain.

"zoompan" filter:

- Feature request: Please allow expressions (especially "iw" and "ih") for the "s" option. In many cases it would make sense to set the output size to a fraction of the input size.
- Feature request: Please allow zoom values smaller than 1, and use a color to fill the unused area.
- Documentation: Please note that the variable "in" begins with 1 (and not with 0).

"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 almost unusable because of missing documentation.

3 Audio processing with FFmpeg

3.1 Concat two or more audio files

```
ffmpeg -i sound1.wav -i sound2.wav -lavfi [0][1]concat=n=2:v=0:a=1 -y sound.wav  
pause
```

Note: It's important to set v=0 because the default value is 1. If you forget this setting, you would get the error message "Stream specifier '' in filtergraph description [0][1]concat=n=2:a=1 matches no streams." because FFmpeg doesn't find the video streams.

3.2 Combine multiple audio files with crossfading

```
rem Combine multiple audio files with crossfading  
  
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%
pause

```

In this example three audio files are concatenated with crossfading. 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.

3.3 Change audio volume

See <https://trac.ffmpeg.org/wiki/AudioVolume>

3.4 Change audio sample rate and number of audio channels

```

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

3.5 Change audio length, sample rate and/or pitch

Command line	Length	SR	Pitch	What has changed?
ffmpeg -f lavfi -i sine=f=1000:r=48000:d=5 s.wav	5 s	48000	1000 Hz	(This is the input file)
ffmpeg -i s.wav -af asetrate=24000,atempo=2.0,aresample=48000 t1.wav	5 s	48000	500 Hz	Pitch
ffmpeg -i s.wav -af aresample=24000 t2.wav	5 s	24000	1000 Hz	SR
ffmpeg -i s.wav -af asetrate=24000,atempo=2.0 t3.wav	5 s	24000	500 Hz	SR and Pitch
ffmpeg -i s.wav -af atempo=2.0 t4.wav	2.5 s	48000	1000 Hz	Length
ffmpeg -i s.wav -af asetrate=24000,aresample=48000 t5.wav	10 s	48000	500 Hz	Length and Pitch
ffmpeg -i s.wav -af atempo=2.0,aresample=24000 t6.wav	2.5 s	24000	1000 Hz	Length and SR
ffmpeg -i s.wav -af asetrate=24000 t7.wav	10 s	24000	500 Hz	Length, SR and Pitch
Change the length by factor L and pitch by factor P: set "L=1.2" set "P=1.5" ffmpeg -i s.wav -af atempo=1/%L%/%P%,asetrate=48000*%P%,aresample=48000 t8.wav	6 s	48000	1500 Hz	Length and Pitch

Please note that the argument of the atempo filter must be in the [0.5 ... 2.0] range.

See also the rubberband filter.

3.6 Add one or more sound effects to an audio track

In this example a short sound effect is added two times to a sound track:

```
set "T1=1"      :: Time where the first sound effect is applied
set "T2=4"      :: Time where the second sound effect is applied

rem Make a stereo input sound track with duration 8 seconds:
ffmpeg -f lavfi -i sine=500:d=8 -af volume=0.1 -ac 2 -y sound.wav

rem Make a sound effect with duration 2 seconds, including fade-in and fade-out:
ffmpeg -f lavfi -i sine=1000:d=2 -af afade=t=in:st=0:d=0.5,afade=t=out:st=1.5:d=0.5 -y soundeffect.wav

rem Apply the sound effect two times (T1 and T2) to the sound track:
ffmpeg -i sound.wav -i soundeffect.wav -i soundeffect.wav -lavfi [1]adelay=%T1%:all=1,apad[a];[2]adelay=%T2%:all=1,apad[b];[0][a][b]amix=3:normalize=0:duration=shortest -y out.wav

rem Show the waveform:
ffmpeg -i out.wav -lavfi "showwavespic=filter=peak" -y waveform.png
pause
```

Note: In the adelay filter, "s" after the time means the time is in seconds. If you omit "s", the time would be in milliseconds.

This is the output waveform of the above example:



3.7 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

pause
```

3.8 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 -f lavfi -i anullsrc=cl=stereo -shortest -y test1.mp4
ffmpeg -i mono.mp4 -f lavfi -i anullsrc=cl=stereo -shortest -y test2.mp4
ffmpeg -i stereo.mp4 -f lavfi -i anullsrc=cl=stereo -shortest -y test3.mp4

pause
```

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:

```
ffmpeg -i none.mp4 -f lavfi -i anullsrc -shortest -ac 1 -y test1.mp4  
ffmpeg -i mono.mp4 -f lavfi -i anullsrc -shortest -ac 1 -y test2.mp4  
ffmpeg -i stereo.mp4 -f lavfi -i anullsrc -shortest -ac 1 -y test3.mp4
```

This example is similar, but the output audio stream is forced to be stereo in all three cases:

```
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>

3.9 Stereo --> mix into one mono channel

Both channels of the stereo stream will be downmixed into the stream:

```
ffmpeg -i stereo.wav -ac 1 mono.wav  
  
pause
```

3.10 Check if both stereo channels are equal

In this example the difference between the left and right stereo channel is calculated and written to a mono file. If the result is silence, then both input channels are equal. The input can be a video or an audio file.

```
ffmpeg -i input.mp4 -af "aeval=val(0)-val(1)" mono.wav  
pause
```

3.11 Check if two mono inputs are equal

In this example the difference between the two mono audio channels is calculated and written to a mono file. If the result is silence, then both input channels are equal.

```
ffmpeg -i input1.wav -i input2.wav -lavfi [0][1]amerge,aeval=val(0)-val(1) -y mono.wav  
pause
```

3.12 Extract one mono channel from stereo

```
ffmpeg -i stereo.wav -filter_complex "[0:a]channelsplit=channel_layout=stereo:channels=FR[right]" -map "[right]"  
front_right.wav  
pause
```

If you only want the left channel use `FL` instead of `FR`.

See `ffmpeg -layouts` for a list of channel layouts.

If you are working with a video file, you can use `-map 0:0 -c:v copy` to preserve the video stream.

3.13 Stereo --> two mono channels

```
ffmpeg -i stereo.wav -filter_complex "[0:a]channelsplit=channel_layout=stereo[left][right]" -map "[left]" left.wav -map "[right]" right.wav  
pause
```

This command line does the same thing:

```
ffmpeg -i stereo.wav -map_channel 0.0.0 left.wav -map_channel 0.0.1 right.wav  
pause
```

3.14 Use only one channel of a stereo signal

Use the left channel of the stereo input signal for both channels of the stereo output signal:

```
rem Make a 5 seconds test sound, left channel 500 Hz sine, right channel 2000 Hz sine:  
ffmpeg -f lavfi -i sine=500 -f lavfi -i sine=2000 -lavfi [0][1]join -t 5 -y test.wav  
  
rem Use the left channel of the stereo input signal for both channels of the stereo output signal:  
ffmpeg -i test.wav -af "channelmap=0|0" -y out.wav  
pause
```

Note: If you want to use the right channel of the input signal, use "channelmap=1|1" instead.

3.15 Mono --> stereo

Of course both stereo channels will be identical.

```
ffmpeg -i input.wav -ac 2 output.wav  
pause
```

Other method for the same thing:

```
ffmpeg -i input.wav -af "channelmap=0|0" output.wav  
pause
```

3.16 Two mono channels --> stereo

```
ffmpeg -i left.mp3 -i right.mp3 -filter_complex "[0:a][1:a]join=inputs=2:channel_layout=stereo[a]" -map "[a]" output.mp3  
pause
```

3.17 Mix two stereo channels to one stereo channel

```
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  
pause
```

Or use this command line, the output may be different:

```
ffmpeg -i input1.wav -i input2.wav -filter_complex "[0:a][1:a]amerge=inputs=2[a]" -map "[a]" -ac 2 output.mp3  
pause
```

3.18 Create a file with multiple audio streams

In this example one video stream and two audio streams are mapped to the output file. The first is the unchanged input stream "-map 1:v" and the second is the modified stream with higher volume "-map[a]".

```
ffmpeg -f lavfi -i testsrc2 -f lavfi -i "sine=1k:b=2,channelmap=0|0" -lavfi "[1]volume=3[a]" -map 0:v -map 1:a -map [a]  
-t 20 -y out.mkv  
pause
```

In FFplay you can toggle between the streams with "a" button.

In VLC player you can toggle between the streams with "b" button.

3.19 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!

3.20 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.

3.21 Remove silence from an audio stream

See also: <http://ffmpeg.org/pipermail/ffmpeg-user/2021-September/053520.html>

See also: <https://medium.com/@jud.dagnall/dynamic-range-compression-for-audio-with-ffmpeg-and-comand-621fe2b1a892>

3.22 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  
pause
```

Do the same thing with a picture:

```
ffmpeg -loop 1 -i picture.jpg -i audio.mp3 -shortest out.mp4  
pause
```

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
```

3.23 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.

```
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  
  
pause
```

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.

```
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%  
  
pause
```

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:

```
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%

pause
```

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 also be made much easier with the "afreqshift" filter:

```
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

pause
```

In this example I did add the "showspectrum" filter for visualizing the spectrum:

```
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: The "afreqshift" filter doesn't operate in the frequency domain. It uses two groups of allpass sections of biquad IIR filters to do most of the work.

3.24 Record sound with the computer's built-in microphone

With this batch file you can see which microphones are available:

```
ffmpeg -list_devices 1 -f dshow -i dummy  
pause
```

With this batch file you can display the properties of a specific microphone:

```
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:

```
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"):

```
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
```

3.25 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 synchrone. 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 mpegs 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.

```
ffmpeg -an -i test.mp4 -f dshow -audio_buffer_size 100 -channels 2 -i audio="Mikrofon (Realtek High Definiti" -y -map
1:a audio.wav -map 0:v -vf scale=iw/2:-1 -f mpegs - | ffplay -f mpegs -autoexit -
```

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.

```
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 "[1:a]atrim=%OFFSET%[2],[0:a][2]amix=weights='%W1% %W2%' -y -shortest -q:v 2
-c:v mpeg4 out.mp4

pause
```

3.26 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
```

3.27 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
```

3.28 Record, process and play sound with FFplay

```
rem ffmpeg -list_devices 1 -f dshow -i dummy
rem ffmpeg -list_options 1 -f dshow -i audio="Line (USB Sound Device )"
ffplay -nodisp -f dshow -audio_buffer_size 5 -sample_rate 8000 -sample_size 16 -channels 1 -i audio="Line (USB Sound
Device )" -af aeval='val(0)'
pause
```

Note: The delay from input to output is about 80ms for one channel with 8kHz sampling rate. The delay time can be minimized by setting the audio_buffer_size to a small value (5ms), choosing the lowest possible sample rate, and disabling the graphics output with "-nodisp".

Note: The expression of the aevel filter must be encapsulated in 'single quotes'. It doesn't work if you use "double quotes" or no quotes at all.

3.29 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%[sound];[sound][1]amultiply,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%,lowpass=f=%BB%" -f nut - | ffplay -
```

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
set /a "N=4096*%F%/%SR%" :: N = 4096 * F / SR
set "AB=10"                :: Audio buffer in milliseconds
```

```
ffmpeg -f dshow -audio_buffer_size %AB% -sample_rate %SR% -sample_size 16 -channels 2 -i audio="Mikrofon (Realtek High  
Definiti" -af volume=%VOL%,afftfilt='real=if(lt(b+%N%,nb),real(b+%N%,ch),0)':'imag=if(lt(b+%N%,nb),imag(b+%N%,ch),0)' -f  
nut - | ffplay -
```

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>

3.30 Extract the audio from a video

```
ffmpeg -i video.mp4 -vn audio.mp3
```

3.31 Split a video into audio-only and video-only

Audio and video are saved in individual files.

```
ffmpeg -i input.mp4 -vcodec mpeg2video output_video.m2v -acodec copy output_audio.mp3
```

3.32 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 -c:v:0 copy output.mp4
```

For more infos about "-itsoffset", see also: <https://trac.ffmpeg.org/wiki/UnderstandingItsoffset>

See also the "compensationdelay" filter for delaying audio.

Note: Synchronizing audio with video is very easy with DaVinci Resolve.

3.33 Sources for royalty-free music

<http://opsound.org>

Eric Matyas: <http://soundimage.org/>

<https://artlist.io/>

<https://filmmusic.io>

<https://www.soundstripe.com/>

<https://freesound.org/>

Please note that "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.

These sources are not free:

<https://audiio.com/>

<https://www.premiumbeat.com/de>

3.34 Sound effects, from frequency domain to time domain

Frequency domain	Time domain	Notes
$f(t)$	$y(t) = \sin(2\pi f(t) t)$	This formula is wrong. It doesn't work this way.
$f(t)$	$y(t) = \sin(2\pi \int_0^t f(t) dt)$	This is the correct way to calculate the signal in the time domain.
$f(t) = f_0$	$y(t) = \sin(2\pi t f_0)$	Sine tone with constant frequency
$f(t) = f_0 + (f_1 - f_0) * t / p$	$y(t) = \sin(2\pi t * (f_0 + t * (f_1 - f_0) / (2\pi p)))$	Linear chirp from f_0 to f_1 in p seconds
$f(t) = f_0 * \exp(t / b * \ln(2))$	$y(t) = \sin(2\pi f_0 * b / \ln(2) * \exp(t * \ln(2) / b))$	Logarithmic chirp from f_0 , frequency doubles in b seconds
$f(t) = f_0 + f_1 * \sin(2\pi t)$	$y(t) = \sin(2\pi t * f_0 - f_1 * \cos(2\pi t))$	Sinusoidally rising and falling tone from f_0-f_1 to f_0+f_1 with a period of one second
$f(t) = f_0 + f_1 * \sin(2\pi t / p)$	$y(t) = \sin(2\pi t * f_0 - f_1 * p * \cos(2\pi t / p))$	Sinusoidally rising and falling tone with a period of p seconds

Here are a few examples:

```

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 Rectangular wave

set "F=1000"          :: Frequency in Hz
set "T=10"             :: Duration in seconds
set "VOL=1"            :: Volume
set "OUT=out.wav"      :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*(-1+2*gt(mod(%F%*t,1),0.5)):c=mono:s=48000' -t %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 Sinusoidally rising and falling tone from 400Hz to 1600Hz with a period of 2 seconds

set "F0=1000"          :: Center 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

```

```

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 "D=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*D%))) :c=stereo:s=44100' -t %D% -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 "D=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*D%))');%VOL%*(sin(ld(0))
+sin(3*ld(0))/3+sin(5*ld(0))/5+sin(7*ld(0))/7):c=stereo:s=48000' -t %D% -y %OUT%

pause

```

```

rem Linear chirp from 20Hz to 2kHz in 10 seconds
rem This is a rectangular wave

set "FF=c:\ffmpe\ffmpe"    :: Path to FFmpeg
set "F0=20"                  :: Start frequency in Hz
set "F1=2000"                 :: End frequency in Hz
set "D=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*D%))');%VOL%*lt(mod(ld(0),2*PI),PI):c=stereo:s=48000' -t
%D% -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 "D=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*D%))');%VOL%*(sin(ld(0))-sin(3*ld(0))/9+sin(5*ld(0))/25-sin(7*ld(0))/49):c=stereo:s=48000' -t %D% -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 "D=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*D%))');%VOL%*lt(mod(ld(0),2*PI),2*PI*%W):c=stereo:s=96000' -t %D% -y %OUT%

pause

```

Logarithmic chirp from 1Hz to 16384Hz in 14 seconds:

```
rem Logarithmic chirp from 1Hz to 16384Hz in 14 seconds

set "D=14"          :: Duration in seconds
set "F0=1"          :: Start frequency in Hz
set "A=1"           :: Frequency doubling time in seconds
                     :: If the stop frequency F1 and the duration D are known, A can be calculated as follows:
                     :: A = D * log(2) / log(F1/F0)
set "VOL=0.1"       :: Volume
set "OUT=out.wav"   :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*sin(2*PI*%F0%*%A%/log(2)*exp(t*log(2)/%A%)) :c=stereo:s=44100' -t %D% -y %OUT%
pause
```

Logarithmic chirp from 20Hz to 20480Hz in 20 seconds:

```
rem Logarithmic chirp from 20Hz to 20480Hz in 20 seconds

set "D=20"          :: Duration in seconds
set "F0=20"         :: Start frequency in Hz
set "A=2"           :: Frequency doubling time in seconds
                     :: If the stop frequency F1 and the duration D are known, A can be calculated as follows:
                     :: A = D * log(2) / log(F1/F0)
set "VOL=0.1"       :: Volume
set "OUT=out.wav"   :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*sin(2*PI*%F0%*%A%/log(2)*exp(t*log(2)/%A%)) :c=stereo:s=44100' -t %D% -y %OUT%
pause
```

3.35 Create a Quadrature Signal

With output to a file:

```
rem Quadrature sin/cos signal

set "F=1000"          :: Frequency in Hz
set "T=10"             :: Duration in seconds
set "V=1"              :: Volume
set "OUT=out.wav"      :: Output filename

ffmpeg -f lavfi -i aevalsrc="%V%*sin(2*PI*%F%*t) | %V%*cos(2*PI*%F%*t)":c=stereo:s=48000 -t %T% -y %OUT%
pause
```

With direct output from FFplay:

```
rem Quadrature sin/cos signal

set "F=1000"          :: Frequency in Hz
set "V=1"              :: Volume

ffplay -f lavfi -i aevalsrc="%V%*sin(2*PI*%F%*t) | %V%*cos(2*PI*%F%*t)":c=stereo:s=48000
pause
```

The output voltage from the soundcard is typically about 2.7 V_{PP} or 1.35 V_P or 0.95 V_{RMS} (with V=1).

3.36 Gravitational waves from binary black hole mergers

See also (in german): Frank Ohme, "Schwarze Löcher und andere Mysterien" <https://onlinelibrary.wiley.com/doi/epdf/10.1002/piuz.202101618>

Frequency of the gravitational wave (this is an approximation that fails near the end of the merger): (Source: see above)

$$f(t) = 1 / (8 * \pi) * (c^3 / (G * M_c))^{5/8} * (5 / dt)^{3/8}$$

with c = Speed of light, G = Gravity constant, dt = Time until merger

Note: The frequency of the gravitational wave is twice the revolution frequency of the black holes.

M_c is the "Chirp mass": $M_c = (m_1 * m_2)^{3/5} / (m_1 + m_2)^{1/5}$

Table for chirp mass (in solar masses):

m_1	m_2											
	2	3	4	5	6	8	10	12	15	20	30	50
2	1.74											
3	2.12	2.61										
4	2.42	3.01	3.48									
5	2.70	3.35	3.89	4.35								
6	2.93	3.65	4.25	4.76	5.22							
8	3.33	4.17	4.87	5.48	6.02	6.96						
10	3.67	4.61	5.40	6.08	6.79	7.78	8.71					
12	3.97	5.00	5.86	6.62	7.30	8.49	9.53	10.45				
15	4.37	5.51	6.47	7.33	8.09	9.44	10.62	11.67	13.06			
20	4.93	6.23	7.34	8.33	9.22	10.79	12.17	13.40	15.05	17.41		
30	5.83	7.39	8.73	9.93	11.01	12.95	14.65	16.19	18.25	21.24	26.12	
50	7.19	9.14	10.82	12.32	13.70	16.16	18.35	20.34	23.04	26.98	33.50	43.53

Amplitude of the gravitational wave (this is an approximation):

$$A(t) = 4 * G^{5/3} * m_1 * m_2 * (\pi * f(t))^{2/3} * (m_1 + m_2)^{-1/3} / (d * c^4)$$

with d = Distance from source to observer

The distance between the two black holes can be calculated from Kepler's 3rd law (this is an approximation):

$$r^4 = 256 * G^3 * m_1 * m_2 * (m_1 + m_2) * dt / (5 * c)^5$$

Rate of orbital decay: (Source: https://en.wikipedia.org/wiki/Gravitational_wave#Binaries)

$$dr/dt = -64 / 5 * G^3 / c^5 * (m_1 * m_2) * (m_1 + m_2) / r^3$$

Time until merger: (Source: https://en.wikipedia.org/wiki/Gravitational_wave#Binaries)

$$t = 5 / 256 * c^5 / G^3 * r^4 / ((m_1 * m_2) * (m_1 + m_2))$$

Let's try an example with two stellar black holes of 10 and 15 solar masses: (solar mass = 2e30 kg)

$$m_1 = 10 * 2e30 \text{ kg} = 2e31 \text{ kg}$$

$$m_2 = 15 * 2e30 \text{ kg} = 3e31 \text{ kg}$$

$$M_c = (m_1 * m_2)^{3/5} / (m_1 + m_2)^{1/5} = (6e62)^{3/5} / (5e31)^{1/5} = 2.12e31 \text{ kg}$$

$$\begin{aligned}f(t) &= 1 / (8 * \pi) * (c^3 / (G * M_c))^{5/8} * (5 / dt)^{3/8} \\&= K * dt^{-3/8}\end{aligned}$$

with $c = 3e8 \text{ m/s}$

$$G = 6.67e-11 \text{ m}^3 / (\text{kg s}^2)$$

$$K = 1 / (8 * \pi) * (c^3 / (G * M_c))^{5/8} * 5^{3/8} = 34.47$$

$$f(t) = 34.47 * dt^{-3/8}$$

dt	$f(dt)$	$A(t) \sim dt^{-1/4}$
0.05 s	106.0 Hz	2.11
0.1 s	81.7 Hz	1.78
0.2 s	63.0 Hz	1.50
0.5 s	44.7 Hz	1.19
1 s	34.5 Hz	1.00
2 s	26.6 Hz	0.84
5 s	18.9 Hz	0.67
10 s	14.5 Hz	0.56
20 s	11.2 Hz	0.47
60 s = 1 min	7.4 Hz	0.36
600 s = 10 min	3.1 Hz	0.20
3600 s = 1 h	1.6 Hz	0.13
86400 s = 1 d	0.5 Hz	0.06
2592000 s = 30 d	0.14 Hz	0.025
31536000 s = 365 d	0.05 Hz	0.013

Note: The frequency is proportional to $dt^{-3/8}$, the amplitude is proportional to $f^{2/3}$ and proportional to $dt^{-1/4}$ (because $-3/8 * 2/3 = -1/4$).

Note: The smallest stellar black holes have about 2 solar masses. For two such objects, $M_c=3.48e30$ kg and $K=58.29$, so that $f(1s)=58.29$ Hz

The time until merger is dt . When creating a sound where the merger takes place at $t = t_m$, we can calculate dt as follows:

$$dt = t_m - t$$

$$f(t) = K * (t_m - t)^{-3/8}$$

$$\int f(t) dt = K * -8/5 * (t_m - t)^{5/8}$$

This is the waveform as a function of time t:

$$y(t) = \sin(2 * \pi * K * 8/5 * (t_m - t)^{5/8})$$

```
rem Binary black hole merger

set "K=34.47"          :: K constant
set "S=1"                :: Frequency upscaling factor, use S=1 for realistic output
set "TM=20"              :: Time of merger
set "D=21"                :: Duration in seconds
set "VOL=0.065"           :: Volume one second before merger (before equalizer)
set "OUT=gw.wav"          :: Output filename

ffmpeg -f lavfi -i aevalsrc='%VOL%*if(lt(t,%TM%),pow((%TM%-t),-1/4)*sin(2*PI*%K%*%S%*8/5*pow((%TM%-
t),5/8))):c=stereo:s=44100' -lavfi
firequalizer=delay=0.1:gain_entry='entry(20,28);entry(25,19);entry(30,17);entry(35,6);entry(40,3);entry(45,4);entry(50,0
)' -t %D% -y %OUT%

ffmpeg -i %OUT% -lavfi showwaves=s=1800x500:n=294:mode=p2p:draw=full -frames 1 -y waveform.png

pause
```

Note: The "firequalizer" filter is used to correct the frequency response of the LD IOCA SUB18A subwoofer below 50Hz:

Frequency:	20 Hz	25 Hz	30 Hz	35 Hz	40 Hz	45 Hz	50 Hz
Gain:	+28 dB	+19 dB	+17 dB	+6 dB	+3 dB	+4 dB	+0 dB

For measuring the frequency response of the subwoofer, it's helpful to use a tone generator software, for example Audio SweepGen:
<http://www.softsea.com/review/SweepGen.html>

3.37 Create band-limited noise

```
rem Create band-limited noise from f0 to f1

set "F0=2000"          :: Lower frequency in Hz
set "F1=4000"          :: Upper frequency in Hz
set "SR=44100"         :: Sample rate in Hz
set "WS=65536"         :: Window size for FFT filter, bigger size reduces click noise
set "T=5"               :: Duration in seconds
set "VOL=.5"            :: Volume
set "OUT=out.wav"       :: Output filename

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%

pause
```

This example sends the sound directly to FFplay:

```
rem Create band-limited noise for two bands from f0 to f1 and from f2 to f3

set "F0=500"           :: Lower band start frequency in Hz
set "F1=600"           :: Lower band stop frequency in Hz
set "F2=1000"          :: Upper band start frequency in Hz
set "F3=1200"          :: Upper band stop frequency in Hz
set "SR=44100"         :: Sample rate in Hz
set "T=5"               :: Duration in seconds
set "VOL=1"             :: Volume

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 -

pause
```

```
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 anoisesrc=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

```

3.38 Show the frequency response of a filter

```

ffmpeg -f lavfi -i aevalsrc='0.1*32767*eq(0,mod(n,32768))':d=10 -lavfi
lowpass=1k,showfreqs=overlap=0:win_size=32768:win_func=rect:ascale=log -y out.mp4

pause

```

How does it work?

The "aevalsrc" source creates needle impulses with a very low frequency, in this case (sample_rate / 32768). With the default sample rate 44100Hz the impulse frequency is 1.34 Hz. These needle impulses have a continuous spectrum (this can easily be shown if you remove the lowpass filter).

3.39 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
```

3.40 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 -
```

Note: "NUL" in Windows can be replaced by "/dev/null" in Unix.

3.41 Convert an audio waveform to a picture

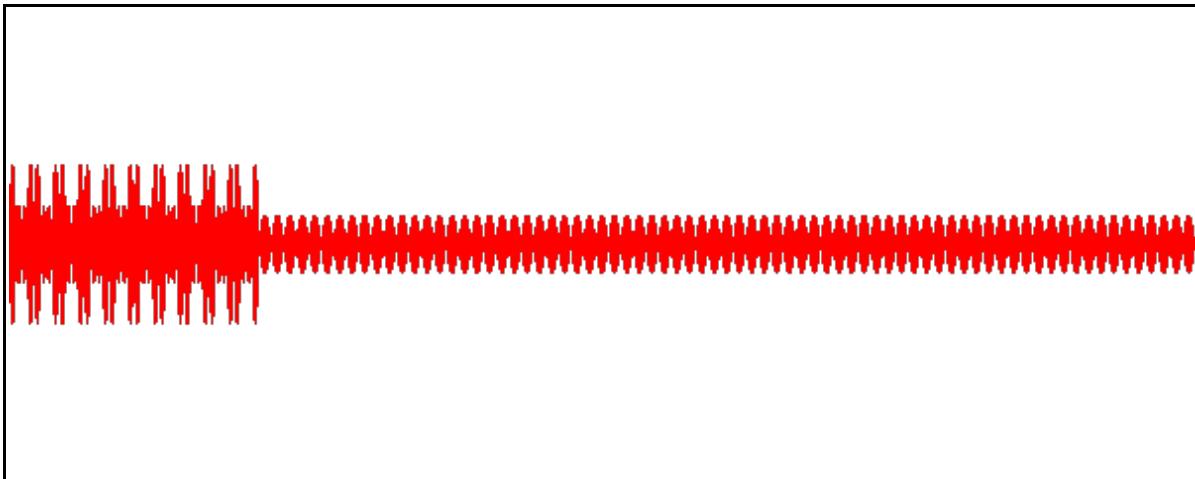
The audio waveform can be done with either of two filters: "showwaves" or "showwavespic". They both have some problems.

The straightforward solution is "showwavespic". The default image size is 600x240, but there is an option for other sizes:

```
rem Create a sample waveform:  
  
ffmpeg -f lavfi -i sine=250:b=2 -t 0.2 -y sine.wav  
  
rem Convert the waveform to an image:  
  
ffmpeg -i sine.wav -lavfi "showwavespic=filter=peak" -y waveform.png  
  
pause
```

The drawback is that "showwavespic" doesn't have a mode for drawing only samples or lines between samples. It does always draw vertical lines from +level to -level, so that the output looks as if the frequency is doubled. I recommend to use the "showwaves" filter instead.

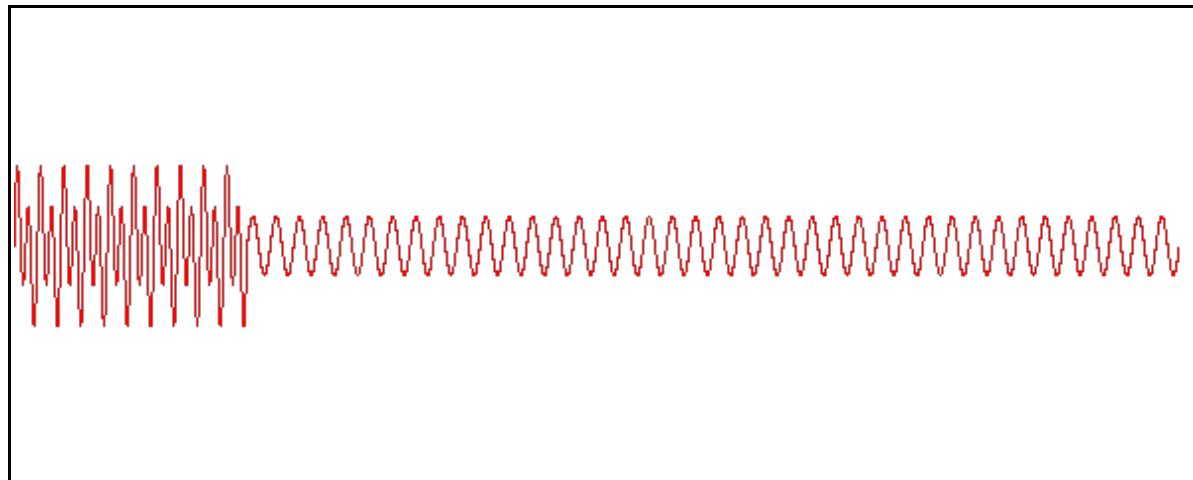
See also <https://trac.ffmpeg.org/wiki/Waveform>



If you want to see the waveform as a line (like in an oscilloscope), you must use the showwaves filter:

```
rem Create a sample waveform:  
ffmpeg -f lavfi -i sine=250:b=2 -t 0.2 -y sine.wav  
  
rem Convert the waveform to an image:  
ffmpeg -i sine.wav -lavfi showwaves=n=15:mode=p2p:draw=full -frames 1 -y waveform.png  
  
pause
```

This is the output:



The drawback of this filter is that you have to know the sample rate and length of the audio file in advance and do some calculation. In this example we have $44100 * 0.2 = 8820$ audio samples and the default width of the image is 600 pixels. This means we have $8820 / 600 = 14.7$ audio samples per pixel. This value must be rounded to an integer (in this case 15) and used for the "n" option.

Note: The "draw" option isn't well documented. Available values are:

- 'scale' Use blending, that means the samples are drawn darker where the slope is larger. This is the default.
- 'full' Draw all samples with full intensity. This is useful for higher values of n.

Alternatively you can calculate the width of the output image as follows, and then use the "size" option to set the window size:

$$\text{WIDTH} = \text{SR} * \text{T} / \text{N}$$

with **SR** = Sample rate, for example 44100

T = Duration in seconds

N = the "n" option in showwaves, audio samples per video pixel

This table lists the width of the image (for 44.1 kHz sample rate, only integer values between 100 and 10000 are shown):

n	Nyquist Frequency [Hz]	Duration [s]											
		0.1	0.2	0.5	1	2	5	10	20	30	40	50	60
1	22050	4410	8820										
2	11025	2205	4410										
3	7350	1470	2940	7350									
4	5512.5		2205										
5	4410	882	1764	4410	8820								
7	3150	630	1260	3150	6300								
9	2450	490	980	2450	4900	9800							
10	2205	441	882	2205	4410	8820							
12	1837.5		735		3675	7350							
14	1575	315	630	1575	3150	6300							
15	1470	294	588	1470	2940	5880							
18	1225	245	490	1225	2450	4900							
20	1102.5		441		2205	4410							
21	1050	210	420	1050	2100	4200							
25	882			882	1764	3528	8820						
28	787.5		315		1575	3150	7875						
30	735	147	294	735	1470	2940	7350						

35	630	126	252	630	1260	2520	6300						
36	612.5		245		1225	2450	6125						
42	525	105	210	525	1050	2100	5250						
45	490		196	490	980	1960	4900	9800					
49	450			450	900	1800	4500	9000					
50	441			441	882	1764	4410	8820					
60	367.5		147		735	1470	3675	7350					
63	350			350	700	1400	3500	7000					
70	315		126	315	630	1260	3150	6300					
75	294			294	588	1176	2940	5880					
84	262.5		105		525	1050	2625	5250					
90	245			245	490	980	2450	4900	9800				
98	225			225	450	900	2250	4500	9000				
100	220.5				441	882	2205	4410	8820				
105	210			210	420	840	2100	4200	8400				
126	175			175	350	700	1750	3500	7000				
140	157.5				315	630	1575	3150	6300	9450			
147	150			150	300	600	1500	3000	6000	9000			
150	147			147	294	588	1470	2940	5880	8820			
175	126			126	252	504	1260	2520	5040	7560			
180	122.5				245	490	1225	2450	4900	7350	9800		
196	112.5				225	450	1125	2250	4500	6750	9000		
210	105			105	210	420	1050	2100	4200	6300	8400		
225	98				196	392	980	1960	3920	5880	7840	9800	
245	90				180	360	900	1800	3600	5400	7200	9000	
252	87.5				175	350	875	1750	3500	5250	7000	8750	

294	75				150	300	750	1500	3000	4500	6000	7500	9000
300	73.5				147	294	735	1470	2940	4410	5880	7350	8820
315	70				140	280	700	1400	2800	4200	5600	7000	8400
350	63				126	252	630	1260	2520	3780	5040	6300	7560
420	52.5				105	210	525	1050	2100	3150	4200	5250	6300
441	50				100	200	500	1000	2000	3000	4000	5000	6000

This table lists the width of the image (for 48 kHz sample rate, only integer values between 100 and 10000 are shown):

n	Nyquist Frequency [Hz]	Duration [s]											
		0.1	0.2	0.5	1	2	5	10	20	30	40	50	60
1	24000	4800	9600										
2	12000	2400	4800										
3	8000	1600	3200	8000									
4	6000	1200	2400	6000									
5	4800	960	1920	4800	9600								
6	4000	800	1600	4000	8000								
8	3000	600	1200	3000	6000								
10	2400	480	960	2400	4800	9600							
12	2000	400	800	2000	4000	8000							
15	1600	320	640	1600	3200	6400							
16	1500	300	600	1500	3000	6000							
20	1200	240	480	1200	2400	4800							
24	1000	200	400	1000	2000	4000	10000						
25	960	192	384	960	1920	3840	9600						
30	800	160	320	800	1600	3200	8000						

32	750	150	300	750	1500	3000	7500						
40	600	120	240	600	1200	2400	6000						
48	500	100	200	500	1000	2000	5000	10000					
50	480		192	480	960	1920	4800	9600					
60	400		160	400	800	1600	4000	8000					
64	375		150	375	750	1500	3750	7500					
75	320		128	320	640	1280	3200	6400					
80	300		120	300	600	1200	3000	6000					
96	250		100	250	500	1000	2500	5000	10000				
100	240			240	480	960	2400	4800	9600				
120	200			200	400	800	2000	4000	8000				
125	192			192	384	768	1920	3840	7680				
128	187.5				375	750	1875	3750	7500				
150	160			160	320	640	1600	3200	6400	9600			
160	150			150	300	600	1500	3000	6000	9000			
192	125			125	250	500	1250	2500	5000	7500	10000		
200	120			120	240	480	1200	2400	4800	7200	9600		
240	100			100	200	400	1000	2000	4000	6000	8000	10000	
250	96				192	384	960	1920	3840	5760	7680	9600	
300	80				160	320	800	1600	3200	4800	6400	8000	9600
320	75				150	300	750	1500	3000	4500	6000	7500	9000
375	64				128	256	640	1280	2560	3840	5120	6400	7680
384	62.5				125	250	625	1250	2500	3750	5000	6250	7500
400	60				120	240	600	1200	2400	3600	4800	6000	7200
480	50				100	200	500	1000	2000	3000	4000	5000	6000

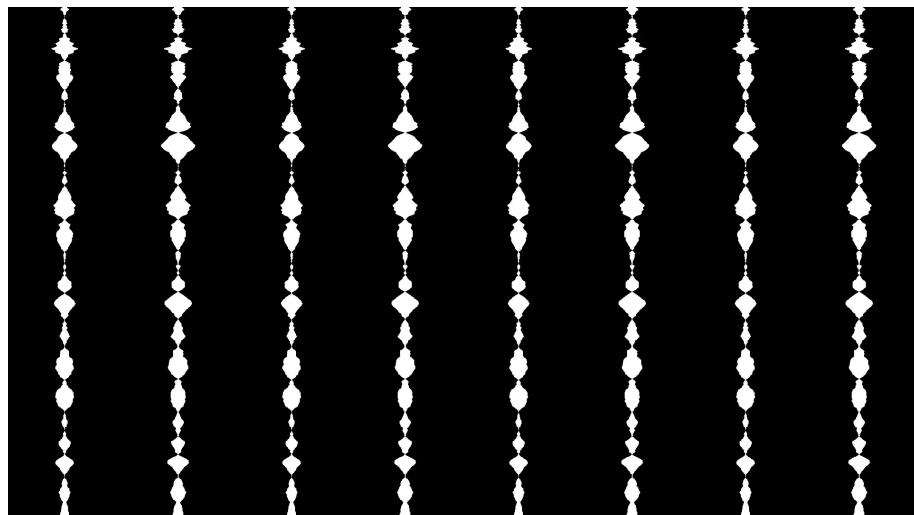
3.42 Optical sound effect

This batch file converts an audio file into a video with optical sound effect, like in the intro of Russ Meyer's film "Faster, Pussycat! Kill! Kill!":

<https://www.youtube.com/watch?v=LmeGKP5Bb20>

The two stereo channels are duplicated to a total of 8 channels.

```
ffmpeg -i blueflowers.mp3 -lavfi  
"showwaves=mode=cline:split_channels=true:s=1080x480:colors=white,transpose,split=4,hstack=4" -y out.mp4  
  
pause
```



In this improved version the optical sound tracks appear one after the other and have individual delay times:

```
ffmpeg -i blueflowers.mp3 -lavfi "asplit=4[a0][a1][a2][a3];  
[a0]asplit[b0][c0];  
[a1]adelay=0.05:all=1,volume='gt(t,5)':eval=frame,asplit[b1][c1];  
[a2]adelay=0.10:all=1,volume='gt(t,10)':eval=frame,asplit[b2][c2];  
[a3]adelay=0.15:all=1,volume='gt(t,15)':eval=frame,asplit[b3][c3];  
[b0]showwaves=mode=cline:split_channels=true:s=1080x480:colors=white[v0];  
[b1]showwaves=mode=cline:split_channels=true:s=1080x480:colors=white[v1];  
[b2]showwaves=mode=cline:split_channels=true:s=1080x480:colors=white[v2];  
[b3]showwaves=mode=cline:split_channels=true:s=1080x480:colors=white[v3];  
[v0][v1][v2][v3]vstack=4,transpose;[c0][c1][c2][c3]amix=4" -y out.mp4  
  
pause
```

Note: The linefeeds were inserted only for clarity. Of course everything must be written in one line.

3.43 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 microphones 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.

3.44 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.

```
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
```

3.45 The "avsynctest" source

It's unclear why [out1] is required. If [out0] is used instead, or no output label at all, it gives an error message.

```
ffmpeg -f lavfi -i avsynctest=d=10[out1] out.mov

pause
```

3.46 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.

Microphone	Measured level of 1kHz sine	Measured noise level	Sensitivity in manufacturer's data sheet	Noise level in manufacturer's data sheet
Rode NTG2	RMS level dB: -22.4	RMS level dB: -40.4	-36.0 dB re 1 Volt/Pascal +2dB @ 1kHz	18 dBA
Rode NT1	RMS level dB: -12.1	RMS level dB: -41.3	-29.0 dB re 1 Volt/Pascal +2dB @ 1kHz	4 dBA

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:

```
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
```

3.47 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, 180, 196, 210, 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, 16, 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

Divisors of 300 (Common divisors of 48000 and 44100):

1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 25, 30, 50, 60, 75, 100, 150, 300

Framerate	Ticks of 1/90000s clock per frame (Mpeg TS system clock timebase)	Audio samples per frame @44.1 kHz	Audio samples per frame @48 kHz
23.976 fps = 24000 / 1001	3753.75	1839.3375	2002
24 fps	3750	1837.5	2000
25 fps	3600	1764	1920
29.97 fps = 30000 / 1001	3003	1471.47	1601.6
50 fps	1800	882	960
59.94 fps = 60000 / 1001	1501.5	735.735	800.8

3.48 Speed of sound

Speed of sound in air at 20°C: 343.2 m/s

Time [ms]	Distance [m]	Samples @ 44100Hz	Samples @ 48000Hz	Notes
0.022675	0.00778	1		
0.020833	0.00715		1	
1	0.3432	44.1	48	
2	0.6864	88.2	96	
2.2675	0.7782	100		
2.9138	1	128.497	139.860	
3.3333	1.144	147	160	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
5	1.716	220.5	240	
5.8275	2	256.699	279.720	
10	3.432	441	480	Both sample numbers are integers
14.569	5	642.483	699.301	
29.138	10	1284.97	1398.60	
100	34.32	4410	4800	Both sample numbers are integers
1000	343.2	44100	48000	Both sample numbers are integers

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:

```
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>

By the way, it's also possible to let FFmpeg examine a file.

To see whether FFmpeg recognizes the file as something:

```
ffmpeg -i myfile.***  
pause
```

To see whether FFmpeg can decode the file:

```
ffmpeg -i myfile.*** -f null -  
pause
```

This is an example for writing the "noise floor count" of an audio file to a CSV log file:

```
ffprobe -f lavfi -i amovie=in.mp3,astats=metadata=1 -show_entries tags=lavfi.astats.Overall.Noise_floor_count -of  
csv=p=0 1> log.csv  
pause
```

4.1 Count the number of frames

```
ffprobe -v error -count_frames -select_streams v:0 -show_entries stream=nb_read_frames -of  
default=nokey=1:noprint_wrappers=1 input.mp4  
  
pause
```

Get the total number of frames of a video: <https://stackoverflow.com/questions/2017843/fetch-frame-count-with-ffmpeg>

4.2 Find the keyframe timestamps

```
ffprobe -select_streams V:0 -show_frames -skip_frame nokey -show_entries frame=best_effort_timestamp_time input.mp4  
  
pause
```

5 FFplay

Keyboard commands while playing:

Key	Notes
q, ESC	Quit
f, left mouse double-click	Toggle full screen
p, SPACE	Pause
s	Step to the next frame. Pause if the stream is not already paused, step to the next video frame, and pause.
m	Toggle mute.
9, 0	Decrease and increase volume respectively.
/, *	Decrease and increase volume respectively.
a	Cycle audio channel in the current program.
v	Cycle video channel.
t	Cycle subtitle channel in the current program.
c	Cycle program.
w	Cycle video filters or show modes.
left/right	Seek backward/forward 10 seconds.
down/up	Seek backward/forward 1 minute.
page down/page up	Seek to the previous/next chapter, or if there are no chapters seek backward/forward 10 minutes.
right mouse click	Seek to percentage in file corresponding to fraction of width.

This is a batch file that you can put on your desktop, and then play a video simply by drag-and-drop:

```
ffplay %1 -autoexit
```

Note: Contrary to FFmpeg, FFplay doesn't need "-i" before the input.

List of the most important FFplay options:

-video_size	Set frame size (WxH or abbreviation) Note: This is different from FFmpeg, where the option is -s . The "-video_size" option is used to manually tell ffplay the size for videos that do not contain a header (such as raw video). It is not used to resize videos. For resizing use either the -x and -y options or the "scale" filter.
-fs	Start in fullscreen mode
-x	Force displayed width
-y	Force displayed height
-left	Set the x position for the left of the window (default is a centered window).
-top	Set the y position for the top of the window (default is a centered window).
-an	Disable audio
-vn	Disable video
-sn	Disable subtitles
-ss pos	Seek to pos
-t	Duration of video (Note: -loop has higher priority than the -t option)
-loop	Defines how often the video is repeated, 0 means infinity (Note: -loop has higher priority than the -t option)
-nodisp	Disable graphical display
-noborder	Borderless window
-alwaysontop	Window always on top
-f fmt	Force format
-loop number	Loops movie playback <number> times. 0 means forever
-vf filtergraph	Create the filtergraph specified by <i>filtergraph</i> and use it to filter the video stream Note: FFplay doesn't allow -filter_complex
-af filtergraph	<i>filtergraph</i> is a description of the filtergraph to apply to the input audio
-autoexit	Exit when video is done playing
-exitonkeydown	Exit if any key is pressed
-exitonmousedown	Exit if any mouse button is pressed

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
```

This batch file is for showing images (drag-and-drop) for infinite duration:

```
ffplay -loop 0 %1
```

This batch file is for showing images (drag-and-drop) for 5 seconds:

```
ffplay -autoexit -loop 125 %1
```

Note: The default framerate is 25.

The same thing can also be done with "loop" and "trim" filters. In this case the multiplication by 25 isn't required:

```
ffplay -vf loop=-1:1,trim=duration=5 -autoexit %1
```

Infinitely repeat the first 2 seconds of the video:

```
ffplay -loop 0 -t 2 %1
```

Note: -loop has the higher priority, -t affects the play interval of the media. This is different from FFmpeg, where the -t option has the higher priority.

Repeat the first 2 seconds of the video 5 times:

```
ffplay -loop 5 -t 2 %1
```

Show UHD 4K spherical test pattern on extended desktop (UHD 4K beamer):

```
c:\ffmpeg\ffplay 2160.png -left 1920 -top 0
```

```
pause
```

6 DaVinci Resolve 15 / 16 / 17

Why DaVinci Resolve and not FFmpeg? FFmpeg has no graphical user interface. For film cutting you have to see what you are doing. These are two programs for different tasks, and they complement each other very well.

DaVinci Resolve is a very complex program with so many functions that you don't know where to begin.

<https://www.blackmagicdesign.com/de/products/davinciresolve/>

Note: The Downloads are described as "Updates", which is misleading. In fact all available downloads are complete versions and don't require any other version installed before.

I got this book: Paul Saccone, Dion Scopettuolo: "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 manuals for versions 16 and 17 are extremely long, more than 3000 pages.

DaVinci Resolve Project Server: This is a tool for working on a project with multiple persons. If you are the only person working on your project, then you don't need it and the icon can be deleted from the desktop.

DaVinci Resolve 16 manual: https://documents.blackmagicdesign.com/UserManuals/DaVinci_Resolve_16_Reference_Manual.pdf

DaVinci Resolve 17 new features guide: https://documents.blackmagicdesign.com/SupportNotes/DaVinci_Resolve_17_New_Features_Guide.pdf

Fusion 9 user manual: https://documents.blackmagicdesign.com/UserManuals/Fusion9_Tool_Reference.pdf

Fusion 16 user manual: It's included as a PDF when you download the latest version of Fusion 17 here:

<https://www.blackmagicdesign.com/support/family/davinci-resolve-and-fusion>

6.1 Tutorials on Youtube

Tutorials by Gunter Wegner (in german language):

Davinci Resolve Tutorial - Folge 1 - Warum Resolve und wichtige Grundeinstellungen <https://www.youtube.com/watch?v=Hwu9yxPcOr0>

Davinci Resolve Tutorial - Folge 2 - Medien: Organisation, Sichtung, Selektion <https://www.youtube.com/watch?v=K-n9nRs8Fcs>

Davinci Resolve Tutorial - Folge 3 - Dramaturgie und Grobschnitt <https://www.youtube.com/watch?v=V-PfQYBZ8Ow>

Davinci Resolve Tutorial - Folge 4 - Feinschnitt, Sprecher und Audio-Bearbeitung https://www.youtube.com/watch?v=yytWIk_SL5M

Davinci Resolve Tutorial - Folge 5 - Grundlagen Farbbearbeitung / Color Grading https://www.youtube.com/watch?v=UzhNOKgu_8g

Davinci Resolve Tutorial - Folge 6 - Titel, Keyframes und Tracking <https://www.youtube.com/watch?v=E5o2bqNII2w>

Davinci Resolve Tutorial - Folge 7 - Video Export und Rendern für Youtube, das Deliver Modul <https://www.youtube.com/watch?v=7jnV1JhzJgQ>

Davinci Resolve: Tutorials, VLOGs und Screencasts effizienter schneiden im Cut-Modul <https://www.youtube.com/watch?v=lnSWVydG7a8>

Note: "VLOG" does here mean "Video Log" and not "Panasonic V-Log".

Davinci Resolve - Eure Fragen 1 - Multiple In/Out, Foto-Import, auf Takt schneiden <https://www.youtube.com/watch?v=RWWEFWqWcts>

Davinci Resolve - Eure Fragen 2 - Projektmanagement, Videos auf dem TV abspielen, Powerbins <https://www.youtube.com/watch?v=tp3FDbK8smw>

Davinci Resolve - Alpha-Masken, Transparenzen und Qualifier <https://www.youtube.com/watch?v=Nbk01sql1Xw>

Farbmanagement, Monitorprofilierung und Kalibrierung - Warum und Wie einfach erklärt! https://www.youtube.com/watch?v=k0ZOY_O4HQA

<https://gwegner.de/know-how/farbmanagement-tutorial-teil-1-grundlagen-farbraeume-farbprofile-und-warum-das-alles/>

<https://gwegner.de/know-how/farbmanagement-tutorial-teil-2-korrektes-farbmanagement-in-anwendungen/>

<https://gwegner.de/know-how/monitor-kalibrieren-spyderx-video/>

<https://gwegner.de/know-how/srgb-oder-adobe-rgb-in-der-kamera-einstellen-und-welchen-bildstil/>

360 Grad Panorama und Little Planet zusammensetzen - Zenit und Nadir korrigieren | gwegner.de <https://www.youtube.com/watch?v=gwQB6to6ttk>

Other tutorials:

Creating and Installing LUTS in Davinci Resolve 16: https://www.youtube.com/watch?v=NuZTCIZX_T0&feature=youtu.be

Speed Editor QUICK tips PLUS - can I use it in the Color Page? <https://www.youtube.com/watch?v=WpJFzBZGaHg&feature=youtu.be>

3D camera tracking: https://www.youtube.com/watch?v=Dos_TTHujwE

Multicam editing tool: https://www.youtube.com/watch?v=ahn_M87thok

High dynamic range: <https://www.youtube.com/watch?v=aoS-yLYtOh8>

DaVinci Resolve 17 - die neuen Funktionen im ausführlichen Überblick <https://www.youtube.com/watch?v=2ABmCxkQhQI>

Tracker in DaVinci Resolve 17 - Bewegung von Objekten im tracken <https://www.youtube.com/watch?v=TJORG1HOwMM>

How to Blur Faces or Objects in DaVinci Resolve 16 | Tutorial <https://www.youtube.com/watch?v=omKYEtqu3Ko>

3D Reflection Mapping or Environment Mapping in DaVinci Resolve 16 <https://www.youtube.com/watch?v=SnAjBanXUVA>

Edit 360° Video w/ DaVinci Resolve - Tripod Removal, PanoMap, Spherical Stabilizer + Ignite Pro <https://www.youtube.com/watch?v=xIOhluai5mk>

Davinci Resolve/Fusion - Awesome Energy Sphere tutorial <https://www.youtube.com/watch?v=pUU7eWwWI4o>

Many tutorials can be found here: <https://motionarray.com/learn/davinci-resolve/tutorials/>

6.2 Mouse buttons and keyboard shortcuts

Icons	Keyboard or mouse	Description
	Middle mouse button	Move the whole timeline to the left or right
< ● >		(seems to be the same as moving the position with the mouse)
◀◀		Jump to the start
◀	J	Play backward
	J+K	Play backward with half speed
	Hold K and press J once, or press ARROW LEFT	One frame backward
■	K	Stop
	Hold K and press L once, or press ARROW RIGHT	One frame forward
	K+L	Play forward with half speed
▶	L	Play forward
	press L twice	Play with double speed
▶▶		Jump to the end
↔		Endless loop mode
	Numpad /	Play around selection
	Space	Play / Pause
▶	I	Set the "In" marker
◀	O	Set the "Out" marker
.		Fine adjustment, one frame to the right
,		Fine adjustment, one frame to the left
F9		Insert

	F10	Overwrite
	F11	Replace
	F12	Place on top
	SHIFT+F10	Ripple Overwrite
	SHIFT+F11	Fit to Fill
	SHIFT+F12	Append at End
	A	Selection Mode
	B	Blade Edit Mode
	D	In "Edit" page: Toggle the selected clip on/off
	M	If a clip is selected: Set a marker inside the clip If no clip is selected: Set a marker in the timeline
	MM	Enter a name and color for the marker
	N	Toggle "Snapping" tool (that's the magnet symbol)
	Q	Toggle between source and timeline viewer View → Source/Timeline Viewer
	T	Trim Edit Mode
	W	Toggle "Dynamic Trim Mode"
	X	Select a clip (or gap) under the playhead and add In and Out points to the timeline. If the wrong track is selected, deactivate the "Auto track selector" in one or more tracks.
	Z	In "Color" page: Zoom to fit window In "Fusion" page: View depth channel
	SHIFT+SPACE	In "Fusion" page, opens a search window for adding new tools
	CTRL+F	Toggle fullscreen mode, this works only in "Edit" and "Color" page.
	CTRL+P	Toggle a node on/off in "Fusion" page
	BACKSPACE	If DEL key deletes too much, use undo and then try the backspace key
	CTRL+W	Toggle wipe mode on/off
	CTRL+Z	Undo Warning: You can't undo things from a prior session.

Note: The symbols were taken from the "Segoe UI Symbol" character set.

Note: You can define your own keyboard shortcuts with DaVinci_Resolve --> Keyboard_Customization

See also this keyboard shortcut sheet from Bernd Klimm: <https://vfxstudy.com/wp-content/uploads/2020/02/Fusion-Keyboard-Shortcuts-v1.pdf>

6.3 GUI Controls

	Here you can control the volume and mute the speakers. It doesn't change the volume of the output video.

6.4 Preferences

Preferences can be set in DaVinci Resolve --> Preferences:

- System --> Memory and GPU Leave 4GB for other applications
- System --> Media Storage Use the fastest drive that you have on your computer
- User --> Project Save and Load Use "Live Save" and "Project Backups"
- User --> Project Save and Load Select the "Project backup location"
- User --> Editing --> New Timeline Settings --> Start Timecode = 00:00:00:00
- User --> Editing Set the "Standard Still Duration"

You can also reset all user preferences to the factory default by clicking on the ●●● symbol in the top right corner and then use "Reset User Preferences...".

If Playback --> Render Cache is set to "User", you can right-click on a clip and use "Render Cache Color Output".

6.5 Project settings

The timeline framerate in the project settings must be set before any media are imported. The timeline resolution can be changed later.

The project settings are in File --> Project Settings, or click on the ☰ symbol in the lower right corner.

- Set the timeline resolution, timeline framerate and playback framerate
- Master Settings: scroll down to "Optimized Media and Render Cache":
 - Set the "Optimized media resolution" to half or quarter, that's the proxy resolution that's used for editing
 - Tick "Enable background caching after 7 seconds"
 - Tick "Automatically cache transitions in user mode"
 - Tick "Automatically cache composites in user mode"
 - Tick "Automatically cache Fusion Effects in user mode"
- Color Management --> Color Space & Transforms
 - Color science: "DaVinci YRGB" or "DaVinci YRGB Color Managed"
 - Color Processing Mode: "SDR" or "HDR"
 - Timeline Color Space: Rec.709 Gamma 2.4
 - In this menu it's also possible to set LUTs
- Image Scaling --> Input Scaling
 - Set "Mismatched resolution files" to "Center crop with no resizing"

Timeline --> Selection follows Playhead should be deactivated.

Playback --> Render_Cache should be set to "Smart".

Note: It's possible to reset the user interface with Workspace --> Reset UI Layout

6.6 Timeline Proxy Mode and Timeline Resolution

Two things can be done for faster editing:

1. You can set Playback --> Timeline_Proxy_Mode to "Half Resolution" or "Quarter Resolution". It's not necessary to set it back to the desired output size before delivering the final video. The output size is set in File --> Project_Settings. This is the recommended method.
2. The timeline resolution (in File --> Project_Settings) can be set to a smaller size. Don't forget that you must set it back to the desired output size before you deliver the final video. Otherwise the video would be scaled up from the smaller to the larger resolution, which means a loss of detail. A drawback of this method is that objects may be shifted a little bit in Fusion.

See also the DR17 manual, part 2, chapter 7, page 159ff: "Improving Performance, Proxies, and the Render Cache"

6.7 The "Media" page in DaVinci Resolve

This page is used for adding media to the project. The media files (video and audio) remain in their original folders and are not changed or overwritten. DaVinci Resolve is completely non-destructive with respect to the media files.

A new project can be created with File --> New Project

In the media pool, if the media files are shown as pictures (click on the :::: symbol), you can change the sorting order by clicking on the ⚡↓ symbol.

In the listing mode (click on the ≡ symbol) sorting isn't possible.

Make a new bin: Right click in the master bin, then "New Bin". A "Bin" is a virtual folder.

"Power Bins" are bins that are available in all your projects.

"Smart Bins" are bins which contain files that fulfill certain search criteria.

Add files to media pool while keeping the original sub-folders:

Select all source folders, right-click and select "Add Folder and SubFolders into Media Pool (Create Bins)".

If you have a numbered sequence of JPG, CR2 or DNG images (*.jpg, *.cr2, *.dng), you can drag and drop them to the timeline as a video. Click in the media pool on the ●●● symbol and set "Frame Display Mode" to "Sequence". If you want individual images instead, set "Frame Display Mode" to "Individual".

Note: Symbols were found in "Segoe UI Black" and "Segoe UI Symbol" character sets.

6.8 Archive projects

File --> Project Manager opens the project manager. Alternatively you can click on the "Project Manager" icon in the bottom right corner.  In the project manager you can right-click on a project and then choose "Export Project Archive". A *.dra archive file contains all files that were used in the project. Usable for example if you want to give the whole project to someone else.

A project can be restored by right-clicking on "Untitled project" and then using "Restore Project Archive".

You can generate new folders by right-clicking in an empty space. Projects can be moved by drag and drop.

Projects can be imported by loading a *.drp file. These files contain only the timeline and not all the media files. It's expected that the media files are already at their correct places. If some or all media is shown as "offline", click on the "Relink Media" icon in the Media Pool (that's the broken chain symbol), then select the correct folder.

To import a *.drf timeline file, right-click in the Media Pool, then use Timelines --> Import --> AAF_EDL_XML_DRT_ADL...

6.9 The "Cut" page in DaVinci Resolve

It's not really necessary to use the "Cut" page for editing a video, as most things can also be done in the "Edit" page.

The Speed Editor is optimized for the "Cut" page.

Introduction for Speed Editor: <https://www.blackmagicdesign.com/de/products/davinciresolve/keyboard>

The "Cut" page is useful if you have one long video and want to cut out many sections.

This is very efficient because you don't need the mouse. It can all be done with these keyboard shortcuts:

Ctrl-B	Cut at the playhead.
Shift-V	Select the clip at the playhead.
Backspace	Delete the selected section of the clip. The rest is shifted towards left.
Left / right arrows	Move the playhead one frame.
Shift left / right arrows	Move the playhead in bigger steps.
Up / down arrows	Move the playhead to the next / previous clip.
Ctrl-Y	Select everything to the left.
Alt-Y	Select everything to the right.

A "smooth cut" inserts motion interpolation.

Boring detector: This is the "zzz" icon at the left side in the "Cut" page. It analyzes the lengths of the clips in a timeline.

See also <https://www.youtube.com/watch?v=yyRfGX2mnPs>

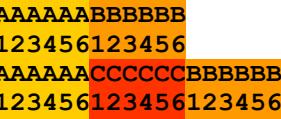
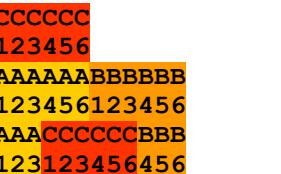
6.10 The "Edit" page in DaVinci Resolve

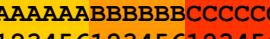
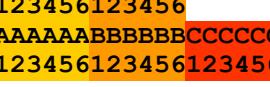
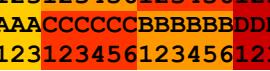
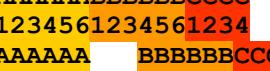
The visibility of the timeline is from above for the video clips and from below for the audio clips!

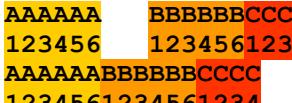
If there is a white line above the timeline, this acts as a protection for the timeline below. Any changes will only be applied to the selected part where the white line is visible. It can be deleted with **Mark → Clear_In_and_Out**

Note: The **DEL** and **BACKSPACE** keys have different meanings.

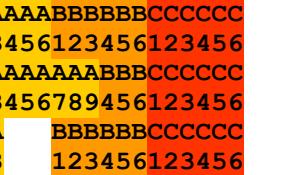
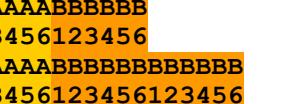
These things are always possible (in "Selection" mode and also in "Trim Edit" mode):

Insert	new clip: 	Drag the new clip (here clip C) and then press CTRL+SHIFT while dropping it in the timeline. The clip B is not overwritten. Alternatively you can place the playhead at the correct position, then select the new clip in the media pool and press F9 . Alternatively you can drag the new clip over the clip window and release the mouse button over the "Insert" icon.
	before: 	
	after: 	
Overwrite	new clip: 	Place the playhead where the new clip shall begin (here in the middle of clip A). Select a clip in the media pool (here clip C) and press F10 . The clip will overwrite parts of clips A and B. Alternatively you can drag the new clip over the clip window and release the mouse button over the "Overwrite" icon.
Replace	new clip: 	Place the playhead where the new clip shall begin (here at the end of clip A). Select the new clip in the media pool (here clip D) and press F11 . The total length of the timeline doesn't change. If the new clip is shorter than the old clip, it will overwrite the beginning of the old clip. If the new clip is longer than the old clip, it will be cut to the old clip's length. Alternatively you can drag the new clip over the clip window and release the mouse button over the "Replace" icon. You can place the playhead at a specific point in the source window, and the other playhead over a specific point in the timeline. When you use replace, these points will match.
Ripple Delete	before:  after: 	Select a clip (here clip B) and press DEL . The gap is automatically closed.

Delete clip, leave gap	before:  after: 	Select a clip (here clip B) and press BACKSPACE . A gap remains.
Roll	before:  after: 	To roll an edit, moving the Out point of the outgoing clip and the In point of the incoming clip at the same time, drag an edit point between two clips to the left or right.
Ripple Overwrite	new clip:  before:  after: 	Replaces a clip by another clip of different length. The total length of the timeline will change. Place the playhead where the new clip shall begin (here after clip A). Select a clip in the media pool (here clip D) and press SHIFT+F10 . Alternatively you can drag the new clip over the clip window and release the mouse button over the "Ripple Overwrite" icon.
Fit to Fill	new clip:  before:  after: 	Place the playhead at the beginning of a gap (here after clip A), select a clip in the media pool (here clip C) and press SHIFT+F11 . The clip will be inserted and its speed will be adjusted so that it fits in the gap. Alternatively you can drag the new clip over the clip window and release the mouse button over the "Fit to Fill" icon.
Append at End	new clip:  before:  after: 	Select a clip in the media pool (here clip B) and press SHIFT+F12 . The clip will be appended at the end of the timeline. The playhead position doesn't care. Alternatively you can drag the new clip over the clip window and release the mouse button over the "Append at End" icon.
Exchange clips	before:  after: 	Drag the clip (here clip B) and press CTRL+SHIFT while dropping it in the new position in the timeline (here between clips C and D). The snapping tool should be activated. You can also select a clip and press CTRL+SHIFT+ , to move it to the right or CTRL+SHIFT- , to move it to the left.
Insert gap	before:  after: 	Set the playhead to the first clip that you want to move to the right (here clip B), then press Y . This will select the current clip and all clips to the right. Then move the clips to the right.

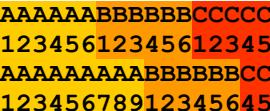
Delete gap	before: 123456		Select the gap and press DEL or BACKSPACE . You can also select multiple clips with gaps and use Edit → Delete Gaps
	after: 123456		

These things are possible in Selection Mode (Click the "Selection" ↗ icon, or press the A key)

Moving	before: 123456123456		To move clips, drag any clip in the timeline to any other position. If you drag a clip to overlap another clip, the clip you're dragging overwrites the clip you're dropping it onto (here clip C). To move clips in the timeline up or down to other tracks while keeping them at the same time: Hold the SHIFT key down while dragging clips.
Resize	before: 123456123456		To shorten or lengthen clips: Move the Selection Mode pointer over the beginning or end of a clip, and when it turns into the Resize cursor, drag the In or Out point to the left or right to change the clip's length. It will either overwrite another clip, or a gap appears.
Duplicate clip	before: 123456123456		Hold the ALT key and drag and drop the duplicated clip to the new position. This works also for audio clips.

These things are possible in Trim Edit Mode (Click the "Trim Edit" ✎ icon, or press the T key)

Ripple	before: 123456123456		To ripple the outgoing or incoming part of an edit to add or remove media to a clip while simultaneously moving all other clips at the right in the timeline to make room, click the Trim tool, and drag an edit point to a new position in the timeline.
Slip	before: 123456123456		To slip a clip's range of content without changing its position in the timeline, click the middle top region of a clip, and then drag to the left or right to "slip" the clip to contain a different range of frames. A dashed overlay shows the total duration of media available for you to slip with, which moves left and right as you drag.

Slide	before:  after: 	<p>To slide a clip, moving it to another position in the timeline while simultaneously adjusting the Out point of the previous clip and the In point of the next clip to accommodate the change in position of the current clip being dragged, click the bottom-middle name bar of the clip and drag it to another position.</p>
-------	---	--

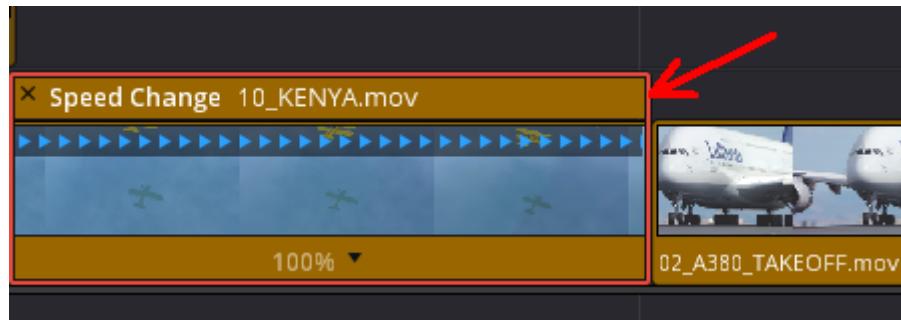
Dynamic trim mode: This does only mean that a small interval around the playhead is played repeatedly. It has nothing to do with trimming. The durations can be set in Preferences --> User --> Editing --> Pre-roll time and Post-roll time

Toggle between one / two clip windows: Click on  and  symbols. Or toggle Workspace --> Single Viewer Mode

You can copy attributes from one clip to another. Select the source clip, right-click --> Copy, then select the target clip, right-click --> Paste_Attributes. Then you can select which attributes you want to copy.

6.11 Speed Change

If you want to change the speed of a clip, make a right-click on the clip in the timeline and select "Retime Controls" or press **CTRL-R**. The clip will get higher with "Speed Change" visible in the headline. Also it gets blue arrows. Pick the clip at the top right corner (see the red arrow) to change its length and speed. When the speed is above 100%, the blue arrows will become more dense, and if the speed is below 100% the arrows will become yellow.



In the inspector you can also reverse the speed or set the speed to zero.

More options can be set in the inspector under "Retime and Scaling", for example you can set "Retime Process" to "Optical Flow".

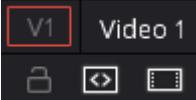
The default for these settings can be set in `Project_Settings --> Master_Settings --> Frame_Interpolation`

Note: Activate "Ripple Timeline" to avoid that a gap is created after the clip.

6.12 Change the framerate of a clip

Right-click on a clip, select "Clip Attributes" and set the framerate to the same value as the timeline framerate. The clip will then play with a different speed and have a different length. No frames will be duplicated or skipped. The audio will be out of sync.

6.13 GUI Controls in the "Edit" page

	<p>The rectangle with red border indicates in which track new clips will be inserted.</p> <p><> Symbol: This is the "Auto Track Selector". It's important in "Trim Edit Mode" when you want to specify which tracks are affected by trimming.</p>
	<p>Linked Selection: If it's activated, selecting one clip will automatically select also the other linked clip. If it's deactivated, you can move the clips individually. If they are already both selected, first deselect them by clicking in an empty area, then select one of them. To link two or more clips, select the clips, right-click in one of them and use "Link Clips".</p>
	<p>Above the preview window there is a "Bypass Color Grades and Fusion Effects" icon, which can be used to disable all color gradings and effects. This can be used for comparing before and after.</p>

6.14 Markers

M	If a clip is selected: Set a marker inside the clip If no clip is selected: Set a marker in the timeline
MM	Enter a name and color for the marker
ALT+draw a marker	Specify a time range
SHIFT+↑	Jump to the next marker
SHIFT+↓	Jump to the previous marker

6.15 Edit the audio volume of a clip

In the "Edit" page the audio volume can be adjusted in the timeline:

- You can change the volume of a whole clip by clicking in the clip with the left mouse button and moving up/down.
- You can add keyframes by holding the ALT key and clicking with the left mouse button in the audio clip. These keyframes can be moved left/right/up/down.
- Of course you can also change the volume in the inspector and set keyframes there.

6.16 Normalize Audio

Audio can be normalized as follows in the "Edit" page:

Use the selection pointer and select one or more audio clips in the timeline. Right-Click one of them and select "Normalize Audio Levels". Set the target level to the desired value. Use "Relative" if all clips shall be normalized to the maximum level of all clips, or "Independent" if each clip is normalized to its own local maximum. Then click on "Normalize".

Rules of thumb for sound levels:

- Normal spoken dialogue: about -12 dB.
- Sound effects: between -10 and -30 dB.
- Music: between -20 and -30 dB.

6.17 The "Fusion" page in DaVinci Resolve

Before you switch to the "Fusion" page, place the playhead over the clip that you want to edit. But you can also select another clip when you open the "Clips" window in the "Fusion" page.

Below the preview window is a frames scale with two yellow markers. These are the "In" and "Out" points from the edit page.

Above the preview window is the "Show Color Channel" icon, where you can select between Color, RTed, Green, Blue and Alpha.



In "Fusion" page press **SHIFT+SPACE** to get a search window for new tools.

A "node" in DaVinci Resolve is the same as a "filter" in FFmpeg.

The colors of the nodes' connections are very important. The positions of the inputs don't care, they will change when you move the nodes.

Foreground inputs: Background inputs: Mask inputs: Outputs:

CTRL+P toggles a node on/off in "Fusion" page.

Export a Fusion composition as a file: File --> Export Fusion Composition

If the composition contains input images, the absolute paths to these images are included in the file.

Import a Fusion composition: Go to the "Fusion" page and then use File --> Import Fusion Composition

Drawback: The imported composition replaces the existing composition. You cannot append the imported composition to the already existing composition.

Workaround for appending a Fusion composition to an existing composition: Draw a rectangle around the existing composition and press **CTRL+C**.

Import the new composition and then paste the old composition with **CTRL+V**.

Save a setting file as a template: Draw a rectangle around all nodes, right-click on one of the nodes and use Settings --> Save As.

The path should be Blackmagic_Design --> DaVinci Resolve --> Support --> Fusion --> Templates --> Fusion

To load a setting file, go to the Effects Library and right-click on "Templates", use "Show Folder", then drag and drop the file to the Fusion window.

To insert a new node in an existing connection between two nodes, drag the new node and drop it when the tip of the mouse pointer is over the connection and the color of the line changes from white to other colors.

To remove a node out of a connection, hold the SHIFT key and drag the node with the mouse. The same technique can be used for inserting it again in the connection.

There are two ways how to connect a node output to two or more inputs of other nodes:

- The simple way is to connect the output to several inputs.
- It's possible to create several "instances" of the node. Copy the node (CTRL+C) and paste an instance (CTRL+SHIFT+V). The instanced node behaves exactly the same as the original node. If properties of the original node are changed, they are also automatically changed in the instanced node. But there is one important exception from this rule: You can deinstance any properties of the instanced node in the inspector. Just right-click on a property and use "Deinstance". Then this property can be changed independently of the corresponding property in the original node. The effect can be reversed by using "Reinstance".

"Sticky Notes" can be used for adding comments to a Fusion composition.

"Wireless Nodes" are useful if you want to connect nodes which are far away from each other, without visible interconnection lines. The input of the "Wireless Node" can be connected to any other node. You can either type the name of the other node, or you can make a right-click on "Input" and then choose "Connect To".

Nodes can be copied with Edit --> Copy and Edit --> Paste. All their properties are also copied.

See also this Youtube video (in german):

Fusion Node besser strukturieren und damit Geld und Zeit Sparen - 7 Tipps <https://www.youtube.com/watch?v=2vltx3BypE>

6.18 Add a new "MediaIn" node

If you need one more "MediaIn" in your Fusion composition, you can just drag and drop the video from the media pool to the composition. A new "MediaIn" node will be generated automatically. However it seems to be impossible to add a "MediaIn" node and then connect this node to an existing video clip.

Note: "MediaIn" is not a clip, but a track (which may contain several clips).

6.19 Fusion Clips

Each Fusion clip has its own timeline!

- Drag and drop a clip in the timeline.
- Right-click on this clip and use "New Fusion Clip".
- Now you have "Fusion Clip 1" in the media pool.
- In the top left corner of the timeline window, click on the symbol and select "Stacked Timelines".
- Drag and drop the "Fusion Clip 1" from the media pool to the "+" symbol in the headline of the timeline window.
- Now you see the timeline of the Fusion clip.
- Optionally you can rename the "Video1" track to "Layer 0 Video"
- Drag and drop another clip to the "Video2" track of the timeline of the Fusion clip.
- Optionally you can rename the "Video2" track to "Layer 1 Video"
- Now click on "Timeline1". Here you see only "Fusion Clip 1" and not the other clip(s).
- Go to the Fusion page.
- Here you see two (or more) Medialn nodes. These correspond to the clips in the timeline of the Fusion clip. In the inspector you can set different values for "layer". "layer 0" corresponds to "Video1" track, "layer 1" corresponds to "Video2" track and so on.
- In the timeline of the Fusion clip it's possible to move or cut the input clips.

Problem:

- If the timeline of the Fusion clip is shown, it's possible to move the clips with respect to each other. However you can't see the output of the Fusion composition simultaneously.
- If "Timeline 1" is shown, you can see the output of the Fusion composition, but you can't move the input clips with respect to each other.
- A possible solution is to insert one or more "TimeSpeed" nodes after the "Medialn" nodes in the Fusion composition. Then you can adjust the "Delay" properties of the "TimeSpeed" nodes to move the clips, and you can see the result in real time.

6.20 The "Color" page in DaVinci Resolve

Primaries - Color Wheels:

Temp	Adjusts the color temperature Note: Acts in wrong direction, positive value means shift towards red.
Tint	Magenta / green adjustment, useful if fluorescent lamps were used
MD	Adjusts the midtone details
Col Boost	Adjusts saturation only for low-saturation regions
Shad	Adjusts the shadow details
HL	Some kind of adjustment for highlights ??? If you can explain it, please let me know.

Lift	Adjust the shadows
Gamma	Adjust the midtones
Gain	Adjust the highlights
Offset	Adjust all of the above

Contrast	Adjusts the contrast
Pivot	Sets the neutral point for contrast adjustments
Sat	Adjusts the saturation
Hue	Adjusts the color rotation in HSB color model.
Lum Mix	??? If you can explain it, please let me know.

Set the black point: Click on the  symbol (above "Lift") and select the black point in the image. The "Lift" value will be modified.

Set the white point: Click on the  symbol (above "Gain") and select the white point in the image. The "Gain" value will be modified.

The  symbol is for "Auto Balance".

For comparing the clip before and after color grading, use the "Bypass Color Grades and Fusion Effects" icon above the preview window.



Alternatively you can press **CTRL+D**.

The pipette icon (above "Lift") is for setting the white balance with a gray card.



If a clip has the wrong color space, right click on it and set "Input Color Space" to the correct value. This is only possible if you die use "DaVinci YRGB Color Managed" in Project_Settings --> Color Management --> Color Space & Transforms --> Color Science.

"Color Managed" means that DaVinci Resolve tries to set the color space automatically, but in some cases it fails and then it's necessary to set it manually.

Gradings can be copied from one clip to another clip with Edit --> Copy and Edit --> Paste or **CTRL+C** and **CTRL+V**.

Alternatively you can select the target clip and then click in the source clip with the middle mouse button.

Alternatively you can copy the color grading from one clip to the next clip: Select the next clip and use Color --> Apply Grade from One Clip Prior

Saving a still: Right-click in the clip or preview window and use "Grab Still". The still will appear as a small image in the top left "Gallery" window. Please note that this is not only a "still image" but instead also all details of the grading are saved.

The gallery has different sections:

- Stills Contains grades that are only available in the current project
- PowerGrade Contains grades that are available from all projects

A still can be exported as an image file by right-clicking on the still (in the gallery) and using "Export". Possible formats are TIFF, JPG, PNG and others.

Apply a color grade from a still to one or more clips: Select one or more clips, then make a right-click on a still (in the gallery) and use "Apply Grade".

Zoom in / out: Mouse wheel

Moving when zoomed in: Middle mouse button

Zoom to fit window: Press **Z** key.

Multiple playheads: <https://www.youtube.com/watch?v=HOrKbHNpujM>

Combining keys:

See "Combining Power Windows with the Mask Control" (Page 2731ff in DR16 manual)

See chapter 126 in the manual, "Combining Keys and Using Mattes" (Page 2852ff in DR16 manual)

Static keyframe: The value remains constant right of the keyframe.

Dynamic keyframe: The value is ramped right of the keyframe.

6.21 Corrector nodes in "Color" page

Each corrector node has two inputs and two outputs:

- Green triangle: Normal input (RGB)
- Blue triangle: Alpha input
- Green square: Normal output (RGB)
- Blue square: Alpha output

Nodes can be labeled by right-clicking and then use "Node Label".

Nodes can be enabled or disabled by clicking on the number in the lower left corner of the node. Alternatively you can select one or more nodes and press **CTRL+D**.

6.22 Secondary Adjustments

Primary adjustments: Work on the entire image.

Secondary adjustments: Work on specific parts of an image.

Power windows:



Hint: If you can't shift the window far enough, it may help to move the entire preview image with the middle mouse button.

Select between inside and outside:



If you have a node for color corrections inside the window, you can also create another node for color corrections outside of the window as follows:

Right-click on the node, Add_Node --> Add_Outside

When you change the first window, the second window changes automatically.

You can apply one FX effect per node. Click on "fx Effects" in top right, then drag and drop an effect to a node. The effect can be removed by right-clicking in the node and using "Remove OFX plugin".

6.23 Save Color Gradings

There are two ways how a color grading can be saved:

- Right-click on a clip (in the clips window) and use "Generate LUT". It's discrete with integer arithmetic. Best method for cameras or monitors.
- Grab a still. These are lossless because they are floating point.

6.24 Wipes

Wipes are an easy way to compare a clip before and after color grading.

- Before you begin color grading, grab a still.
- After you have done color grading, select the reference still and press **CTRL+W** to toggle the wipe mode (or use View --> Show Reference Wipe), or double-click on the still in the gallery.
- You can click in the image and move the wipe with the mouse.
- The wipe can be inverted with **ALT+W** (or use View --> Invert Wipe).

6.25 The "Fairlight" page in DaVinci Resolve

Rules of thumb for sound levels:

- Normal spoken dialogue: about -12 dB.
- Sound effects: between -10 and -30 dB.
- Music: between -20 and -30 dB.

6.26 The "Deliver" page in DaVinci Resolve

All image processing is done with the timeline resolution. It's possible to deliver with a higher resolution, but that means upscaling and in most cases it doesn't make sense. You can change the timeline resolution anytime (but not the framerate!). If you want to deliver at multiple different resolutions, use the highest resolution in the timeline.

Settings for upscaling (superscaling) can be found in Project_Settings --> Image_Scaling --> Output_Scaling

Above the timeline you can set "Render" to either "Entire Timeline" or "In/Out Range", to define which part of the timeline will be delivered. You can set "In" and "Out" points.

For selecting the whole timeline, use the "Pos1" and "End" keys to set the playhead, before setting the "IN" and "Out" points. Or simply set "Render" to "Entire Timeline".

How to deliver only one frame: If you set both the "In" and "Out" points to the same position, only one frame will be delivered (for example as a TIFF image).

Set "Encoder" to "NVIDIA" if a NVIDIA GPU is available.

Quality "High" is sufficient in most cases. "Best" is time consuming and produces large files.

You can save your own preset by clicking on the ●●● symbol and then use "Save New Preset".

Recommended bitrate for FHD videos: about 30000 kBit/s

Recommended bitrate for 4K videos: about 60000 kBit/s

Hint: You can also export clips from other pages (for example from the "Edit" page) by using File --> Quick_Export

6.27 Day-for-Night

See <https://www.youtube.com/watch?v=XKoGpT4CaM0>

6.28 Dynamic Zoom

Select a clip in the "Edit" page and open the inspector. Switch on "Dynamic Zoom". Click on "Dynamic Zoom" to show more properties. There are only two, "Dynamic Zoom Ease" and "Swap".

Click on the  icon in the bottom left corner of the preview window.

Select "Dynamic Zoom". Now you see two rectangles over the preview video. These are the size at the beginning and end of the dynamic zoom. You can change their size and position.

If the zoom is in the wrong direction, click on "Swap" in the inspector. Now the two rectangles are swapped.

The "Dynamic Zoom Ease" option can be used to generate smooth transitions at the beginning and end.

6.29 Synchronize audio with video

Let's assume you have a video with audio recorded by the camera, and additionally you have a second audio track that was recorded by an audio recorder. The two audio tracks aren't synchronised because the camera and audio recorder weren't started simultaneously.

Load both files to the media pool and select them both. It's important that this is done before the clip is drawn to the timeline.

Make a right click on one of the clips and select "Auto Sync Audio". Then you can select between four options. The first two are only available with timecode.

"Based on Waveform" means the audio tracks are synchronized automatically and the original audio track (from the camera) is replaced by the better audio track from the audio recorder. This is the recommended method if you are sure that the audio track from the recorder is the better one.

"Based on Waveform and Append Tracks" means the audio tracks are synchronized automatically and both audio tracks are appended to the video. You can later adjust the volume of both audio tracks individually. This is the recommended method if you aren't yet sure which of the audio tracks is the better one.

Now you can draw the clip to the timeline. The synchronized audio track is shown in the timeline with a ● symbol in front of the filename.

6.30 Cutting on the beat of the music

Play the music and press the "M" key on each beat. Each keypress will set a marker. The playhead will snap to the markers and you can cut the clips at these positions.

6.31 Video stabilization

- Select a clip.
- Open the inspector and double click on "Stabilization".
- Set the parameters and click on "Stabilize".
- If you also select "Camera Lock", the panning of the camera is also compensated.
- If you did change the In or Out points of a clip, then you must stabilize the clip again. If the "Stabilize" button is deactivated, use the workaround to tick "Camera Lock" two times. Then you can stabilize again.
- There is also a "SphericalStabilizer" in Fusion --> Add Tool --> VR --> Spherical Stabilizer

You can also use "Smart Reframe" which is in the inspector at the bottom of "Transform".

6.32 Processing of drone videos

- Use video stabilization.
 - "Mode" = Translation
 - Tick "Zoom"
 - "Cropping Ratio" = 0.9
 - "Smooth" and "Strength" = 1.0
- Use deflicker to suppress the propellers in the top left and top right corners. Effects --> OpenFX --> Deflicker
 - Temporal NR:
 - Use "Advanced Controls".
 - "Frames Either Side" = 5
 - "Mo. Est. Type" = None
 - "Luma Threshold" and "Chroma Threshold" = 100
 - Speed Optimization Options
 - Tick "Limit Analysis Area" and set the box to the upper part, where the propellers are visible.
 - Restore Original Detail After Deflicker
 - "Detail to Restore" = 0

6.33 Track an object

It's important that in the project settings you have set Image Scaling --> Input Scaling --> "Mismatched resolution files" to "Center crop with no resizing" (not sure if this is correct)

Go to "Edit" page and insert two clips to the timeline:

Track "Video1" contains the clip with the moving object, which we want to track.

Track "Video2" is above "Video1" and contains the clip that shall be overlaid over the moving object.

Select "Video2", go to the inspector and move the small video to the position that it shall cover.

Select both clips, make a right-click and use "New Fusion Clip". It's normal that now the upper track has vanished.

Now go to "Fusion" page and add these nodes:



MedialIn1 is the clip which contains the moving object, which we want to track.

MedialIn2 is the clip that shall be overlaid over the moving object.

Select the "PlanarTracker1" and make these settings in the inspector:

- Operation Mode = Track
- Tracker = Point
- Motion Type = Translation

- **Output = Background** This means the output signal of the PlanarTracker is the same as its input signal.

Now you must define the search window for the object.

Use the "Click Append" tool (it's the leftmost icon above the image) to draw a polygon consisting of several points. The last point must close the polygon. A "+" must appear next to the mouse pointer.

Now set the playhead to the first frame and click on "Track to End" to track the object. The PlanarTracker will generate the spline for the moving object.

This spline must somehow be transferred to the PlanarTransform node. There are two ways how to do this:

1. First let the PlanarTracker track the object. Then select the PlanarTracker, go to the inspector and click on "Create Planar Transform". This will create a "PlanarTransform" node that is already filled with the spline data.
2. If you add the "PlanarTransform" node with "Add Tool" function, then the new node will be "empty" (although it looks exactly the same as if generated with method 1). The PlanarTransform node doesn't know the spline data from the PlanarTracker. In this case you must select the new PlanarTransform, go to the inspector and right-click on "Right-click here for Track spline", then use "Connect To" --> "PlanarTransform1" --> "Track".

Can the spline somehow be saved in a file? I haven't yet found out.

Fusion --> Add Tool --> Tracking --> Tracker

How to save a tracker path in a file:

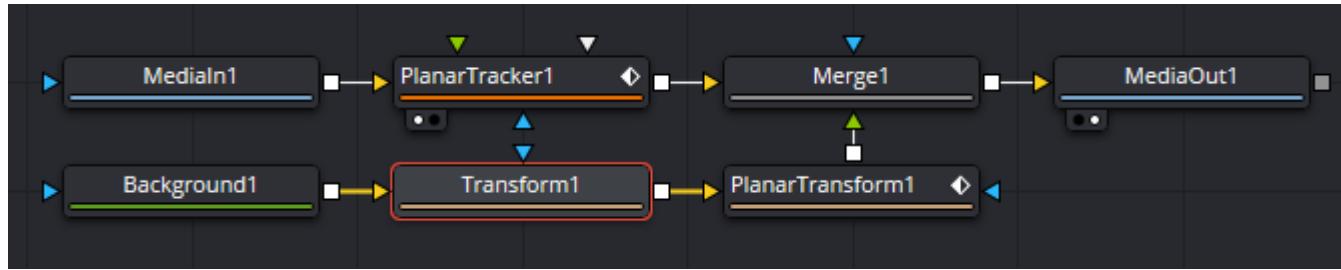
- Select the tracker
- In the inspector, choose "Modifiers".
- Right-click on "Tracker1Tracker1Path" and select "Edit Splines".
- In the "Spline" window, right-click on "Displacement", then "Export" --> "Samples". The *.spl file contains the frame numbers and the displacements, but not the x,y coordinates.
- This method is better: In the "Spline" window, right-click on "Tracker1" --> "Settings" --> "Save As". The *.settings file contains many data:
 - PatternX = X coordinate of pattern center (relative to image width)
 - PatternY = Y coordinate of pattern center (relative to image height)
 - PatternCenter1 = same as above?
 - PatternWidth1 = width of pattern (relative to image width)
 - PatternHeight1 = height of pattern (relative to image height)
 - SearchWidth1 = width of the dashed search box (relative to image width)
 - SearchHeight1 = height of the dashed search box (relative to image height)
 - X = X coordinate with respect to image center (relative to image width)
 - Y = Y coordinate with respect to image center (relative to image height)
 - LX = relative change in X coordinate to the previous frame (relative to image width)
 - LY = relative change in Y coordinate to the previous frame (relative to image height)
 - RX = relative change in X coordinate to the next frame (relative to image width)
 - RY = relative change in Y coordinate to the next frame (relative to image height)

6.34 Track and blur (or pixelize) an object



- Place the playhead over the clip and go to the "Color" page.
- Click on the "Window" icon and place a rectangular or circular window over the object.
- Click on the "Tracker" icon, place the playhead on the first frame of the clip
- Tick "Pan" and "Tilt", and perhaps also "Zoom".
- Track the object by clicking on the ► icon.
- Click on the "Blur" icon and adjust the Radius.
- Instead of applying blur, you can also open the "OpenFX" window, select "ResolveFX Blur --> Mosaic Blur" and drop it in the corrector node.
- "Blur" and "Mosaic Blur" can also be combined.

6.35 Track an object and cover it by a colored rectangle



MediaIn1 is the clip which contains the moving object, which we want to track and cover.

Make these settings in the inspector:

PlanarTracker1 --> Controls

- Operation Mode = Track
- Tracker = Point
- Motion Type = Translation
- Output = Background
- Track Channel = Luma

Draw a closed polygon around the object and then click the "Track to End" symbol.

Merge1 has the default settings.

Background1 --> Color --> Background

- Set the desired color

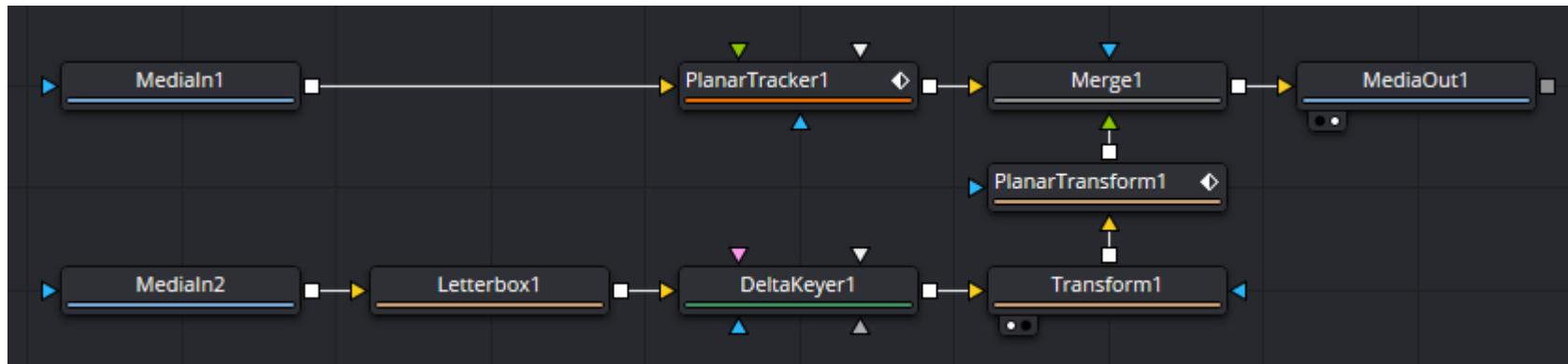
Transform1 --> Controls --> Transform

- Set Center X, Center Y, Size X and SizeY so that the rectangle covers the object.

PlanarTransform1 --> Controls

- Right-click on "Right-click here for Track spline" and select "Connect to" --> "PlanarTracker1" --> "Track"

6.36 Track an object and cover it by an image



MediaIn1 is the background video, and **MediaIn2** is an image.

The purpose of the **Letterbox** node is to convert the dimensions (width x height) of the image to the same values as the video. Go to the inspector and set Mode = "Pan-and-Scan". If the image does already have the same dimensions as the video, then **Letterbox** can be omitted.

If the image is a PNG file and if one color in this file is defined as transparent, then the **DeltaKeyer** node can be omitted.

However the **DeltaKeyer** has very good results for keying and spill removal. Don't try to define the transparent color in IrfanView. The result will be much better when you use the **DeltaKeyer**.

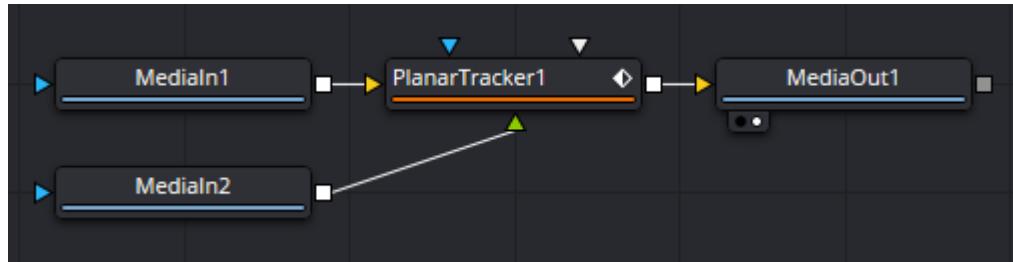
The **Transform** node is used for adjusting the size and position of the overlay picture.

Problem: The image doesn't correctly track the object. It's moving too slow. How can this be explained?

Solution: This problem appears when the image has different dimensions than the video. The tracker spline is saved in relative coordinates, that means the coordinates are relative to the video width and height. If these relative coordinates are applied relative to the smaller image, the motion will be too slow. That's why the image must be converted to the same dimensions as the video, before applying **PlanarTransform**.

Theoretically it should be possible to convert the image automatically to the same size by setting File --> Project Settings --> Image Scaling --> Input Scaling to "Scale entire image to fit". But this didn't work as expected when I tested it. Who can explain it?

6.37 Track an object and corner pin an image



Medialn1 is the background video and Medialn2 is an image.

- Select the clip and go to "Fusion" page.
- Select PlanarTracker and make these settings in the inspector:
 - Set Operation Mode to "Corner Pin" and adjust the four corners to the object.
 - Set Operation Mode to "Track", set the playhead to the beginning and track the object.
 - Set operation mode back to "Corner Pin".

6.38 Isolate Humans with "Magic Mask"

<https://www.youtube.com/watch?v=31uWz7a464Q>

6.39 Overlay text

- Open the Effects Library
- Choose Toolbox --> Titles
- Drag "Text" or "Text+" and drop it in a new video track (above your clips) in the timeline.
- Set the playhead over the text clip. If you forget this step, you wouldn't see the text.
- Select the text clip, open the inspector and edit the properties.
- Fade-in and fade-out can be adjusted directly in the timeline.
- If you want to apply other special effects to the text, you must first convert the text to a clip. Right-click on the text and use "New Compound Clip". This can be reverted by right-clicking and "Decompose in Place".

6.40 Color matching

- Use a clip that contains a color checker, and set the playhead to the correct position.
- Set the lower left window to "Color Match" mode.
- Select the correct color checker, for example "X-Rite ColorChecker Classic".
- Set the "Source Gamma" to the same as in the camera:
 - For example "VLog" (even if the camera was set to VLog-L).
 - For example "ARIB STD-B67 HLG" if the camera was set to HLG.
- Set "Target Gamma" and "Target Color Space", for example to Rec.709
- Set the desired output color temperature, if required. The default is 6500K. **This is not the color temperature of the illumination during the shot, and it's not the color temperature that was set in the camera. It's just an adjustment of the output color temperature.**
- Set the white level. The checkbox is disabled by default. It lets you manually choose the target white point that the automatic correction should use. Raising or lowering this value will stretch or compress the contrast of the final correction.
- Click on View --> Viewer Overlay --> Color Chart to overlay the color chart over the clip. You can toggle the overlay on/off by pressing the ` key two times.
- Drag and drop the corners of the overlay to the correct positions. You can zoom in with the mouse wheel and move the selection with the middle mouse button.
- Click on "Match".

Go to Project Settings --> Color_Management --> Color_Science and choose "DaVinci YRGB Color Managed".

Also tick the box "Use separate color space and gamma".

Now when you right-click on a clip in the Media page, you can set the "Input Color Space" for this clip. The new value becomes applied when you move the playhead.

These are the possible settings for HLG clips:

- Rec.709 HLG ARIB STD-B67
- Rec.2020 HLG ARIB STD-B67 **I think this is the right one for HLG videos from Panasonic GH5S.**
- Rec.2100 HLG (Scene) **This is exactly the same as the previous one.**
(Source: <https://xtremestuff.net/recording-editing-with-hybrid-log-gamma-part-2/> **Highly recommended!**)
- Rec.2100 HLG

You can also right-click on a clip in the Media page and set the "Input Gamma". The new value becomes applied when you move the playhead.

LUT for converting from BT.2020 color space (HLG) to Rec.709: <https://cc-lut.hotglue.me/>

6.41 Generate a LUT

Generate a LUT in the "Color" page:

- Right-click on a clip and select "Generate LUT", then choose either 17, 33 or 65 Point Cube. Save the file in the suggested default path.

6.42 Apply a look-up-table to a clip

There are several methods how to apply a LUT to a clip:

- In the "Color" page, right-click on a clip and select "LUT". The list contains many pre-defined LUTs and at the bottom are your own LUT's.
- Right-click on the clip in the media pool, then select 3D_LUT --> Panasonic --> V-Log to V-709. Warning: It's not shown in the timeline that a LUT was already applied to the clip. You have to know what you have already done.

6.43 Spline Editor



- Step In
- Step Out
- or **CTRL-A** Set loop

6.44 Special effects

Example: "Film Damage" effect

- This can be done in "Edit" page.
- Click on "Effects Library", then choose "OpenFX". Scroll down the list and choose "ResolveFX Texture / Film Damage".
- Drag "Film Damage" and drop it onto the clip in the timeline.
- Select this clip.
- In the inspector you have now a second page called "OpenFX".
- In this page you can edit the properties of the "Film Damage" effect.
- The last property at the bottom of the list is "Global Blend", which can be used for fading the effect in or out.

Removing a special effect that's in its own video track: Just select it and press DEL.

Removing a special effect that's overlaid over a clip: Click the trashcan symbol in the inspector.

Overview of special effects:

Toolbox --> Transitions --> Dissolve --> Cross Dissolve	Normal crossfading
Toolbox --> Titles --> L Lower 3rd	Text in left lower third
Toolbox --> Titles --> M Lower 3rd	Text in middle lower third
Toolbox --> Titles --> R Lower 3rd	Text in right lower third
Toolbox --> Titles --> Scroll	Text scrolling up, alignment is adjustable
Toolbox --> Titles --> Text	Normal text
Toolbox --> Titles --> Text+	Normal text, different parameters
Toolbox --> Titles --> Subtitles --> Subtitle	Subtitle
Generators --> 10 Step	10 Step grayscale

Generators --> 100mV Steps	8 Step grayscale in 100mV steps (0.7V analog video signal)
Generators --> EBU Color Bar	8 Color bars: white, yellow, cyan, green, magenta, red, blue
Generators --> Four Color Gradient	The colors in the corners are selectable, can be used for making a background with a gradient
Generators --> Grey Scale	Grey scale that can also be rotated
Generators --> SMPTE Color Bar	A test image
Generators --> Solid Color	A uniform color image
Generators --> Window	I don't understand this. If you can explain it, please let me know.
Generators --> YCbCr Ramp	???
OpenFX --> ResolveFX Blur --> Box Blur	Makes an image unsharp, can be adjusted independently in X and Y direction
OpenFX --> ResolveFX Blur --> Directional Blur	Similar to box blur, but the blur angle is adjustable
OpenFX --> ResolveFX Blur --> Gaussian Blur	This is superior to box blur
OpenFX --> ResolveFX Blur --> Lens Blur	Simulates lens blur
OpenFX --> ResolveFX Blur --> Mosaic Blur	Makes larger pixels
OpenFX --> ResolveFX Blur --> Radial Blur	This is in fact a tangential blur!
OpenFX --> ResolveFX Blur --> Zoom Blur	This is in fact a radial blur!
OpenFX --> ResolveFX Color --> ACES Transform	Rec.709, Rec.2020, sRGB If you can explain it, please let me know.
OpenFX --> ResolveFX Color --> Chromatic Adaption	Can be used to correct for different illumination types. Illuminant type can also be set to color temperature.
OpenFX --> ResolveFX Color --> Color Compressor	I don't understand this. If you can explain it, please let me know.
OpenFX --> ResolveFX Color --> Color Space Transform	Input Gamma can be set to Panasonic V-Log
OpenFX --> ResolveFX Color --> Color Stabilizer	I don't understand this. If you can explain it, please let me know.
OpenFX --> ResolveFX Color --> Contrast Pop	Seems to be a contrast enhancement filter
OpenFX --> ResolveFX Color --> DCTL	I don't understand this. If you can explain it, please let me know.
OpenFX --> ResolveFX Color --> Dehaze	Dehazing filter
OpenFX --> ResolveFX Color --> Gamut Limiter	I don't understand this. If you can explain it, please let me know.

OpenFX --> ResolveFX Color --> Gamut Mapping	I don't understand this. If you can explain it, please let me know.
OpenFX --> ResolveFX Color --> Invert Color	You can select which color channels are inverted (R, G, B, A)
OpenFX --> ResolveFX Generate --> Color Generator	Same as "Generators --> Solid Color"?
OpenFX --> ResolveFX Generate --> Color Palette	Superimposes a palette with colors from the clip, I don't yet understand for which purpose this might be useful.
OpenFX --> ResolveFX Generate --> Grid	Superimposes a grid
OpenFX --> ResolveFX Light --> Aperture Diffraction	Simulates aperture diffraction
OpenFX --> ResolveFX Light --> Glow	Glow effect, difficult to explain
OpenFX --> ResolveFX Light --> Lens Flare	Simulates a lens flare, consisting of many elements with many adjustable parameters
OpenFX --> ResolveFX Light --> Lens Reflections	Simulates lens reflections, difficult to explain
OpenFX --> ResolveFX Light --> Light Rays	Simulates light rays, difficult to explain
OpenFX --> ResolveFX Refine --> Alpha Matte Shrink and Grow	I don't understand this. If you can explain it, please let me know.
OpenFX --> ResolveFX Refine --> Beauty	Seems to be a filter for making faces beauty.
OpenFX --> ResolveFX Revival --> Automatic Dirt Removal	Automatic dirt removal
OpenFX --> ResolveFX Revival --> Chromatic Aberration	Corrects chromatic aberration
OpenFX --> ResolveFX Revival --> Dead Pixel Fixer	Corrects dead pixels
OpenFX --> ResolveFX Revival --> Deband	Deband filter
OpenFX --> ResolveFX Revival --> Deflicker	Deflicker filter
OpenFX --> ResolveFX Revival --> Patch Replacer	Replaces a region of the clip by another region
OpenFX --> ResolveFX Sharpen --> Sharpen	Sharpening filter
OpenFX --> ResolveFX Sharpen --> Sharpen Edges	Edge sharpening filter
OpenFX --> ResolveFX Sharpen --> Soften & Sharpen	Has adjustments for small, medium and large texture
OpenFX --> ResolveFX Stylize --> Abstraction	Abstraction filter
OpenFX --> ResolveFX Stylize --> Blanking Fill	Makes unsharp stripes at the edges
OpenFX --> ResolveFX Stylize --> Drop Shadow	I don't understand this. If you can explain it, please let me know.
OpenFX --> ResolveFX Stylize --> Edge Detect	Edge detect filter

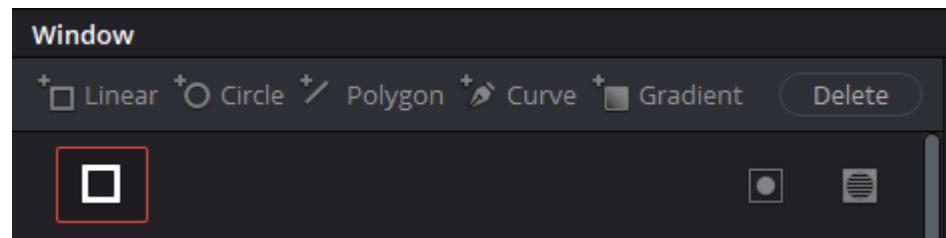
OpenFX --> ResolveFX Stylize --> Emboss	Emboss effect, kind of a high pass filter
OpenFX --> ResolveFX Stylize --> JPEG Damage	Simulates JPEG damages
OpenFX --> ResolveFX Stylize --> Mirrors	Simulates one or more mirrors
OpenFX --> ResolveFX Stylize --> Pencil Sketch	Simulates a pencil sketch
OpenFX --> ResolveFX Stylize --> Prism Blur	Prism blur effect
OpenFX --> ResolveFX Stylize --> Scanlines	Overlays scanlines (sine or square wave)
OpenFX --> ResolveFX Stylize --> Stop Motion	This simulates a smaller framerate, one frame is duplicates multiple times
OpenFX --> ResolveFX Stylize --> Stylize	Difficult to explain
OpenFX --> ResolveFX Stylize --> Tilt-Shift Blur	Makes some regions unsharp
OpenFX --> ResolveFX Stylize --> Vignette	Simulates vignetting
OpenFX --> ResolveFX Stylize --> Watercolor	Watercolor simulation
OpenFX --> ResolveFX Texture --> Analog Damage	Simulates many different errors in analog TV transmission
OpenFX --> ResolveFX Texture --> Film Damage	Simulates many different errors in cine film
OpenFX --> ResolveFX Texture --> Film Grain	Simulates film grain
OpenFX --> ResolveFX Transform --> Camera Shake	Adds camera shake
OpenFX --> ResolveFX Transform --> Flicker Addition	Adds flicker, type can be set to Lift, Gamma, Gain or Vignette
OpenFX --> ResolveFX Warp --> Dent	Different types of warping, can be used for "Black hole" simulation! "Type 2" is good for enlarging eyes, try Size=0.2 and Strength=0.4 "Sine" can also be used with strength= 0.3
OpenFX --> ResolveFX Warp --> Lens Distortion	Simulates lens distortions
OpenFX --> ResolveFX Warp --> Ripple	Simulates concentric waves
OpenFX --> ResolveFX Warp --> Vortex	Simulates a vortex
OpenFX --> ResolveFX Warp --> Warper	Warping by setting points: Shift-click: Define points that don't move, these are red. Click and drag: Define the warp points, these are gray. Important: Enable the "OpenFX Overlay" in lower left of clip window.
OpenFX --> ResolveFX Warp --> Waviness	Simulates linear waves

6.45 Alpha masking

- In "Color" page: Right-click in the nodes window and select "Add Alpha Output".
- Connect the blue alpha output of the last node with the blue alpha output.
- alpha=0 means transparent, alpha=255 means opaque

6.46 Qualifier

- Go to "Color" page and use the pipette tool.
- Mark several points in the image by drawing a line with the left mouse button pressed.
- You can add more points by using the "Pipette +" tool.
- You can remove points by using the "Pipette -" tool.
- You can use the "Highlight" tool in the clip window to see which area is selected.
- You can change the hue, saturation and luminance ranges in the "Qualifier" window.
- Hue, saturation and luminance can also be switched on / off.
- If required, use the "Invert" icon in the "Qualifier" window.
- If it's not possible to select the desired region by hue, saturation and luminance alone, then you can add additional windows to exclude some areas.
- More than one window can be generated with +Linear +Circle +Polygon +Curve +Gradient buttons:



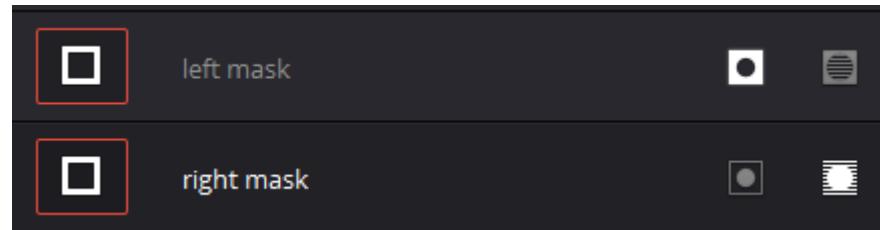
Each mask has two modification buttons (below "Gradient" and "Delete").

- The left symbol inverts the mask.
- The right symbol switches between add (logical or) and subtract (logical and) mode.
- You can give each mask a name by double clicking in the black space in the middle.

6.47 Exclude two regions from a mask

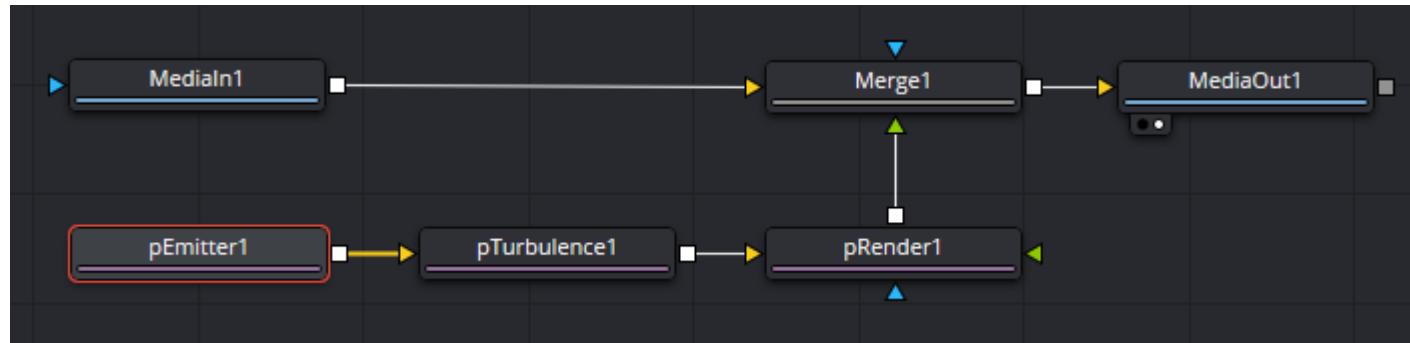
Let's assume you want to convert a clip to black and white, except two small regions that shall keep their colors.

- Select the clip in the timeline and go to the "Color" page.
- Create two rectangular windows by clicking two times on "+Linear". This should also work with other shaped windows.
- It's important that one of the masks is inverted (that's the left symbol) and the other one is switched to subtract mode (that's the right symbol).
- Now change the saturation to zero. This will be applied only to the background, but not to the two masked regions.



6.48 Snowfall

Let it snow!



"MediaIn" is the background clip. Make the following settings in the inspector:

pEmitter --> Controls --> Emitter

- Number = 50
- Number Variance = 50
- Lifespan = 200

pEmitter --> Controls --> Velocity

- Velocity = 0.05
- Velocity Variance = 0.08
- Angle = -90

pEmitter --> Style --> Style

- Style = Blob
- Noise = 0.5

pEmitter --> Style --> Color Controls

- Color = white

pEmitter --> Style --> Size Controls

- Size = 0.25
- Size Variance = 0.25

pEmitter --> Region --> Region

- Region = Line

pEmitter --> Region --> Start

- Start X Offset = -0.5 That means the start point of the line is in the upper left corner.
- Start Y Offset 0.28

pEmitter --> Region --> End

- End X Offset = 0.5 That means the end point of the line is in the upper right corner.
- End Y Offset 0.28

pTurbulence --> Controls

- X Strength = 0.025
- Y Strength = 0.025
- Z Strength = 0

pRender

- Tick "Automatic Pre-Roll" These two parameters are important if you also want to have snow at the bottom in the first frames
- Pre-Generate Frames = 200 You can set this parameter to values greater than 100 by double clicking and using the keyboard.

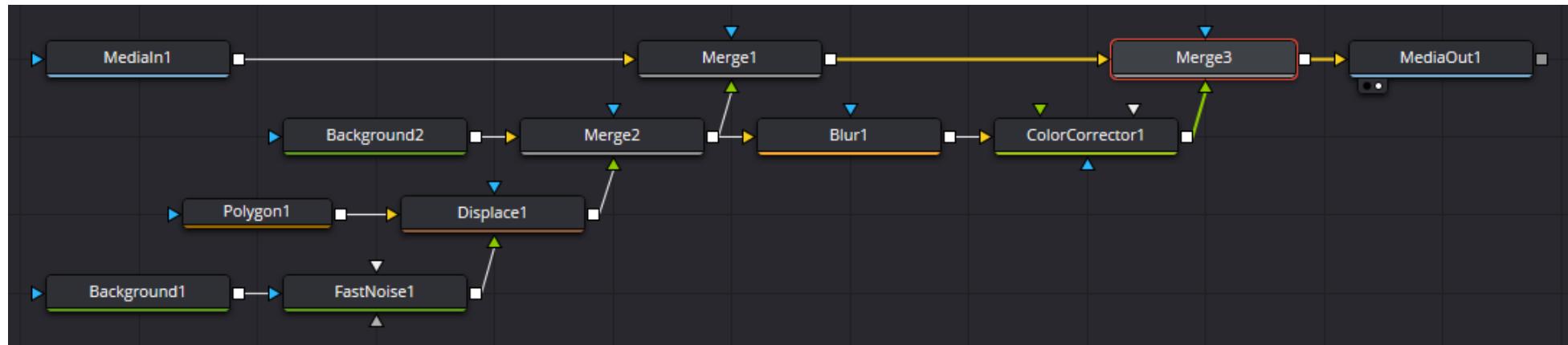
6.49 Polyline toolbar

This is the pure horror :-(

Who can explain it with simple words?

6.50 Artificial lightning

How to overlay realistic dynamic lightning over a clip (the original idea is from Gerald Schulze, simplified by me):



"Medialn" is the background clip. Make the following settings in the inspector:

Background1 --> Color --> Background

- Type = Gradient
- Gradient Type = Linear
- Start X = 0
- Start Y = 0.6 This must equal the Y coordinate of the lowest point of the lightning, where it hits the ground or house.
- End X = 0
- End Y = 1
- Click on the left symbol below the "Gradient" image and set the color to black.
- Click on the right symbol below the "Gradient" image and set the color to white.

Fastnoise1 --> Noise --> Noise

- Detail = 5
- Contrast = 1.2
- Brightness = 0

- Lock XY can be either ticked or unticked, it gives different effects
- Scale = 5
- Angle = 0
- Seethe = 0
- Seethe Rate = 0.5

Fastnoise1 --> Color

- Type = Gradient

Fastnoise1 --> Color --> Gradient

- Gradient Type = Uni
- Color = black

Polygon1 --> Controls

- Tick "Show View Controls"
- Level = 1
- Filter = Fast Gaussian
- Soft Edge = 0
- Border Width = 0.003
- Tick "Solid"
- Position = 0
- Length = 1

Draw a line from the lowest point of the lightning (where it hits the ground or house) to the highest point (slightly above the top border of the frame).

Displace1 --> Controls

- Type = XY
- X Channel = Luma
- X Offset = 0
- X Refraction = 0.07

Background2 --> Color --> Background

- Type = Solid Color
- Color = black with alpha = 0 (fully transparent)

Blur1 --> Controls

- Blur Size = 25

ColorCorrector1 --> Correction

- Channel = Blue

- Gain = 9.0

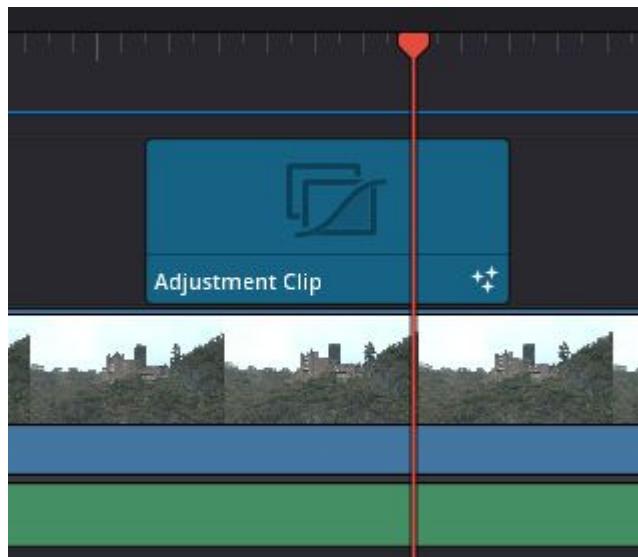
Merge1, Merge2 and Merge3 have the default settings.

6.51 Adjustment clips

An adjustment clip can be created by dragging Effects Library --> Effects --> Effects --> Adjustment Clip to the timeline and dropping it above the video track. Then you can right-click on the adjustment clip and select "Open in Fusion Page", where you can apply any Fusion compositions.

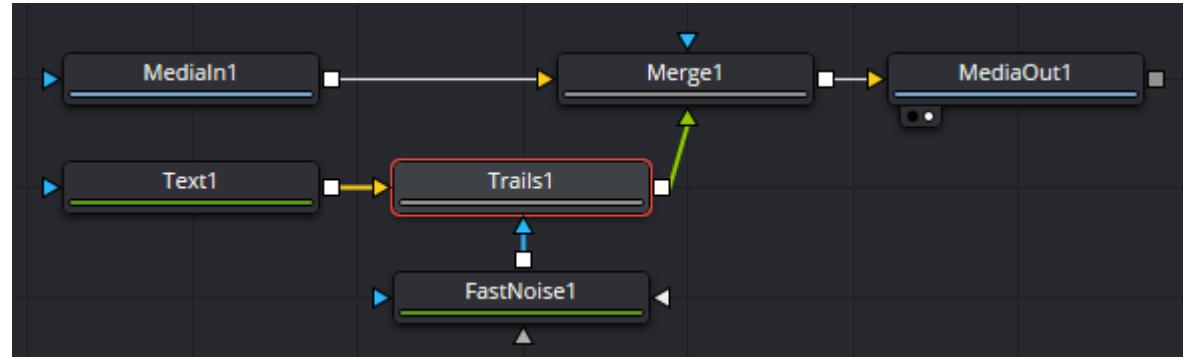
The adjustment clip has many advantages:

- It can be easily moved in the timeline and adjusted in length.
- It's easy to add fade-in and fade-out.
- You can see the effect in real time in the viewer window.



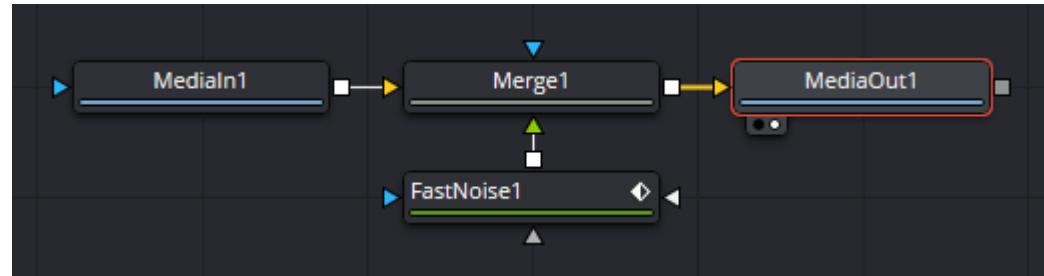
6.52 Trails

Example for trails:



6.53 Fog Effect

How to overlay realistic dynamic fog over a clip:

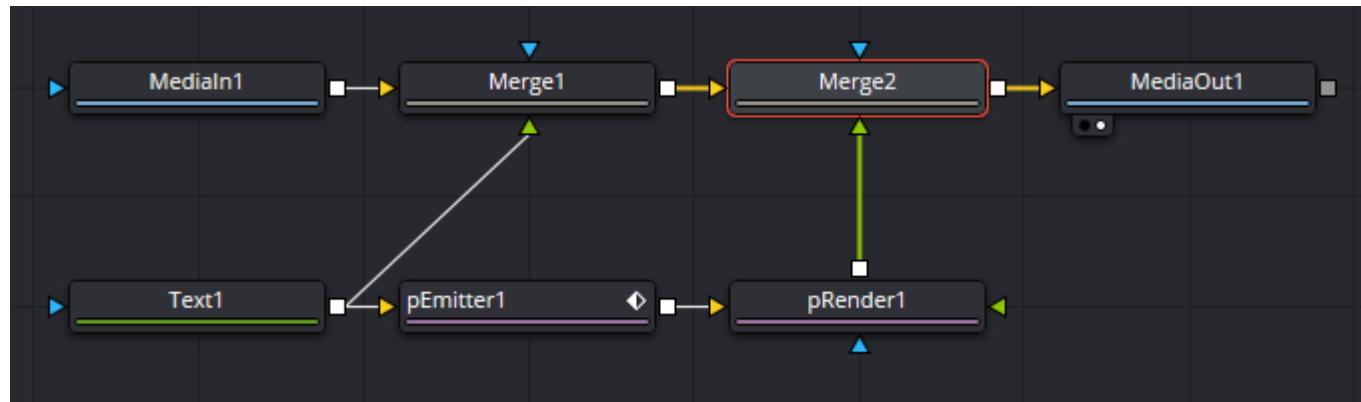


"MediaIn" is the background clip. Make the following settings in the inspector:

pFastNoise --> Noise --> Noise

- **CenterX** Set to any value, if you want dynamic moving fog. Use two or more keyframes.
- **Detail** = 2
- **Contrast** = 1
- **Brightness** = 0
- **Scale** Try out different values
- **Angle** Try out different values
- **Seethe** Try out different values
- **Seethe Rate** Try out different values

6.54 Explosion effect



"MediaIn" is the background clip. Make the following settings in the inspector:

Text --> Text

- Enter the text and set the color and size.
- Global In/Out: Set the range from 0 to the middle of the explosion, so that the text disappears during the explosion

pEmitter --> Controls

- Number = 0 at the beginning,
- Number = 0 just before the explosion,
- Number = 500 at the explosion for about 5 to 10 frames,
- Number = 0 after the explosion
- Velocity = 0.4
- Angle Variance = 360

pEmitter --> Style --> Style

- Style = Blob or nGon

pEmitter --> Style --> Color Controls

- Set the color

pEmitter --> Style --> Size Controls

- Size = 3

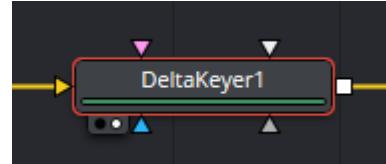
pEmitter --> Region --> Region

- **Region = Bitmap**
- **Region Bitmap = Text1**
- **Channel = Alpha**
- **low = 0.5**
- **high = 1.0**

6.55 Bluescreen / greenscreen

- Drop the background video to the timeline (as track V1).
- Drop the greenscreen video on top of the background video (as track V2).
- Select the greenscreen video and go to the Fusion page.
- Click at an empty spot in the "Nodes" window, type shift-space and then enter "Delta Keyer".
- Alternatively, right-click in the "Nodes" window, use "Add Tool" --> "Matte" --> "Delta Keyer".
- Drag the DeltaKeyer over the line, hold the shift key and release the left mouse button when the line becomes blue and yellow.
- Alternatively, you can connect the lines manually with the mouse.
- In the inspector window you see the background color. Use the pipette tool (right of "Background Color") and pick the background color from the video.
- Then click on the  symbol and fine adjust the two threshold values, and if required also the other parameters.

Note: The two small black / white dots at the bottom left of the "Delta Keyer" specify if the output is shown in the left or right (or both) windows.



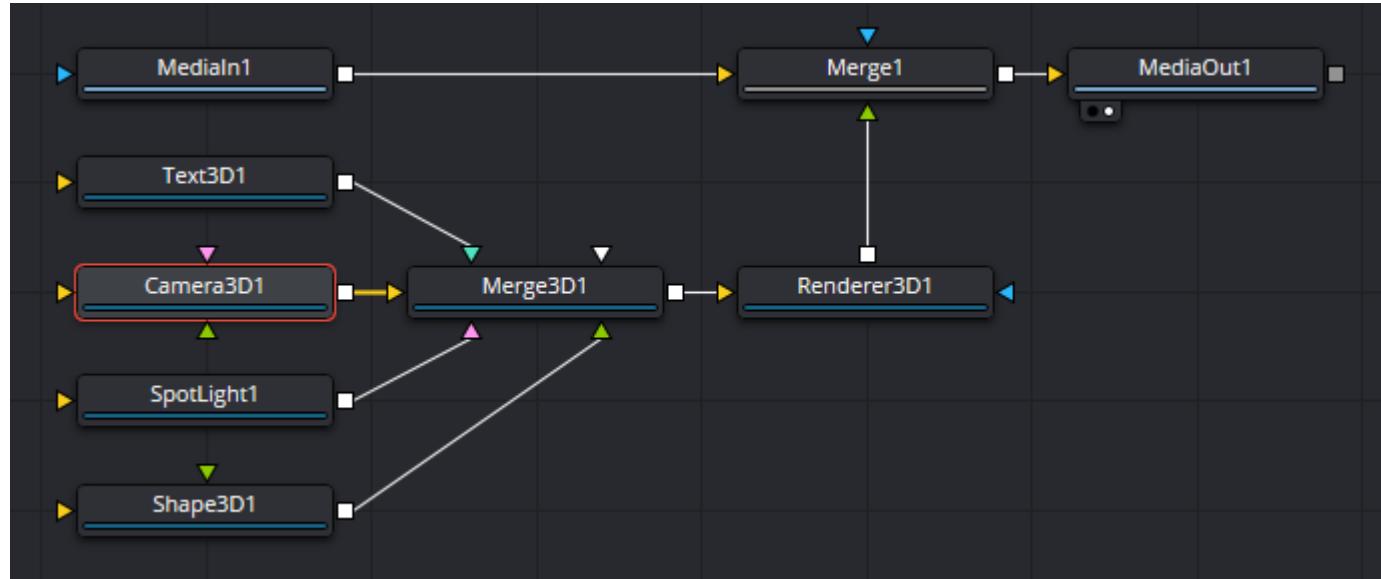
Note: There is another (more complicated) method for creating a greenscreen video, in Color page with a qualifier and alpha channel.

Other method for greenscreen, very easy in "Edit" page:

- Drag and drop the background clip to the timeline.
- Drag and drop the foreground (greenscreen) video above the background clip in the timeline.
- In the bottom left corner of the video window, select "Open FX Overlay".
- Open the Effects Library and drag the "3D Keyer" over the greenscreen clip in the timeline.
- Click on the greenscreen clip and open the inspector.
- Click the pipette "Pick" symbol and draw a line in the video, over those areas that shall become transparent.

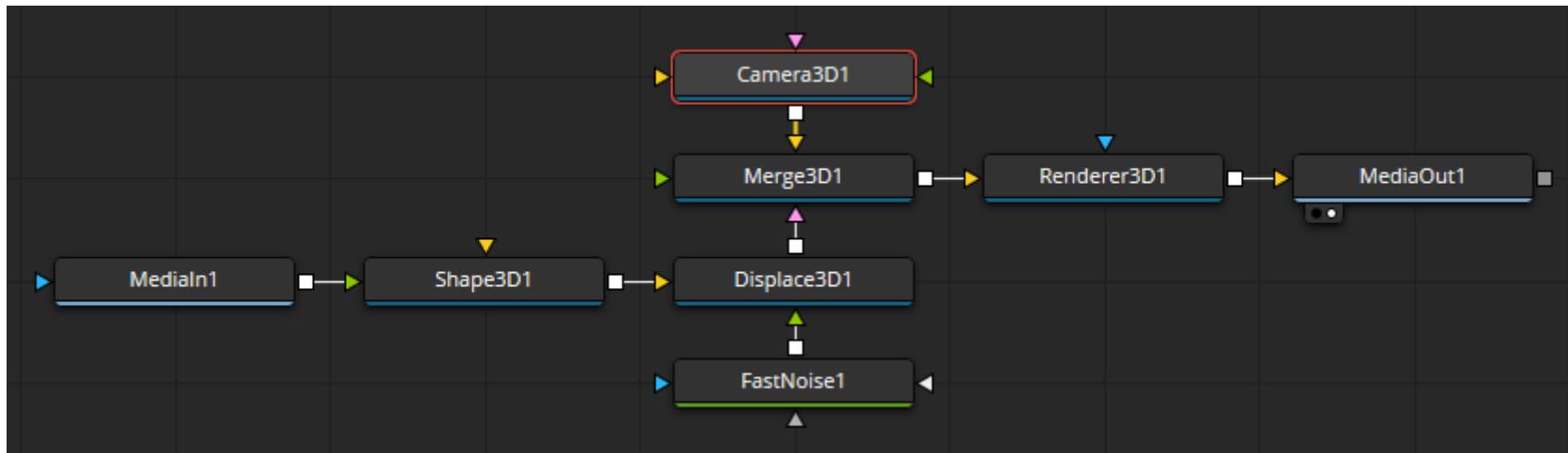
6.56 3D Effects

This is an example for 3D effects:



Many 3D effects can be found in Add Tool --> 3D --> ...

6.57 3D Water Effect



- Shape3D --> Controls --> Shape = Plane

6.58 3D Camera Tracking

See page 1577ff in DaVinci Resolve 17 manual.

See also the video tutorial "3D-Compositing in Fusion" <https://www.blackmagicdesign.com/de/products/davinciresolve/training>

- Step 1: Select the clip and go to the "Fusion" page. Select the first frame.
- Step 2: Define the regions that are used for camera tracking. These must be regions that are "cast to the scene". No persons, no animals, no waves, no trees that move in the wind, no moving clouds. Make sure that you also mask the shadows of moving objects!
 - Select the "Polygon" tool and drag and drop it to the nodes window.
 - Select the first region (rocks) by left-clicking several points around the rocks. Rename the node "POLY_ROCKS".
 - Go to the end of the clip and move the polygon, if required.
 - You can select all points of the polygon by drawing a rectangle around it. Then all points will move together. **This doesn't always work. I don't understand why.**
You can also select the "Select All Points" icon or press **SHIFT+A** to select all points of the polygon, and then move them together.
 - By holding down the **S** key, left-clicking the neutral point and moving the mouse left/right you can scale the polygon or a part of it.
 - By holding down the **T** key, left-clicking the neutral point and moving the mouse left/right you can rotate the polygon or a part of it.
 - By holding down the **X** key, left-clicking the neutral point and moving the mouse left/right you can scale the polygon in horizontal direction.
 - By holding down the **Y** key, left-clicking the neutral point and moving the mouse up/down you can scale the polygon in vertical direction.
 - Repeat the previous steps at multiple keyframes, until the mask is at the right place in all frames.
 - You can switch from one frame to the next or previous frame with the cursor keys. But this works only if the polygon is not selected (it must be either invisible or green). With an english keyboard the [] keys can always be used for switching to the previous or next frame.
Left-clicking in the inspector before pressing the cursor keys does also work.
 - Repeat the previous steps to create other polygon nodes and rename them POLY_BEACH, POLY_WOMAN, POLY_PIRATES. The nodes are all connected in series. For the polygons for the persons, change the "Paint Mode" in the inspector to "Subtract".
 - You can see the masks as black/white images by selecting the small white dot under the last "Polygon" node.
- Step 3: Add a "CameraTracker" node in the main signal path between the input and output nodes.

- Connect the output of the last "Polygon" node to the "Track Mask" input of the "CameraTracker" node.
- Select the "CameraTracker" node, select the "Track" tab (which is the default) and make the following settings in the inspector:
 - Activate "Preview AutoTrack Locations"
 - Select suitable values for "Detection Threshold" and "Minimum Feature Separation". Green points will appear in the preview window. To make these points better visible, you can click on the ●●● icon above the preview window, select "Gain / Gamma" and make the image darker. This is only an adjustment for the preview window.
 - Activate "Bidirectional Tracking".
 - "Gutter Size" defines how close the points can be to the borders.
 - Set "Track Channel" to "Luma".
- Click on "Auto Track".
- Select the "Camera" tab in the inspector and make the following settings:
 - Set the focal length, for example 15.36mm if a 24mm lens is used with the 0.64x SpeedBooster
 - Set the "Film Gate", that's a synonym for the chip size of the camera. You can choose one from the list, in this case the "Aperture width" and "Aperture height" values are set automatically. These are the chip size in Inches. If your camera isn't listed, choose "User".
 - In the latter case you must enter the values for "Aperture width" and "Aperture height" manually, for example for the Panasonic GH5S in FHD or 4K mode: 18.8mm x 10.6mm (0.740" x 0.417").
 - If you don't know the chip size and/or the focal length, you can calculate a valid combination from the measured field of view:

$$x / f = 2 * \tan(\text{fov} / 2)$$
 with x = image width or height in mm, f = focal length in mm, fov = Field of view
 - For example a drone camera has 2688x1512 pixels (16:9) and it's known that the horizontal field of view is 120°. If we assume that the chip size is 8mm x 4.5mm, the focal length would be 2.31mm. It's not important if the chip size and focal length are correct. Only the ratio x / f matters.
 - If you don't know the date, you could also have a look at the metadata of the clip. Click on the "SubView" icon above the preview window and select "Metadata".
- Select the "Solve" tab in the inspector and make the following settings:
 - Click on "Solve". Solving may take some time.
 - Check the average solve error, it should be smaller than 1 for FHD, or smaller than 2 for 4K footage.
 - You can improve the average solve error by removing some problematic tracks, especially those that are red, orange or yellow. Just select a track or a group of tracks and use "Delete" under "Operations On Selected Tracks"

- You can also use the track filtering in the inspector for removing tracks. Then click on "Select TRacks Satisfying Filters" and click on "Delete".
 - Click on "Solve" again. The solve error should now be smaller than before.
- Select the "Export" tab in the inspector and make the following settings:
 - Under "3D Scene Transform" change from "Aligned" to "Unaligned".
 - Select a group of points that are on the ground plane. "Selection is" should be set to "XZ plane (ground)". Now under "Orientation" click on "Set from Selection".
 - Select a group of points that will become the origin of the coordinate system. Now under "Origin" click on "Set from Selection".
 - Click the "Export" button. A group of five nodes will appear in the nodes window. You can move them to an empty space.
- The "Camera3D" node is our virtual camera that matched the movement of the real camera.
-

"Command key" (Mac) is equivalent to "CTRL key" (Windows)

"Option key" (Mac) is equivalent to "ALT key" (Windows)

Navigating in the 3D viewer:

To pan in a 3D Viewer, hold the middle mouse button and drag in the viewer.

To zoom in/out hold down the CTRL key and use the scroll wheel of the mouse.

To rotate around, hold down the ALT key and middle-button-drag in the viewer.

Press Shift-F to Fit all objects in the viewer.

Press F to Fit to selection (or Fit All if nothing is selected).

Press D to Rotate the viewer to look at the center of the currently selected object without moving the viewer's position.

Important: There are three icons "Move", "Rotate" and "Scale" above the preview window. These icons are not for manipulating the 3D view. They are for manipulating individual items in the 3D view, for example for moving or rotating the camera.

Problem with CameraTracker: The "Cancel" button doesn't work while solving.

See also <https://www.youtube.com/watch?v=4I4C8gDOQS4>

6.59 3D Plant Models

3D Plants, overview

Website	Price range	Notes
https://free3d.com/3d-models/pflanzen	\$0 - \$15	only a few models
https://de.3dexport.com/3d-models/plants	\$0 - \$50	many trees, plants and bushes
https://c4ddownload.com/category/free-3d-models/free-3d-architecture/free-3d-plants/	free?	many trees and plants, but the download is suspect
https://www.artstation.com/marketplace/game-dev/resources/3d-models/environment/plants?page=2	cheap	many trees and plants
https://globeplants.com/collections/fbx	\$7 - \$160	many trees and plants
https://www.turbosquid.com/de/3d-model/free/plants/fbx	\$0 - \$150	many trees and plants
https://www.vizplants.com/shop/	unclear	many trees and plants
https://docs.unrealengine.com/5.0/en-US/procedural-foliage-tool-in-unreal-engine/		

You can use Blender to place many plants with varying size on a surface:

<https://www.youtube.com/watch?v=TWZgMg56YMQ>

6.60 Resolve Live

This works only with special hardware, either Blackmagic Design DeckLink or UltraStudio video interfaces can be used to connect a camera.

6.61 Time remapping

Make a right click on a clip in the timeline, then select "Retime Controls" and "Retime Curve".

Set the playhead to the position where you want to begin the time remapping, then set a keyframe by clicking the ♦ icon in the top right corner of the "Retime Frame" window. Repeat this step at the end of the remapping interval, and at one or more positions in between.

Now you remap the timing by moving the points in the curve.

You can also move the white vertical markers in the "Speed Change" window. These have different behaviour if you pick them at the top or at the bottom.

If you click on the small ▼ icons in the "Speed Change" window, you get a context menu with many functions. One of them is "Reset Clip" for returning to normal speed.

Select a clip and open the inspector. Double click on "Retime And Scaling". One of the options for "Retime Process" is "Optical Flow", this means motion interpolation. Even better motion interpolation is possible if you set "Motion Estimation" to "Enhanced Better" or "Speed Warp".

6.62 Slow motion and timelapse

To change the speed of a clip, make a right click on the clip and select "Change Clip Speed".

Question: What does "Ripple Sequence" mean in this context?

6.63 Reducing the framerate (Super 8 simulation)

The framerate can be reduced by using two "TimeSpeed" nodes one after the other. In the first node "Speed" is set to a value greater than 1, and in the second node it's set to the reciprocal value. In both nodes "Interpolation Mode" must be set to "Nearest", so that no frames are interpolated.

Super-8: 18, 24, 25 fps

Normal-8: 16, 18, 24, 25 fps

16mm: 18, 24 fps

35mm: 24 fps

6.64 Noise reduction

It seems that DaVinci Resolve's noise reduction doesn't work for 4K 10-bit videos from the Panasonic GH5S.

These videos must be preprocessed as follows:

```
ffmpeg -i P1000860.MOV -map_metadata 0 -pix_fmt yuv422p10le -c:v dnxhd -profile:v 4 -c:a pcm_s24le -color_range pc  
-movflags write_colr -y out.mov  
  
pause
```

The output video can then be denoised in DaVinci Resolve with ResolveFX --> ResolveFX Revival --> Noise Reduction.

Recommended settings for videos of stars:

- Temporal NR
 - Frames Either side = 5
- Temporal Threshold
 - Luma = 100
 - Chroma = 100
 - Motion = 100
 - Blend = 0
- Spatial Threshold
 - Luma = 0
 - Chroma = 0

The same thing can also be done in the "Color" page. Click on the "Motion Effects" icon.

Temporal averaging (like "tmix" filter in FFmpeg) is possible with ResolveFX --> ResolveFX Temporal --> Smear.

6.65 Reverse a video

Make a right-click on the clip, then select "Change Clip Speed", then tick "reverse speed".

6.66 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" page, 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" page).

Question: When a voice-over is recorded, the speaker must be muted to avoid feedback. When you want to play the record, the speaker must be on again. In most cases the voice-over must be recorded several times, which means you must manually switch the speaker on and off many times. Isn't it possible to mute the speaker automatically during the voice-over?

Workaround: Use headphones.

Other possible workaround: In Fairlight --> Input_Monitor_Style you can set how you want to monitor inputs while recording. There are five possible options, and "Mute" is the option to avoid acoustic feedback:

Input	You only hear the live signal being input, and not the content of the tracks (i.e. what is currently or what has previously been recorded).
Auto (this is the default)	When one or more tracks are armed for recording, you hear the live input signal, while on playback you hear the contents of each track.
Record	You only hear the live input signal while actively recording, meaning the Record button has been pressed while one or more tracks are armed for recording. You don't hear the input signal while tracks are merely armed.
Repro	While recording, you only hear what's just been recorded, played from the track. In other words, you're not listening to the live input, but you're reviewing what's just been recorded as it's recording.
Mute	You hear nothing while actively recording. The drawback is that you can't see the waveform while it is recorded. But it will appear immediately after you have stopped the recording.

6.67 ADR (Automated Dialog Replacement)

- 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". This setting must be done only one time.
- Connect a good microphone to your computer.
- Go to Fairlight page ( icon) and select "ADR", which means automated dialog replacement.
- 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", but in most cases mono is sufficient.
- Give the new track a name, for example "Dialogs".
- 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 the way down), or simply activate the "Mute" icon for the "Main1" output (this is the "M" icon).
- Speak a dialog and set the level so that the level meter doesn't go into the red range.
- Set the In and Out points in the timeline. This defines the maximum length of the dialog.
- In the ADR window, select "List" and click on "New Cue". You see the length of the dialog in the list.
- Mute the other audio tracks, because you don't want to hear them when you record the dialog. Alternatively you can set the "Dialog" track to "Solo" by activating the "S" icon.
- In the ADR window, select "Record" and start recording by clicking on the ● icon.

6.68 VR360

- Create a new project.
- Open the project settings and set the resolution to "custom", using a 2:1 aspect ratio, for example 3840x1920.
- Also set the correct framerate.
- Import the equirectangular video.
- Go to "Fusion" page and make a right-click on the clip. Select "360° View" and "Auto".
- Move around in the 360° video by holding down the shift key and the right mouse button, or by holding down the ALT key and the middle mouse button.
- Zoom in / out by holding down the CRTL key and the middle mouse button. **This doesn't work if "Scale" is set to "Scale to Fit". Right-click on the clip and set "Scale" to any other value.** You can also zoom in with NUM+ and zoom out with NUM-.
- Open the "Effects Library" and select Tools --> VR --> PanoMap
- Drag and drop it as a new node in the signal path
- Open the inspector and set both "From" and "To" to "LatLong".
- Double click on "Rotation" and set the rotation angles for several keyframes. You can also set the rotation order.

This doesn't yet work:

- Select Tools --> VR --> LatLong Patcher and insert it into the signal path after PanoMap.
- In the inspector set "Mode" to "Apply".
- Right-click in the nodes window and add a "Paint" tool after the LatLong Patcher.
- Select "Brush Controls"--> Brush Shape, set the size to a large value.
- Select a suitable color.

<https://www.youtube.com/watch?v=xIOhluai5mk>

6.69 Reframing 360° videos with KartaVR

KartsVR is a plugin for DaVinci Resolve written by Andrew Hazelden.

Found here: DaVinci Resolve 17 - reframe ANY 360 video (Insta360 ONE X2, GoPro MAX, Qoocam 8K) FREE in REAL TIME
<https://www.youtube.com/watch?v=CWw2DaXC7OU>

Documentation for KartaVR: <http://www.andrewhazelden.com/projects/kartavr/docs/>

- Download the "Reactor Installer" here (the filename is "Reactor-Installer.lua"):
<https://www.steakunderwater.com/wesuckless/viewtopic.php?f=32&t=3067>
- Drag and drop this file into the DR Fusion Nodes window. A window "Ready to Install" will appear.
- Click "Install and Launch".
- After a few minutes, a window "Fusion Reactor" appears and you can close this window.
- Go to Workspace --> Scripts --> Reactor --> Reactor Preferences
 - Set "Concurrent cURL Transfers" to 1 (if it's not already set to 1), then click OK
- Go to Workspace --> Scripts --> Reactor --> Open Reactor
- At the left side, select "Karta VR". Check either all lines in the right window, or select only "KartaVR Tools | Reframe360 Ultra"
- Wait 10-20 minutes until everything is installed, then restart DaVinci Resolve.
- Drag and drop an equirectangular video to the timeline.
- Drag "Effects Library --> Toolbox --> Effects --> Fusion Effects --> KartaVR/Viewer/kvrReframe360Ultra" and drop it on the clip in the timeline.
- Alternatively you can in the Fusion page use Add Tool --> KartaVR --> kvrReframe360Ultra and insert the node in the signal path.
- In the inspector you can change these properties:
 - Field of view
 - Pitch, Yaw, Roll
 - Rectilinear Projection, Tiny Planet Projection

These macros are also interesting:

- Add Tool --> Macros --> KartaVR --> Conversion --> Equirectangular2Fisheye
- Add Tool --> Macros --> KartaVR --> Conversion --> Fisheye2Equirectangular
- Add Tool --> Macros --> KartaVR --> Conversion --> Equirectangular2Angular
- Add Tool --> Macros --> KartaVR --> Conversion --> Angular2Equirectangular
- Add Tool --> Macros --> KartaVR --> Conversion --> Equirectangular2InverseAngular
- Add Tool --> Macros --> KartaVR --> Conversion --> Rectilinear2Equirectangular
- Add Tool --> Macros --> KartaVR --> Conversion --> Equirectangular2MeshUV (requires a *.fbx file)
- Add Tool --> Macros --> KartaVR --> Conversion --> Equirectangular2DomeMaster180
- Add Tool --> Macros --> KartaVR --> Conversion --> Equirectangular2DomeMaster220

Problem: DaVinci Resolve crashes, if "kvrReframe360Ultra" is used and then Playback --> Proxy Mode is changed from full to half.

Workaround: Remove the "kvrReframe360Ultra" effect from the clip, then change the proxy mode, then apply the effect again to the clip.

Problem: A black circle region appears in the zenith and nadir zones of the panoramic video when reframing footage in the Resolve Edit/Fusion page.

Workaround: The issue is related to the source MP4 video file having a 2:1 aspect ratio, if the editing timeline was set to use a 16:9 aspect ratio or 1:1 aspect ratio for the output. Resolve's default Timeline Settings will typically fit the panoramic footage to the frame size on one axis, and crop the footage on the other axis which creates the black hole artifact.

To fix the black region at the bottom of the frame issue, you should edit the Resolve Timeline settings. In the Timeline Settings window, uncheck the "[x] Use Project Settings" checkbox. Then edit the "Mismatched Resolution" preference so it is set to "Stretch frame to all corners". This will fit the source MP4 video so the width and height are scaled to precisely match the frame size of the rendered video. The "Mismatched resolution files" setting can also be defined in the "Project Settings > Image Scaling > Input Scaling" preferences window, too.

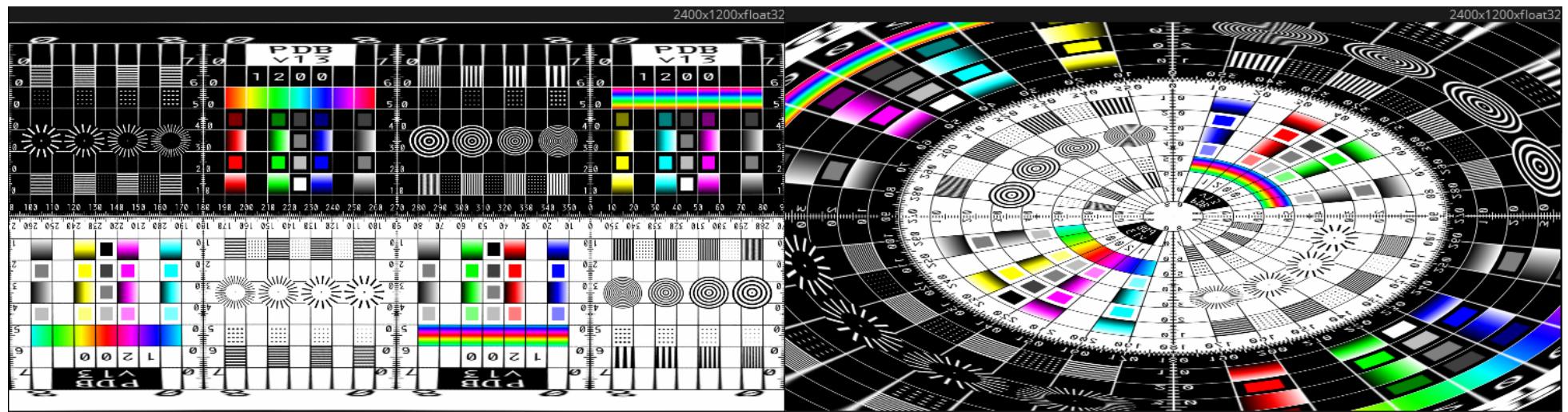
Source: Andrew Hazelden, <https://www.facebook.com/groups/kartavr/permalink/983312762472770/>

6.70 Little Planet

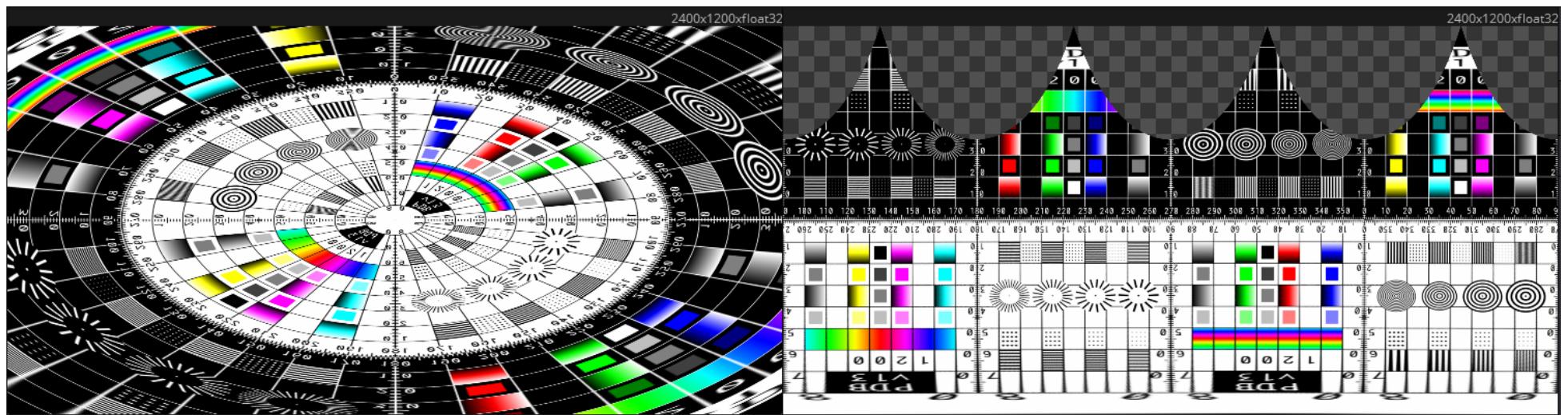
- Create a new project.
- Open the project settings and set the resolution to "custom", using a 2:1 aspect ratio, for example 3840x1920.
- Also set the correct framerate.
- Import the equirectangular video.
- Go to "Fusion" page and add the following nodes:
- "CoordinateSpace", set it in the inspector to "Polar to Rectangular".
- Connect the output to a "Scale" node, and in the inspector untick "Lock X/Y" and set "X Size" to 0.5 or "Y Size" to 2.0

<https://www.youtube.com/watch?v=hYgvn68dklo>

What's the CoordinateSpacs node doing? This is "Polar to Rectangular":

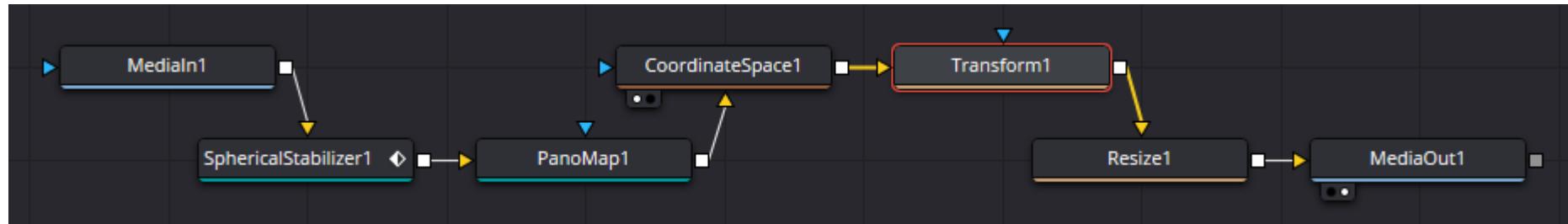


This is Rectangular to Polar:



6.71 Spherical stabilizer

This is an example for SphericalStabilizer and Little Planet projection and making a square output video.



Set the timeline resolution to 960x960

MedialIn is an equirectangular input video with resolution 1920x960.

Make the following settings in the inspector:

SphericalStabilizer --> Controls

- Tick "Reject Dominant Motion Outliers While Tracking"

SphericalStabilizer --> Controls --> Stabilizing Rotation

- Stabilize Strength = 0.5 or 1.0
- Smoothing = 1.0

SphericalStabilizer --> Controls --> Offset Rotation

- Set as many keyframes as required to keep the trees in the video vertical. Use all three rotation angles.

Then set the playhead to the beginning and click on "Track forward".

PanoMap --> Controls --> Rotation

- X = -45 Adjustment of viewing direction (This could have also been done already in SphericalStabilizer)

- **From = Auto**
- **To = LatLong**

CoordinateSpace --> Controls

- **Shape = Polar to Rectangular**

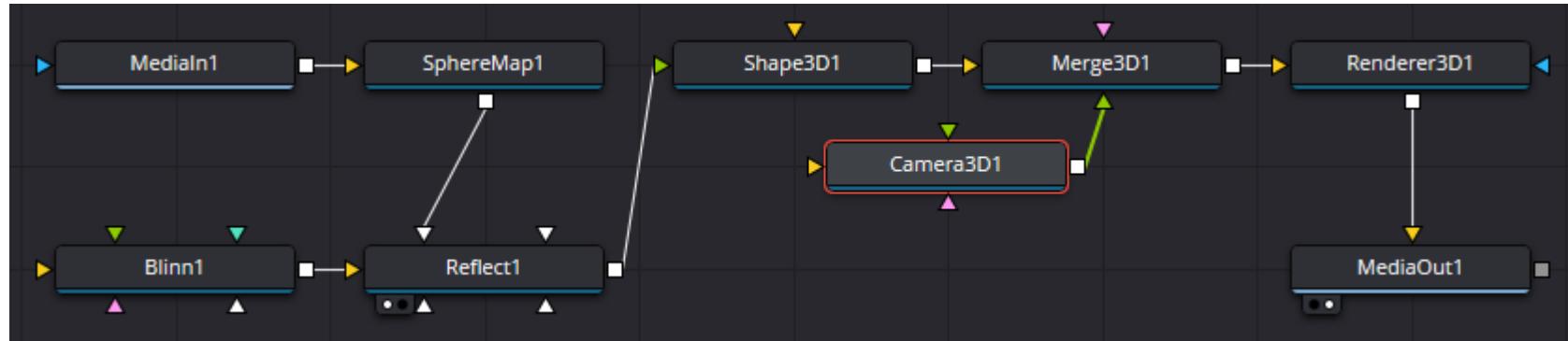
Transform --> Controls --> Transform

- **X Size = 0.5**
- **Y Size = 2.0**
- **Angle = -90**
- **Tick horizontal flip**

Resize --> Controls

- **Width = 960**
- **Height = 960**

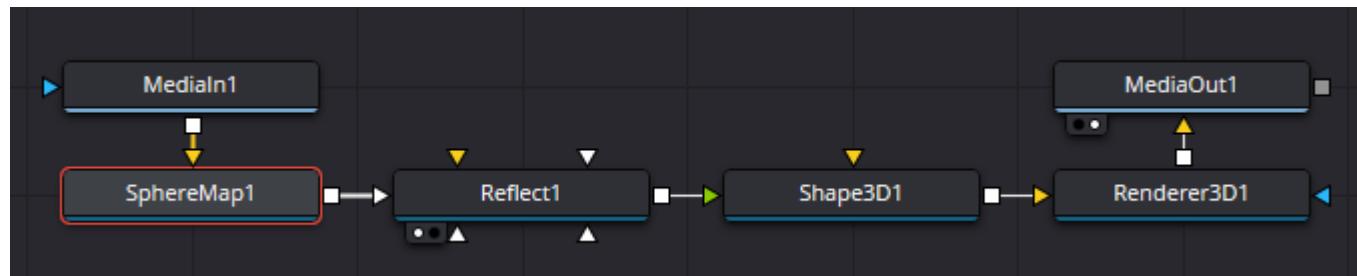
6.72 Mirror sphere



- Right click in the media pool and use "New Fusion Composition". Drag the new Fusion composition to the timeline.
- Add a "Shape3D" node.
 - Shape = Sphere
 - Base Subdivisions = 100
 - Height Subdivisions = 100
- Add a "Merge3D" node. (Can be removed if the camera is omitted.)
- Add a "Renderer3D" node.
- Add a "Camera3D" node. (It's possible to omit the Camera, if the sphere is made smaller.)
 - Adjust the Z translation so that the whole sphere is visible in the right window.
- Drag and drop the equirectangular video to the Fusion window.
- Add a "SphereMap" node.
 - You can adjust the rotation angles.
- Add a "Reflect" node.

- Controls --> Reflection --> Strength Variability = "Constant" or "By Angle"
- Controls --> Reflection --> Glancing Strength = 0.2
- Controls --> Reflection --> Face On Strength = 1.0
- Controls --> Reflection --> Falloff = 4
- Add a "Blinn" node (This allows more settings for the sphere's surface properties. Can be omitted if reflection is 100%).
 - Set the color to black.
- Connect the output of "SphereMap" to the "Reflection.Color.Material" input of "Reflect".

This is the simplified version:

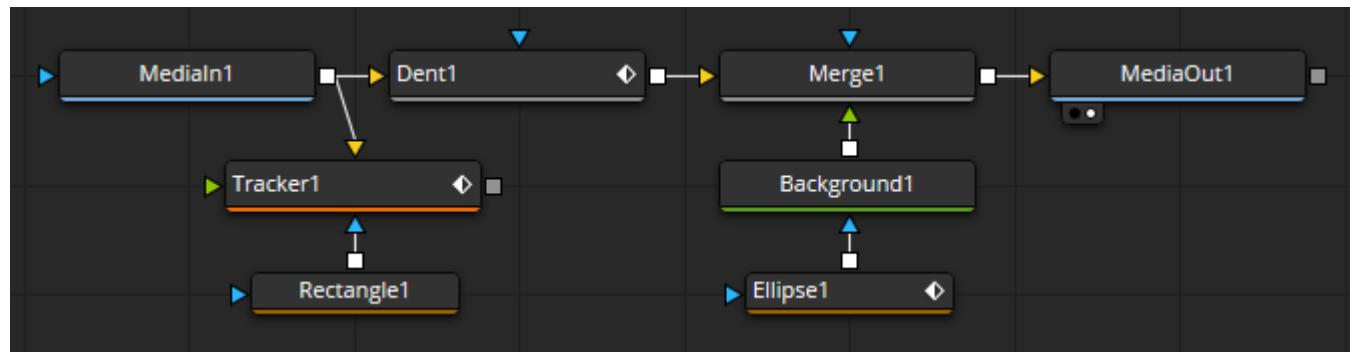


6.73 Black hole effect

YouTube Video:

- Schwarze Löcher erschaffen in Davinci Resolve - Tutorial (german): <https://www.youtube.com/watch?v=Cst9-uSGtdY>

These are the nodes for the black hole effect:

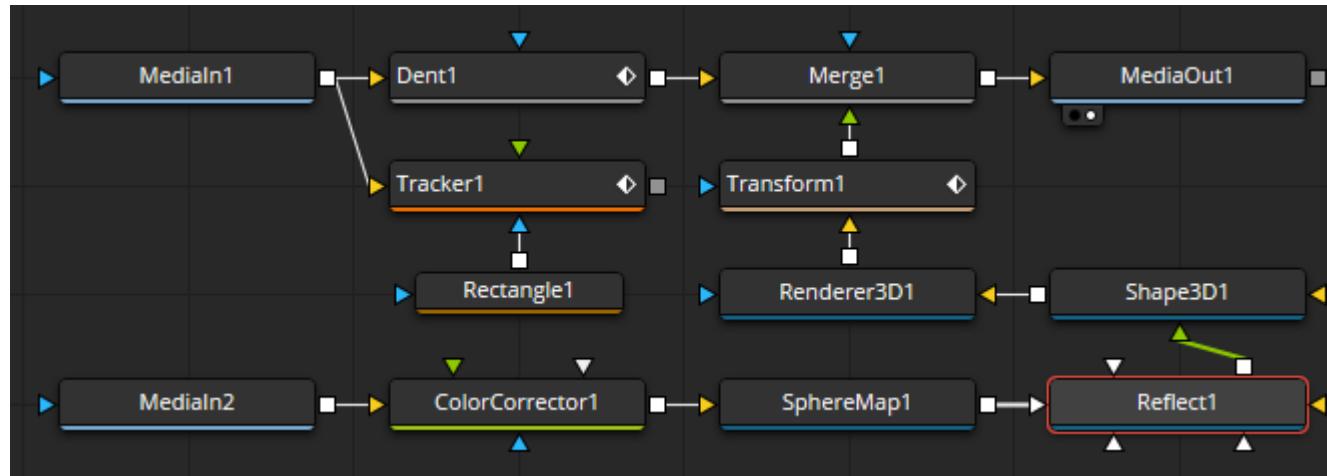


- Track an object in the video, either 2D or 3D tracking (3D tracking is only available in the DR Studio version).
The tracked object can be one of the following:
 - Small sphere on invisible tripod. This is my preferred method.
 - Small sphere located at the correct position, hold by four very thin fishing lines.
 - Any other object which is at the same distance as the black hole, but not too far away from it. See (1) below.
- Go to the Fusion page and add a "Tracker" node. Use "Frames per Point" = 1, "Adaptive Mode" = "none" and "Path Center" = "Pattern Center".
- Make the search window around the object (small sphere on invisible tripod) as small as possible.
- Track the object.
- Add a "Dent" node. **Warning: There are two different "Dent" nodes available:**
 - **"Dent"** is available by using Add_Tool --> Resolve_FX_Warp --> Dent, or by pressing SHIFT-SPACE and then select "Dent".
 - **"Dent (Dnt)"** is available by using Add_Tool --> Warp --> Dent, or by pressing SHIFT-SPACE and then select "Dent (Dnt)".
This is the wrong node because it doesn't have a "Black Hole" mode.
- Set "Dent Type" to "Black Hole" and adjust "Size" and "Strength".

- Right-click on "Position" and select Connect_To --> Tracker1Tracker1Path --> Position
- Add a "Merge" node.
- Add a "Background" node and use the default color black.
- Add an "Ellipse" node. In the explorer, right-click on "Center" and select "Connect_To --> Tracker1Tracker1Path --> Position"
- right-click on "Height", select "Expression" and enter "Width". Now you can adjust the width so that it fits to the central part of the black hole. The height will automatically be set to the same value.
- (1) If the black hole isn't yet at the intended position, click in the inspector on "Modifiers", select "Tracker1Tracker1Path" and change the X,Y positions.
- It's possible to couple the size of the ellipse to the size of the black hole as follows:
Right-click on Ellipse --> Width, select "Expression" and enter "(0.5 * Dent1.size) - 0.01". I did subtract 0.01 because that's the minimum size of the "Dent" node.
It's important that "size" is lower case. If in doubt about the correct syntax, hold the mouse pointer over the source parameter (in inspector for "Dent") and look at the lower left of the screen.
- Now you can animate the size of the "Dent" node. Because the size of the black hole can't be set smaller than 0.01, to let it disappear you must also set "Strength" to 0.

6.74 Wormhole effect

A simplified wormhole effect can be realized by using a black hole effect and overlaying a mirror-sphere effect over the central part of the black hole.



6.75 Correcting Lens Distortion

Lens distortion can also be corrected in DaVinci Resolve, use the "LensDistort" node in Fusion. The various correction models are explained here:
https://www.3dequalizer.com/user_daten/tech_docs/pdf/lstk.pdf

6.76 Programming fuses

https://www.youtube.com/watch?v=_cFUlOKd_RM

6.77 Multicam Editing

"Multicam" means you have filmed the same scene simultaneously with several cameras and/or audio recorders, and now you want to cut between these clips.

- Go to "Media" page and select all clips.
- Make a right-click and select "Create New Multical Clip Using Selected Clips".
- Enter a multicam clip name, set "Angle Sync" to "Sound" and then click "Create".
- Go to the "Edit" page and drag and drop the new multicam clip to the timeline. Only one track is shown, but it contains the tracks from all cameras.
- Below the bottom left corner of the left image window is a symbol where you can select "Multicam".
- Now you see all "Angles" simultaneously in the left window.
- Now you can play the video. When you want to make a cut, just click on the angle that you want to use. The mouse pointer will become a blade.

6.78 Subtracting a darkframe

See <https://vimeo.com/251055924>

6.79 Subtitles

One or more subtitle tracks can be added to a timeline. Open the Effects Library and drag and drop "Subtitle" in the timeline. Use one subtitle track for each language.

Select "Custom" in the "Deliver" page and scroll down to "Subtitle Settings", where you tick "Export Subtitle".

There are several choices for "Format":

- If the file format is set to "MP4", then you can select between
 - "As a separate File" This is a *.srt file (normal ASCII text, readable with each editor)
 - "Burn into Video" This means the subtitles are painted into the frames, so that they can't be switched on/off in the player.
- If the file format is set to "Quicktime", then you can additionally select
 - "As embedded captions" These are subtitles that can be switched on/off in the player. But in DaVinci Resolve 16 this works only with one single subtitle track.

If you want two or more subtitle tracks (which can be switched on/off in the player) in a video, then you can use the following workaround:

Set the format to "As a separate File", set "Export As" to "SRT" and tick both "Subtitle 1" and "Subtitle 2".

After the render process has finished, you have a video (without subtitles) and two *.srt files.

These three files can be combined with FFmpeg to a single *.mov video. It's not possible to do this with normal *.mp4 containers.

This is the FFmpeg batch file (assuming that subtitle track 1 is german and track 2 is english):

```
ffmpeg -i input.mp4 -i "Subtitle 1.srt" -i "Subtitle 2.srt" -c:v copy -c:a copy -c:s mov_text -map 0 -map 1 -map 2 -metadata:s:s:0 language=ger  
-metadata:s:s:1 language=eng -y output.mov
```

pause

Note: The quotation marks "" around the subtitle filenames are only required because the filenames contain a space character.

The output video can be played for example with the VLC player and you can toggle the subtitle tracks with the "V" key.

There was one problem to solve in Davinci Resolve before this worked:

The problem was that the timecode in the subtitle files did begin with 01:00:00:00 (which could mean either "1 hour" or historically "Reel 1").

This can be corrected in Preferences --> User --> Editing --> New Timeline Settings --> Start Timecode, where you can enter 00:00:00:00.

To change the starting timecode of an already existing timeline, right-click on the timeline in the media pool, select Timelines --> Starting Timecode.. and set it to 00:00:00:00.

SMPTE timecode is presented in *hour:minute:second:frame* format. (Source: https://en.wikipedia.org/wiki/SMPTE_timecode)

If the timecode is 01:aa:bb:cc then the time in seconds can be calculated as $(60 * aa) + bb + (cc / fps)$. This is only valid if fps is an integer.

6.80 Supported Codecs

Transcodings can be made under File --> Media_Management

See also: https://documents.blackmagicdesign.com/SupportNotes/DaVinci_Resolve_16_Supported_Codec_List.pdf

What H.264 and H.265 Hardware Decoding is Supported in DaVinci Resolve Studio?

<https://www.pugetsystems.com/labs/articles/What-H-264-and-H-265-Hardware-Decoding-is-Supported-in-DaVinci-Resolve-Studio-2122/>

6.81 Convert *.mkv videos for DaVinci Resolve

Davinci Resolve can't import *.mkv videos, but you can convert them lossless and very fast to *.mp4 with FFmpeg:

```
ffmpeg -i in.mkv -c:v copy -c:a copy -y out.mp4  
pause
```

6.82 Convert 10-bit videos from GH5S for DaVinci Resolve

The free DaVinci Resolve 15, 16 and 17 versions can't import 4K 4:2:2 10 bit videos (from Panasonic GH5S). But this FFmpeg conversion does the job:

```
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 -y out.mov  
  
pause
```

What's the meaning of the options in this command line?

-c:v dnxhd	Video codec: "DNxHD" (Digital Nonlinear Extensible High Definition) You won't find much about this in FFmpeg documentation.
-profile:v 4	Can be in range [0..5], see below output of <code>ffmpeg -h encoder=dnxhd</code>
-c:a pcm_s24le	Audio codec: "PCM signed 24-bit little-endian"
-color_range pc	Set the color range in the stream, 'tv' means limited range and 'pc' means full range.
-movflags write_colr	This is an option of the "mov" muxer, see output of <code>ffmpeg -h muxer=mov</code> Write colr atom even if the color info is unspecified (Experimental, may be renamed or changed, do not use from scripts)

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.....
```

Here are some more details about the different DNxHD / DNxHR profiles:

-profile in FFmpeg	Short name	Long name	Bit depth	Color sampling	Compression rate	Data rate for 4096x2160 @ 25fps
0	DNxHD	?	?	?	?	?
1	DNxHR LB	DNxHR Low Bandwidth	8	4:2:2	22:1	19.04 MB/s
2	DNxHR SQ	DNxHR Standard Quality	8	4:2:2	7:1	61.23 MB/s
3	DNxHR HQ	DNxHR High Quality (8 bit)	8	4:2:2	4.5:1	92.68 MB/s
4	DNxHR HQX	DNxHR High Quality (12 bit)	12	4:2:2	5.5:1	92.68 MB/s
5	DNxHR 444	DNxHR 4:4:4	12	4:4:4	4.5:1	185.25 MB/s

Possible resolutions of DNxHD are HD (1920x1080) and HD720 (1280x720).

Possible resolutions of DNxHR are 4K (4096x2160), UHD (3840x2160), 2K (2048x1080) and HD (1920x1080).

See also: https://en.wikipedia.org/wiki/List_of_Avid_DNxHD_resolutions

See also: <https://web.archive.org/web/20190529091336/https://www.avid.com/static/resources/US/documents/DNxHD.pdf>

See also: https://avid.secure.force.com/pkb/articles/en_US/White_Paper/DNxHR-Codec-Bandwidth-Specifications

You can also apply the VLog-L to Rec709 LUT in the same command line:

```
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 original Vlog-L to Rec709 LUT from Panasonic can be downloaded here:

<https://av.jpn.support.panasonic.com/support/global/cs/dsc/download/lut/index.html>

or here: <https://na.panasonic.com/us/resource-center/v-log-v-709-3d-lut>

Other VLog-L to Rec709 LUTs are available here: <https://nickdriftwood.com/product/driftwood-v-log-v-gamut-sample-lut>

and here: <https://luts.iwltpap.com/free-lut-log-to-rec709-and-rec709-to-log-conversion/>

and here: <https://groundcontrolcolor.com/products/free-sony-v-log-to-rec-709-lut>

This batch file is doing the same thing, but you can drag and video over its 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:

```
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:

```
rem Convert all videos in current folder for DaVinci Resolve

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

:for_body
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
```

6.83 Metadata

It's possible to show properties of clips (metadata) in the "Media", "Edit" and "Fusion" page. Click on "Metadata" in the top right corner. If some informations are missing, try FFprobe instead.

6.84 Sound library

You can download the "Blackmagic Fairlight Sound Library 1.0" here: <https://www.blackmagicdesign.com/support/family/davinci-resolve-and-fusion>

The default installation path is C:\Users\username\Movies\Fairlight Sound Library

In my opinion it's not worth the download.

How to list all sounds in the library:

- Type *** in the search window.

How to remove the whole sound library:

- If you delete the sound files or the whole folder, the sounds will still be listed in DR.
- To remove them from the database, open the Project Manager and click on the icon to the left of Projects that says "Show\Hide Databases". Right click on "Local Database" under "Disk" and select "Open File Location". A folder "Resolve Projects" should appear, click on it and delete the file "SoundLib.db". Now you can rebuild your database in DR using the "Add Library" function.
- The path to the deleted file is C:\ProgramData\Blackmagic Design\DaVinci Resolve\Support\Resolve Disk Database\Resolve Projects

6.85 Blackmagic RAW speed test

This test can be found by clicking on the Windows icon in the bottom left corner, then click on "Blackmagic Design" --> "Blackmagic RAW speed test"

6.86 DaVinci Resolve, Problems and Solutions

DaVinci Resolve is designed for Windows10. There are many problems when you run it on Windows7. Don't do it!

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 Windows7 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."

Solution: Use Windows10 instead of Windows7.

Error message: "The GPU failed to perform image processing because of an error. Error Code: 702"

Solution: Do not use the NVIDIA gaming driver. Use the NVIDIA studio driver instead. It's available on the NVIDIA website.

Error message: Attempting to view depth channel, but none exists in this image.

Solution: Click above the image on the ▾ symbol right of "Z", and select "Color".

Error message: "Media offline" for only a part of a clip.

Solution: Sometimes it helps to close and re-start DaVinci Resolve.

Problem: Something went wrong and the whole timeline is destroyed.

Solution: Do not save the project and close DR. Restart DR, and in the first dialog where you can choose the project, right click on the project and choose "Revert to last saved Project".

6.87 Miscellaneous unsorted things

View --> Safe Area This overlays a mask for showing the safe area that's always visible on a TV screen

Playback → Proxy Mode Off / Half Resolution / Quarter Resolution

***.svg** Standard Vector Graphics

7 Shotcut

Shotcut is a free, open-source video editor.

<https://www.shotcut.org/>

It does have a 360° stabilizer.

The software OpenShot <https://www.openshot.org/> seems to be similar, but Shotcut might be the better one of these two.

8 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:

```
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:

```
@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>

The option -u means "extract unknown tags":

```
exiftool -u %1  
pause
```

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.

Example for extracting GPS metadata from a video:

```
exiftool -p gpx.fmt -ee input.mp4 > output.gpx
```

-p FMTFILE Print output in specified format
-ee Extract information from embedded files

Example for setting a metadata tag:

```
exiftool -ProjectionType="equirectangular" out.jpg
```

Note: The original file is renamed as "out.jpg_original" and the new file is saved as "out.jpg".

9 IrfanView

How to show the mouse coordinates:

Options → Properties/Settings → Viewing → Show_mouse_coordinates_in_the_Statusbar

How to set (or remove) resampling:

View → Display_Options_(window_mode) → Use_Resample_for_fitting_(better_quality)

and

View → Display_Options_(window_mode) → Use_Resample_for_zooming

Enable Color Management: Options --> Properties/Settings --> Tick "Enable color management..."

9.1 Create an image with a transparent color

- Open the image
- Click on Image --> Replace_Color
- Set the transparent color by clicking in the image, and set a tolerance value (for example 20)
- Choose a "New Color", use a color that doesn't exist in the image.
- Click "ok".
- If the result isn't as expected, repeat the previous steps with a different color or tolerance value.
- Click "Save as", set the file format to "PNG".
- Choose a suitable filename.
- In the "PNG/PNM/ICO save options" window tick "Save Transparent Color".
- Click on "Save".
- Select the transparent color in the image.

10 Gimp

How to inspect pixel values:

Use the "Color picker" tool and tick the box "Use info window"

11 Faststone Image Viewer

This is a free image viewer that can automatically the screen when the image file is overwritten by a new image.

<https://www.faststone.org/FSViewerDetail.htm>

12 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".

13 Batch files (DOS, 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>

13.1 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:

Syntax	Result	Example for F:\Meteors_2019\CUT00380.MOV
%1		CUT00380.MOV
%~1	%1 with no enclosing quotation marks	CUT00380.MOV
%~f1	Full path with a drive letter	F:\Meteors_2019\CUT00380.MOV
%~d1	Drive letter	F:
%~p1	Drive-less path with the trailing backslash	\Meteors_2019\
%~n1	For a file, the file name without path and extension For a folder, the folder name	CUT00380
%~x1	File name extension including the period	.MOV

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"

13.2 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
```

13.3 Loop over all files in a directory

```
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.

13.4 Create short text files or append text to a file

```
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.

13.5 Calculate variables in a batch file

It's possible to calculate variables, but only integer arithmetic is supported:

```
set "A=5"
set /a "B=2*%A%"
```

13.6 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.

13.7 Start a new process

This is an example for starting two new processes:

```
start ffplay -noborder -x 640 -y 480 -left 0 -top 200 udp://239.0.0.1:1234  
start ffplay -noborder -x 640 -y 480 -left 640 -top 200 udp://239.0.0.1:1234  
  
pause
```

14 Batch files (Unix, Linux)

Unix batch files (shell scripts) have a different syntax than DOS/Windows batch files. For converting a batch file from DOS to Unix, see this website:

<https://tldp.org/LDP/abs/html/dosbatch.html>

15 VLC Player

<https://www.videolan.org/vlc/>

Documentation: https://wiki.videolan.org/Documentation:User_Guide/ or <https://wiki.videolan.org/Documentation:Documentation/>

This is a subset of VLC's keyboard hotkeys:

Key	Notes
F	Toggle fullscreen
ESC	Leave fullscreen/close dialogue
space	Play/pause
E	Next frame
+	Faster
-	Slower
=	Normal rate
]	Faster (fine)
[Slower (fine)
S	Stop
T	Position/time
Ctrl + Q	Quit
M	Mute
B	Cycle audio track
V	Cycle subtitle track
Shift + S	Make a snapshot picture

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 pipes the video to VLC. Simply drag and drop the video on the batch file's icon. The parameter "%1" causes the file name to be passed to FFmpeg. The option "-b:v 10M" sets the bitrate.

```
ffmpeg -i %1 -vf scale=w=iw/2:h=ih/2 -b:v 10M -f mpegts - | "C:\Program Files\VideoLAN\VLC\vlc.exe" -
```

Note: VLC does also play 360° videos, if they contain the required metadata.

15.1 Show fullscreen video on the extended desktop

Tools --> Preferences --> Video --> Tick "Fullscreen" and select the "Fullscreen Video Device".

16 MPV

MPV is a video player and can be downloaded here: <https://mpv.io/installation/>

You must install two files: mpv.exe and mpv.com

How to show filter curves:

```
mpv av://lavfi:aevalsrc="not(mod(n\,32768)) :d=50" -lavfi-complex  
"[aid1]lowpass,asplit[ao],showfrcs=overlap=0:win_size=32768:win_func=rect:ascale=log,format=rgb0[vo]"
```

MPV supports the same filters as FFmpeg. This is an example for translating a FFmpeg command with a complex filterchain to MPV:

```
rem FFmpeg:  
ffmpeg -i 7Z7A2027.jpg -filter_complex  
"split[1][2];[1]hue=h=0:s=1:b=-2[3];[2][3]hstack" -y out.jpg  
  
rem This is the same thing with MPV:  
mpv 7Z7A2027.jpg --keep-open=yes  
--lavfi-complex="[vid1]split[1][2];[1]hue=h=0:s=1:b=-2[3];[2][3]hstack,scale=iw/2:ih/2[vo]"
```

Notes:

- Don't use "-i" before the input file.
- "--keep-open=yes" means that mpv doesn't close shortly after showing the output image.
- "-filter_complex" must be replaced by "--lavfi-complex="
- The input pad in the filter chain must be called [vid1]. You can't omit it as in FFmpeg.
- The output pad in the filter chain must be called [vo]. You can't omit it as in FFmpeg.

17 CD Rippers

Free CD to MP3 converter: <https://www.eusing.com/CDRipper/CDRipper.htm>

Exact Audio Copy: <https://www.exactaudiocopy.de/>

18 DVD Rippers

Overview: <https://www.videohelp.com/software/sections/decrypters-dvd-rippers>

MakeMKV: <https://www.makemkv.com/>

19 SpyderX Elite, Monitor calibration

The *.icm calibration files are saved in this path: C:\Windows\System32\spool\drivers\color

20 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.

```
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:

```
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:

```
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.

```
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.

```
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:

```
lut3d=C:/test/test.cube
```

Instead use one of these solutions:

```
lut3d=C\\:/test/test.cube    // don't use '' but escape twice
lut3d='C\\:/test/test.cube'  // use '' and escape once
```

Here is a converter for different types of color-look-up-tables:

<https://grossgrade.com/en/downloads-en/>

This software can also fit colors to a ColorChecker:

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.

20.1 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/>

20.2 Beginner tutorials for 3D LUT Creator

1. What is the LUT? 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
3. Working with Lightroom and 3D LUT Creator <https://www.youtube.com/watch?v=o968FH1kV3w>
4. Working with the Image window <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-gAg

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=AiSYkjDdqs>

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-_Ik

15. Batch processing in 3D LUT Creator <https://www.youtube.com/watch?v=1wv1NqXywiY>

20.3 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/XQezpVjCUSl>

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>

8. Blend Modes in 3D LUT Creator https://youtu.be/SKvZg_Zdl9M

9. New way of color toning in 3D LUT Creator <https://youtu.be/xpoU3ZqlGLE>
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/NYzJXdpJDPU>
12. Adjusting the Skin Tone using color match tool <https://youtu.be/rgVFTuu9KIs>
13. Color Masks <https://youtu.be/rQHooXewsN0>

20.4 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>
4. Color matching of 2 cameras in 3D LUT Creator using X-Rite Color Checker https://youtu.be/6er_Pl8XqvI

20.5 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>
4. Working with V-Log footage in 3D LUT Creator and Adobe Premiere <https://youtu.be/EWsJ81UjPBu>

This is the custom ColorChecker file for the cheap chinese ColorChecker, using the RGB values printed on teh 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.

```

NUMBER_OF_SETS      24
LGOROWLENGTH       6
INFO    "sRGB"
INSTRUMENTATION    "3DLUTCreator"
BEGIN_DATA_FORMAT
SampleID   SAMPLE_NAME RGB_R RGB_G RGB_B LAB_L LAB_A LAB_B
END_DATA_FORMAT
BEGIN_DATA
1     A1    0.45  0.32  0.27  38.24  12.86      13.38
2     A2    0.80  0.63  0.55  69.89  14.32      16.84
3     A3    0.40  0.53  0.70  54.78  -2.82     -27.79
4     A4    0.35  0.43  0.24  43.46  -14.44     24.19
5     A5    0.55  0.54  0.76  59.09  11.19     -29.35
6     A6    0.52  0.89  0.82  84.26  -33.27      0.42
7     B1    0.98  0.46  0.14  65.10  48.17      65.24
8     B2    0.31  0.36  0.71  41.41  16.72     -50.62
9     B3    0.87  0.36  0.49  57.07  54.23      8.61
10    B4    0.36  0.25  0.48  32.01  22.42     -29.94
11    B5    0.68  0.91  0.36  85.90  -35.40      60.19
12    B6    1.00  0.64  0.10  75.41  28.21      75.36
13    C1    0.17  0.22  0.56  26.83  18.77     -50.29
14    C2    0.29  0.58  0.32  55.33  -35.21      27.97
15    C3    0.70  0.16  0.20  41.03  55.03      31.12
16    C4    0.98  0.89  0.08  89.78  -4.29      85.94
17    C5    0.75  0.32  0.63  51.41  51.68     -20.96
18    C6    0.02  0.56  0.67  53.96  -24.51     -25.37
19    D1    0.99  0.99  0.99  98.96  0.00      0.00
20    D2    0.90  0.90  0.90  91.29  0.00      0.00
21    D3    0.78  0.78  0.78  80.60  0.00      0.00
22    D4    0.56  0.56  0.56  59.38  -0.14      0.53
23    D5    0.39  0.39  0.39  42.37  0.00      0.00
24    D6    0.20  0.20  0.20  20.79  0.00      0.00
END_DATA

```

Color converter for different color spaces: <https://www.nixsensor.com/free-color-converter/>

21 OBS - Open Broadcaster Software

The software is available here and the installation is quite simple: <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 --> Show Recordings (R). The videos are saved in the folder Username / My videos

21.1 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=5QRx4GRNxSU> 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".

21.2 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 or 10) desktop as follows:

Right mouse click on a free space on the desktop, then New --> Shortcut and then enter "%SYSTEMROOT%\system32\devmgmt.msc".

21.3 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" [Beitrag 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".

21.4 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.

21.5 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-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 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 menues

- 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.

21.6 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/>

See also: <http://johnriselvato.com/ffmpeg-how-to-stream-a-file-to-youtube/>

21.7 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 (that 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 (75Ω) or the white 75Ω 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.

21.8 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.

21.9 Using the same source multiple times

It's possible to use the same video source multiple times. For example you can crop and enlarge multiple regions from the same source and show them in the output.

The trick is to add the source to a group and apply the (crop) filters to the groups instead of the source.

21.10 Studio Mode

In the bottom right corner you can activate the "Studio Mode". Then you see a "Preview" image at the left and a "Program" image at the right.

The following text was copied from here: <https://obsproject.com/wiki/OBS-Studio-Overview>

Activating Studio Mode allows you to change your Scenes in the background without your viewers being able to see you making those changes. After you click on the Studio Mode button, you will see the current Live Scene (what your viewers see) on the right while your edit Scene on the left.

After you are done editing the Scene you can click on "Transition" (or use a Quick Transition/Hotkey if you added one) to swap the left and right, making the Scene you were editing the live Scene. If you are changing Scenes, the last active Scene will be shown in the edit area on the left. After you are done with everything and transitioned to the changed Scene, you can deactivate Studio Mode until you need to edit again. Viewers cannot see when Studio Mode is enabled or not.

21.11 Virtual Camera

In the bottom right corner you can click on "Start Virtual Camera". The virtual camera is not listed in the Windows Device Manager.

Possible applications:

- Capture yourself in front of a green screen and replace the green screen with a background image or video.
- The virtual camera can be used by other programs, for example in BigBlueButton or Zoom.
- It's also possible to use the virtual camera in OBS as a Video Capture Device, select the device "OBS Virtual Camera".

22 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 advertising: 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

22.1 Neutral Density Filters

ND Optical Density	f-stops	Transmission	Filter type	Exposure time factor (Sometimes also called "ND...")
0	0	100%		1x
0.3	1	50%	101	2x
0.45	1.5	35%		2.8x or 3x
0.6	2	25%	102	4x
0.9	3	12.6%	103	8x
1.0	3.33	10%		10x
1.2	4	6.3%	104	16x
1.5	5	3.125%	105	32x
1.8	6	1.6%	106	64x
2.0	6.67	1%		100x
2.1	7	0.78%	107	128x
2.4	8	0.39%	108	256x
2.6	8.67	0.25%		400x
2.7	9	0.2%	109	512x
3.0	10	0.1%	110	1000x or 1024x
3.3	11	0.05%	111	2000x or 2048x
3.6	12	0.024%	112	4000x or 4096x
3.9	13	0.012%	113	8000x or 8192x
4.0	13.33	0.01%		10000x
4.2	14	0.006%	114	16000x or 16384x
4.5	15	0.003%	115	32000x or 32768x

5.0	17	0.001%	117	100000x
6.0	20	0.0001%	120	1000000x
7.0	23	0.00001%	123	10000000x
8.0	27	0.000001%	127	100000000x

Transmission = $100\% / 2^{(f\text{-stop})}$

Note: "ND4" can mean two different things:

- A filter with 13 f-stops which has $0.01\% = 1e-4$ transmission, this is the widely accepted definition
- A filter with 2 f-stops which has $25\% = 1/4$ transmission, I saw this definition for Chinese filters

22.2 Shutter Angle

Shutter angle tutorial: <https://www.red.com/red-101/shutter-angle-tutorial>

22.3 Panning Speed

The rule of thumb is to pan no faster than a full image width every seven seconds, otherwise judder will become too detrimental. This rule is especially simple and powerful because it applies regardless of camera lens, model or sensor size.

The seven second rule of thumb is based on traditional theatrical viewing at 24 fps with a 180° shutter angle. Varying either setting can influence the appearance of panning and change the optimal panning speed.

Source: <https://www.red.com/red-101/camera-panning-speed>

Maximum panning speed with Panasonic GH5S, without and with SpeedBooster (7° rule of thumb):

Lens	Field of view GH5S FHD or 4K (18.8mm x 10.6mm) without SpeedBooster	Maximum panning speed without SpeedBooster	Effective focal length and f/ratio with SpeedBooster 0.64x	Field of view GH5S FHD or 4K (18.8mm x 10.6mm) with SpeedBooster 0.64x	Maximum panning speed with SpeedBooster 0.64x
Olympus M.Zuiko 8-25mm f/4.0	99.2° x 67.0° - 41.2° x 23.9°	14°/s - 6°/s	--	--	--
Leica DG 12-60mm f/2.8-4.0	76.1° x 47.7° - 17.8° x 10.1°	11°/s - 2.5°/s	--	--	--
Sigma 14mm f/1.8	67.8° x 41.5°	10°/s	9.0mm f/1.1	92.7° x 61.2°	13°/s
Canon CN-E 24mm T1.5 L F	42.8° x 24.9°	6°/s	15.4mm T 0.96	62.9° x 38.1°	9°/s
Canon CN-E 50mm T1.3 L F	21.3° x 12.1°	3°/s	32mm T 0.83	32.7° x 18.8°	5°/s
Canon CN-E 85mm T1.3 L F	12.6° x 7.14°	2°/s	54.4mm T 0.83	19.6° x 11.1°	3°/s

23 Screenwriting

Overview article in german language: <https://www.linux-community.de/ausgaben/linuxuser/2021/01/drehbuecher-mit-linux-schreiben/>

Fountain is a plain text markup language for screenwriting: <https://fountain.io/>

Any text editor can be used for writing. The syntax is quite simple: <https://fountain.io/syntax>

Here is the quick reference: https://fountain.io/_downloads/fountain-reference.pdf

Introduction in german language: <https://www.linux-community.de/ausgaben/linuxuser/2021/01/markdown-sprache-fountain-fuer-drehbuecher/>

Screenplain is a browser for Fountain. It's available as an online version: <http://www.screenplain.com/>

And also as a stand-alone version: <https://github.com/vilcans/screenplain/>

Kit Scenarist is a software for writing screenplays: <https://kitscenarist.ru/en/index.html>

Introduction in german language: <https://www.linux-community.de/ausgaben/linuxuser/2021/02/mit-kit-scenarist-drehbuecher-verfassen/>

Breaking down the script, with script breakdown coloring: https://en.wikipedia.org/wiki/Breaking_down_the_script

24 Unix (Ubuntu), compiling FFmpeg

Unix is very different when you're coming from Windows, and unfortunately you need a lot of time to get familiar with it. It's a totally different world.

Warning: Everything in this chapter might be wrong.

Some useful Unix (Ubuntu) commands:

Command	Description
\$ apt-get install ffmpeg	Install FFmpeg (but this might not be the latest version!) Problem: Error messages "Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)" or "Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?" Solution: This happens when you haven't used the computer for a long time, and now many upgrades are automatically running in the background. Just wait a few hours and restart the computer a few times... hopefully the problem will disappear.
\$ git clone http://git.ffmpeg.org/ffmpeg.git	Download the FFmpeg source code (write all in one line). Unfortunately the source code is badly commented (at least those files that I had a look at). ²
\$ cd ffmpeg	Go to the ffmpeg folder.
\$./configure	"configure" is a script file in the ffmpeg folder. You must run it before building FFmpeg. Please note that if you use configure without any options, only a very basic FFmpeg version will be created and many features are disabled. For example PNG as output format isn't possible. Question: What does the configure script do, what's its output file?
\$./configure --list-encoders	This command lists all encoders that can be enabled.
\$./configure --enable-encoder=png	This command does for example enable the png encoder. Well, only theoretically. In the real world you get the error message "Warning: Disabled png_encoder because not all dependencies are satisfied: zlib".
\$ sudo apt-get install zlib1g-dev	This installs the "zlib" library, which is required for the png encoder.
\$./configure --enable-zlib	This command enables all features that are possible with the "zlib" library (not only the png encoder).
\$ make	These two commands are required for building FFmpeg. The "make" command may take 30 minutes or

² Theory: "All nontrivial functions should have a comment above them explaining what the function does, even if it is just one sentence." (Source: <http://www.ffmpeg.org/developer.html#Comments>)

Reality: Comments must be avoided under all circumstances. If you use too many comments, someone else could actually understand the holy code.

\$ sudo make install	so, if you have changed the configuration. For detailed instructions see also https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu Question: In which folder is the executable file ffmpeg.exe? It must be somewhere, but I can't find it.
\$ sudo apt install vlc	Install VLC player (or any other program) What's the difference to apt-get?
\$ chmod +x my_script	Makes a script file executable. Unix script files have no extension. The first line in the script file must contain: #!/bin/bash Problem: Error message: "bash: ./test: /bin/bash^M: bad interpreter: No such file or directory" Solution: Most probably the batch file came from Windows and has a carriage return character in addition to \n at the end of the lines. This can be corrected with the next command.
\$ sed -i -e 's/^M\$//' my_script	This command removes all carriage returns in a script file by substituting \r for an empty string using the linux stream edit command (sed). Press Ctrl+V Ctrl+M to insert ^M.
\$ gsettings set org.gnome.nautilus.preferences executable-text-activation 'launch'	Makes all script files executable by double-clicking in the GUI (write all in one line)
\$./my_script	Execute a script file In case of error message "Permission denied" you may have forgotten "chmod +x my_script"
\$ cd ..	Go back to the parent folder. Please note that (different from Windows) there must be a space character between cd and the two dots.
\$ apt-get install build-essential \$ apt-get build-dep ffmpeg \$ apt-get install yasm	Use these commands to install the necessary software to compile C code (after you got the error message "gcc is unable to create an executable file. If gcc is a cross-complier, ...")
\$ mv test.txt .text.txt	"mv" is the command for renaming. To hide a file or folder, simply add a "." at the beginning of the name
ctrl-alt-T	Open a terminal window

see also <https://www.xmodulo.com/compile-ffmpeg-ubuntu-debian.html>

24.1 GIT

Warning: Don't use this chapter for learning anything about GIT. I never really figured out how GIT works, and everything in this chapter might be wrong.

Sadly, this link does best describe my experience with GIT: <https://xkcd.com/1597/> It's so true. Delete the project and download a fresh copy. I can't remember how many times I have done that.

Install GIT, if it isn't yet installed:

```
$ apt-get install git-core
```

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:

```
$ 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 way a beginner can understand.

A tutorial for GIT (in german language) is here <https://open.hpi.de/courses/git2020> and the required time is estimated as 2-5 hours per week over 4 weeks.

Another tutorial for GIT: <https://cworth.org/hgbook-git/tour/>

Another tutorial (in german): <https://www.dev-insider.de/git-und-die-wichtigsten-git-befehle-kennenlernen-a-720311/>

The following workflow seems to work, but I don't really understand what I'm doing:

```
$ git clone http://git.ffmpeg.org/ffmpeg.git
$ cd ffmpeg

(now edit something in the file libavfilter/vf_v360.c)
```

Now you can use one of these two possibilities:

1. \$ git add libavfilter/vf_v360.c
 \$ git commit
2. \$ git commit -a

The resulting file is written to a hidden folder. Very confusing! You must enable "Show Hidden Files" in the GUI!

```
$ git format-patch HEAD^
```

This command opens an editor where you can insert the commit message. Then it produces a file in the current folder (should be ffmpeg) and this file can be sent to ffmpeg-devel@ffmpeg.org (In an e-mail? Or as an attachment? I don't know.)

The meaning of the ^ character after HEAD is difficult to understand. It is part of the specification of the commit range. The commit range is specified as a half-open range (a, b, c] which means only b and c are in the range. In the above example HEAD^ is the commit before the commit where HEAD is pointing to. Which means the new commit contains all changes that were made after the last commit. (Did I understand this? No.)

Not yet tested:

```
$ git reset
This removes the commits from the history, but the files stay unchanged.
```

```
$ git reset --hard HEAD^
This reverses all changes in all files since the last commit. Only untracked files stay unchanged.
With other words: With this command you get one step back. The command can be used multiple times, if required.
```

```
$ git reset --hard origin/master
This command sets the branch back to the last synchronized state.
```

FFmpeg: Handle (apply and undo) patches from an email: <http://www.das-werkstatt.com/forum/werkstatt/viewtopic.php?f=24&t=2002>

My summary for GIT:

C programming is fun, but GIT is extremely complicated and no fun at all. I don't want to use it.

24.2 Compiling FFmpeg under Windows

This is very complicated, and I decided not to give it a try. There are several projects on github that might be helpful:

<https://github.com/rdp/ffmpeg-windows-build-helpers>

https://github.com/m-ab-s/media-autobuild_suite

25 Cameras and lenses for fulldome video production

	Canon 6D	Panasonic LUMIX GH5S	PanoView XDV360	Kodak SP360_4K
				
Fulldome resolution	180°: 3648 x 3648 (Pictures) 180°: 1080 x 1080 (Video)	180°: 2880 x 2880 (Pictures) 180°: 2496 x 2496 (Video)	220°: 2448 x 2448 180°: 2104 x 2104	235°: 2880 x 2880 180°: 2456 x 2456
Sound recording	stereo 48000 Hz, but both channels are identical, if no external microphone is connected	stereo 48000 Hz, but both channels are identical, if no external microphone is connected	mono 8000 Hz, there is no connector for an external microphone	stereo 48000 Hz, but both channels are almost equal because the microphones are close together; no connector for external microphones
Suitable for fulldome video?	yes, if a fisheye lens is used which has a 180° image diameter less than 20.2mm	yes, if a fisheye lens is used which has a 180° image diameter less than 13.0mm	yes	yes
Suitable for fulldome video at night?	yes	yes, very good	no, too much noise	no, too much noise
Suitable for fulldome timelapse?	yes, arbitrary interval times with external timer	yes, arbitrary interval times with external timer	yes, with internal timer	yes, with internal timer

25.1 Pixel Size in Fulldome Planetariums

Pixel_size = Dome_diameter * PI / (2 * Pixel_number)

Dome diameter	Pixel size for 1080x1080	Pixel size for 1200x1200	Pixel size for 1600x1600	Pixel size for 2048x2048	Pixel size for 3072x3072	Pixel size for 4096x4096	Pixel size for 6144x6144	Pixel size for 8192x8192
1.5 m	2.2 mm	2.0 mm (1)	1.5 mm	1.2 mm	0.8 mm	0.6 mm	0.4 mm	0.3 mm
2 m	2.9 mm	2.6 mm	2.0 mm	1.5 mm	1.0 mm	0.8 mm	0.5 mm	0.4 mm
3 m	4.4 mm	3.9 mm	2.9 mm	2.3 mm	1.5 mm	1.2 mm	0.8 mm	0.6 mm
4 m	5.8 mm	5.2 mm	3.9 mm	3.1 mm	2.0 mm	1.5 mm	1.0 mm	0.8 mm
5 m	7.3 mm	6.5 mm	4.9 mm	3.8 mm	2.6 mm	1.9 mm	1.3 mm	1.0 mm
6 m	8.7 mm	7.9 mm	5.9 mm	4.6 mm	3.1 mm	2.3 mm	1.5 mm	1.2 mm
8 m	11.6 mm	10.5 mm	7.9 mm	6.1 mm	4.1 mm	3.1 mm	2.0 mm	1.5 mm
10 m	14.5 mm	13.1 mm	9.8 mm	7.7 mm	5.1 mm	3.8 mm	2.6 mm	1.9 mm
12 m	17.4 mm	15.7 mm	11.8 mm	9.2 mm	6.1 mm	4.6 mm	3.1 mm	2.3 mm
15 m	21.8 mm	19.6 mm	14.7 mm	11.5 mm	7.7 mm	5.8 mm (2)	3.8 mm	2.9 mm
20 m	29.1 mm	26.2 mm	20.0 mm	15.3 mm	10.0 mm	7.7 mm (3)	5.1 mm	3.8 mm

(1) Planetarium in Sankt Andreasberg observatory

(2) Planetarium in Wolfsburg

(3) Planetarium in Bochum

25.2 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 greater than the read-out height of the chip at the set video resolution.

Camera	Chip Size	Pixels	Video Resolution	Read-out Part of the Chip, Width x Height
Canon 6D	35.8mm x 23.9mm	5472 x 3648	640 x 480 (4:3)	31.87mm x 23.9mm
Canon 6D	35.8mm x 23.9mm	5472 x 3648	1920 x 1080 Full HD (16:9)	35.9mm x 20.19mm
Canon 5D MK4	36mm x 24mm	6720 x 4480	1920 x 1080 Full HD (16:9)	36mm x 20.25mm
Canon 5D MK4	36mm x 24mm	6720 x 4480	4096 x 2160 C4K (17:9)	21.94mm x 11.57mm (Not the whole chip width is used)
Canon 7D	22.3mm x 14.9mm	5184 x 3456	1920 x 1080 Full HD (16:9)	22.30mm x 12.54mm
Canon EOS R	36mm x 24mm	6720 x 4480	1920 x 1080 Full HD (16:9)	36mm x 20.25mm
Canon EOS R	36mm x 24mm	6720 x 4480	3846 x 2160 4K (16:9)	20.57mm x 11.57mm (Not the whole chip width is used)
Sony A7S II	35.6mm x 23.8mm	4240 x 2832	1920 x 1080 Full HD (16:9)	35.6mm x 20.0mm
Sony A7S II	35.6mm x 23.8mm	4240 x 2832	3840 x 2160 4K (16:9)	35.6mm x 20.0mm (The whole chip width is used)
Panasonic LUMIX DC-GH5S	19.2mm x 13.0mm	4096 x 2760	1920 x 1080 Full HD (16:9)	18.8mm x 10.6mm (yet to be confirmed)
Panasonic LUMIX DC-GH5S	19.2mm x 13.0mm	4096 x 2760	3846 x 2160 4K (16:9)	18.8mm x 10.6mm (yet to be confirmed)
Panasonic LUMIX DC-GH5S	19.2mm x 13.0mm	4096 x 2760	4096 x 2160 C4K (17:9)	19.2mm x 10.12mm (The whole chip width is used)
Panasonic LUMIX DC-GH5S	19.2mm x 13.0mm	4096 x 2760	3328 x 2496 Anamorphic (4:3)	17.3mm x 13.0mm (The whole chip height is used)
Nikon D800	35.9mm x 24.0mm	7360 x 4912	1920 x 1080 Full HD	32.0mm x 18.0mm (Not the whole chip width is used)
ZWO ASI178MM	7.4mm x 5.0mm	3096x2080	3096x2080	7.4mm x 5.0mm (The full chip size is used)
Pulnix TM-9701	8.9mm x 6.6mm	768 x 484	768 x 484	8.9mm x 6.6mm (The full chip size is used)

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

25.3 Overview of available fisheye lenses

Lens	Mount	Aperture	Image Angle and Image Circle Diameter	Remarks
Canon EF 8-15mm at 8mm	Canon EF	f/4.0	180° 22.9mm (measured myself)	Very good image quality
Sigma EX DG 8mm	Canon EF ...	f/3.5	180° 22.7mm (measured myself)	Mediocre image quality
Nippon Kogaku 8mm	M42 / Canon EF	f/2.8	180° 23.0mm (measured myself)	M42 mount with adapter to Canon EF
Sigma EX DG 4.5mm	Canon EF ...	f/2.8	180° 12.3mm (measured myself)	Mediocre image quality
Meike 6-11mm at 6mm	Canon EF ...	f/3.5	180° 15.1mm (measured myself)	Good image quality
Meike 6-11mm at 7.5mm	Canon EF ...	f/3.5	180° 18.4mm (measured myself)	Good image quality
Meike 6-11mm at 9.5mm	Canon EF ...	f/3.5	180° 23.7mm (measured myself)	Good image quality
Meike 6-11mm at 11mm	Canon EF ...	f/3.5	180° 28.7mm (measured myself)	Good image quality
Meike 8mm	Canon EF ...	f/3.5	180° approx. 26.9mm 200° approx. 29.9mm	
Opteka 6.5mm	Canon EF	f/3.5	180° approx. 30mm	Bad image quality, true focal length is about 9mm
Entaniya HAL250 6.0mm	Canon EF ...	f/5.6	180° 18.2mm 250° 23.7mm	Only suitable for mirrorless cameras, very expensive
Entaniya HAL250 4.3mm	Canon EF ...	f/4.0	180° 13.1mm 250° 17.0mm	Only suitable for mirrorless cameras, very expensive
Entaniya HAL250 3.6mm	Canon EF ...	f/2.8	180° 11.0mm 250° 14.25mm	Only suitable for mirrorless cameras, very expensive
Entaniya HAL250 3.0mm	Canon EF ...	f/2.8	180° 9.2mm 250° 11.9mm	Only suitable for mirrorless cameras, very expensive
Entaniya HAL200 6.0mm	Canon EF ...	f/4.0	180° 18.2mm 200° 19.9mm	Only suitable for mirrorless cameras, very expensive
Entaniya HAL200 5.0mm	Canon EF ...	f/5.6	180° 15.2mm 200° 16.6mm	Only suitable for mirrorless cameras, very expensive
Samyang 8mm Fisheye II	EF-M, Sony E	f/2.8	180° approx. 29.7mm 188° approx. 31mm	Only suitable for mirrorless cameras, short flange distance
Meike 6.5mm	MFT	f/2.0	180° 15.4mm 190° 15.85mm (measured myself)	Only suitable for mirrorless cameras, short flange distance
Meike 3.5mm	MFT	f/2.8	180° 11.0mm 220° 12.5mm	Only suitable for mirrorless cameras, short flange distance
Olympus M.Zuiko 8mm	MFT	f/1.8	180° approx. 22mm	Lens hood must be removed mechanically
7artisans (Viltrox) 7.5mm	MFT ...	f/2.8	about 27mm (APS-C without vignetting)	Lens hood must be removed mechanically
ZLKC (OCDAY) 7.5mm	MFT ...	f/2.8	about 27mm (APS-C without vignetting)	
Laowa 4mm	MFT	f/2.8	180° 11.6mm 210° 12.9mm	Only suitable for mirrorless cameras, short flange distance

iZugar MKX200-ASPH 3.8mm	MFT	f/2.8	180° 11.7mm 200° 13.0mm	Only suitable for mirrorless cameras, short flange distance
iZugar MKX22 3.25mm	MFT	f/2.5	180° approx. 8.2mm 220° approx. 10mm	Only suitable for mirrorless cameras, short flange distance
Yumiki 2.5mm	CS-Mount	f/1.6	180° approx. 6.1mm 190° approx. 6.4mm	
SMTSEC 2.27mm	CS-Mount	f/1.4	185° 7.2mm	
Fujinon 1.8mm	C-Mount	f/1.4	180° 5.5mm 185° 5.7mm	
Fujinon 2.7mm	C-Mount	f/1.8	180° 8.4mm 185° 8.6mm	

For measurements of fisheye lens nonlinearity, see also Paul Bourke's website: <http://paulbourke.net/dome/fisheyecorrect/>

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).

25.4 Favorable camera / fisheye combinations

Camera	Video resolution	Lens	Aperture	Fully illuminated image circle	Diameter of fully illuminated image circle in pixels
Canon 6D	640 x 480 (4:3)	Canon EF 8-15mm at 8mm	f/4.0	180°	460 Pixel
	1920 x 1080 Full HD (16:9)	Sigma EX DG 4.5mm	f/2.8	180°	656 Pixel
		Meike 6-11mm at 8.2mm	f/3.5	180°	1080 Pixel
		Canon EF 8-15mm at 8mm	f/4.0	159°	952 Pixel
Canon 5D MK4	1920 x 1080 Full HD (16:9)	Sigma EX DG 4.5mm	f/2.8	180°	654 Pixel
		Meike 6-11mm at 8.2mm	f/3.5	180°	1080 Pixel
		Canon EF 8-15mm at 8mm	f/4.0	159°	955 Pixel
	4096 x 2160 C4K (17:9)	Sigma EX DG 4.5mm	f/2.8	170°	2160 Pixel
Canon EOS R	3840 x 2160 4K (16:9)	Entaniya HAL250 3.6mm	f/2.8	180°	2054 Pixel
Sony A7S II	1920 x 1080 Full HD (16:9)	Sigma EX DG 4.5mm	f/2.8	180°	663 Pixel
		Meike 6.5mm	f/2.0	180°	832 Pixel
		Meike 6-11mm at 8.2mm	f/3.5	180°	1080 Pixel
	3840 x 2160 4K (16:9)	Sigma EX DG 4.5mm	f/2.8	180°	1325 Pixel
		Meike 6.5mm	f/2.0	180°	1663 Pixel
		Meike 6-11mm at 8.2mm	f/3.5	180°	2160 Pixel
		Olympus M.Zuiko 8mm	f/1.8	approx. 164°	2160 Pixel
		Sigma EX DG 8mm	f/3.5	180°	2060 Pixel
Sony A7S II with external recorder	3840 x 2160 4K (16:9)	Olympus M.Zuiko 8mm	f/1.8	180°	ca. 1996 Pixel
Panasonic LUMIX GH5S	1920 x 1080 Full HD (16:9)	Sigma EX DG 4.5mm, SpeedBooster 0.71x	f/2.0	180°	888 Pixel
		Sigma EX DG 4.5mm, SpeedBooster 0.64x	f/1.8	180°	800 Pixel
		Meike 6-11mm at 6.8mm, SpeedBooster 0.64x	f/2.2	180°	1080 Pixel
	3328 x 2496 Anamorphic (4:3)	Sigma EX DG 4.5mm	f/2.8	180°	2356 Pixel
		Nippon Kogaku 8mm, SpeedBooster 0.64x	f/1.8	159°	2496 Pixel
		Meike 6.5mm	f/2.0	152°	2496 Pixel

		Meike 6-11mm at 7.5mm, SpeedBooster 0.71x	f/2.5	180°	2496 Pixel
		Meike 6-11mm at 8.2mm, SpeedBooster 0.64x	f/2.2	180°	2496 Pixel
		Olympus M.Zuiko 8mm	f/1.8	106°	2496 Pixel
		Meike 3.5mm	f/2.8	220°	2400 Pixel
3840 x 2160 4K (16:9)		Sigma EX DG 4.5mm, SpeedBooster 0.71x	f/2.0	180°	1775 Pixel
		Sigma EX DG 4.5mm, SpeedBooster 0.64x	f/1.8	180°	1600 Pixel
		Meike 6-11mm at 6.8mm, SpeedBooster 0.64x	f/2.2	180°	2160 Pixel
		Meike 6-11mm at 7.3mm	f/3.5	180°	1080 Pixel
Nikon D800	1920 x 1080 Full HD (16:9)				

25.5 Fisheye projection lenses

Lens	Focal Length	Image Circle	Max. Half-Angle	F/#	Transmittance	Max. Lumens	MTF Center	MTF Edge	Lateral Color R-G	Lateral Color B-G	F-Theta Distortion	Relative Illumination	Approx. Price
Navitar Hemistar HS30	3.0mm	9.072mm 8.2512mm	92° 83.7°	2.5	81%-89%	12	66% @66 lp/mm	60% @66 lp/mm	<4µm	<2µm	-6.0% max.	>95%	\$4800
Navitar Hemistar HS41	4.08mm	12.96mm	97.6°	3.0	>71%	?	65% @66 lp/mm	35% @66 lp/mm	<5µm	<2µm	-5.7% max.	>95%	
Navitar Hemistar HS44	4.4mm	14mm	97.5°	2.3	80%	15	74% @42 lp/mm	25% @42 lp/mm	3.9µm	2.6µm	-5.0% max.	?	
Navitar Hemistar HS45	4.5mm	13.84mm	93°	2.5	80%	15	70% @66 lp/mm	40% @66 lp/mm	<3.4µm	<3.6µm	-5.0% max.	95%	
Navitar Hemistar HS48	4.8mm	14.868mm	93°	2.5	80%	15	72% @66 lp/mm	45% @66 lp/mm	<3.6µm	<4.4µm	-4.0% max.	95%	
Navitar Hemistar HT49	4.87mm	14.606mm	93°	3.0	69%	15	73% @46 lp/mm	40% @46 lp/mm	<5.5µm	<2.5µm	-8.0% max.	89%	
Navitar Hemistar HS68	6.75mm	19.882mm	90°	3.0	>82%	12	55% @46 lp/mm	20% @46 lp/mm	<7µm	<9µm	-5.7% max.	95%	
Navitar Hemistar HMR84	8.35mm	24.2mm	90°	2.5	70%	45	50% @66 lp/mm	50% @66 lp/mm	<3.5µm	<3.0µm	-8.0% max.	93%	
Navitar Hemistar HM4K-96	9.6mm	19.4mm	62.96°	2.4	?	12	65% @125 lp/mm	63% @125 lp/mm	<0.7µm	<1µm	8.1% max.	?	
Navitar Hemistar HMR113	11.27mm	32.24mm	90°	2.5	73%	45	55% @66 lp/mm	55% @66 lp/mm	<3.8µm	<3.8µm	-9.0% max.	95%	
Navitar Hemistar HM117	11.7mm	24.454mm	65.3°	2.5	75%	20	85% @66 lp/mm	59% @66 lp/mm	<3µm	<2µm	-9.0% max.	>88%	
Navitar Hemistar HMT119	11.88mm	26.0mm	68°	2.5	75%	45	90% @66 lp/mm	41% @66 lp/mm	<3.75µm	<3.75µm	-8.5% max.	>97%	
ISCO-Optic 1:4 14.3mm	14.3mm	?	90° ?	4.0	?	?	?	?	?	?	?	?	
Navitar Hemistar HM4K-178	17.8mm	39.25mm	69°	2.6	>88%	?	80% @66 lp/mm	50% @66 lp/mm	<1.5µm	<3.2µm	-8.0% max.	>98%	
ISCO-Optic 1:2 21mm	21.0mm	?	90° ?	2.0	?	?	?	?	?	?	?	?	

Paragon Optics 26mm f/2.8 180° (Used in the planetarium in Sankt Andreasberg observatory)	26mm	66mm	90°	2.8	?	?	?	?	?	?	?	?	?	?	?
---	------	------	-----	-----	---	---	---	---	---	---	---	---	---	---	---

25.6 Non-fisheye projection lenses

Lens	Focal Length	Throw Ratio	Image Diagonal	Max. Half-Angle	F/#
Navitar NuView MCL2125	21mm		1.3" = 33mm	38°	?
Navitar NuView MCZ1218	32.5mm - 45.7mm		1.3" = 33mm	20° - 27°	?
Paragon Optics 63.5mm F/2.5 60°	63.5mm		?	30° ?	2.5
Digital Projection 112-499	?	0.76	?	?	?
Digital Projection 115-339	?	0.75 - 0.93	?	?	?
Digital Projection 112-500	18.7mm - 26.5mm	1.24 - 1.79	?	?	1.9 - 2.5
Digital Projection 112-501	25.7mm - 33.7mm	WUXGA: 1.73 - 2.27 UHD 4K: 1.71 - 2.25	?	?	1.64 - 1.86
Digital Projection 112-502	?	WUXGA: 2.22 - 3.67 UHD 4K: 2.20 - 3.67	?	?	?
Digital Projection 112-503	52.8mm - 79.1mm	3.58 - 5.38	?	?	1.85 - 2.41
Digital Projection 112-504	?	5.31 - 8.26	?	?	?

Projection_Ratio = Distance / Image_Width

25.7 Beamers for fulldome projection

Beamer	ANSI Lumen	Pixel size	Pixels on DMD chip	DMD size
ACER P7605	5000	?	1920x1200	?
Digital Projection E-Vision Laser 11000 4K-UHD	9500	5.4 µm (Texas Instruments DLP670S)	2716x1600 for whole DMD chip 2560x1600 for WUXGA = 16:10 2716x1528 for UHD 4K = 16:9	14.6664mm x 8.6400mm for whole DMD chip 13.8240mm x 8.6400mm for WUXGA = 16:10 14.6664mm x 8.2512mm for UHD 4K = 16:9 8.2512mm --> 167.4° Field of view with Navitar 3mm lens ??? That's wrong.

25.8 Flange distances

MFT (Micro 4/3)	19.25 mm
Canon EF und EF-S	44.0 mm
Canon EF-M	18.0 mm
Canon R	20.0 mm
Canon FD	42.0 mm
M42 = M42x1.0	45.46 mm
T 2 = M42x0.75	55.0 mm
C-Mount = 1"-32	17.526 mm = 0.69"
CS-Mount = 1"-32	12.526 mm = 0.69" - 5.0 mm
Sony E-Mount	18.0 mm
Nikon F	46.5 mm
ZWO ASI178MM	12.5 mm

Passive adapter from M42 lens to Canon EF camera: 45.46mm - 44.0mm = 1.46mm

Passive adapter from Canon EF lens to MFT camera: 44.0mm - 19.25mm = 24.75mm

Passive T2 / M42 Adapter: 55.0mm - 45.46mm = 9.54mm

25.9 Aperture numbers, rounded and exact

0.8	0.9	1.0	1.1	1.2	1.4	1.6	1.8	2.0	2.2	2.5	2.8	3.2	3.5	4.0	4.5	5.0	5.6
0.794	0.891	1.000	1.122	1.260	1.414	1.587	1.782	2.000	2.245	2.520	2.828	3.175	3.564	4.000	4.490	5.040	5.657

Formula for exact numbers: $f_no = 2^{(n/6)}$ with n = -2 to 15

25.10 Test patterns for fulldome projection

Very nice fulldome test patterns on Paul Bourke's website: <http://www.paulbourke.net/dome/testpattern/>

Update January 2022: Paul has added equirectangular test patterns at the bottom of his website.

Make a double-fisheye test image and an equirectangular test image:

```
set "IN=1200.png"          :: Test pattern from http://www.paulbourke.net/dome/testpattern/1200.png
set "OUT=double_fisheye_test.png"  :: Double fisheye test image

ffmpeg -i %IN% -i %IN% -lavfi "[0]transpose=1[left];[1]transpose=2,negate[right];[left][right]hstack" -y %OUT%

set "IN=double_fisheye_test.png"
set "OUT=equirectangular_test.png"  :: Equirectangular test image

ffmpeg -i %IN% -lavfi "v360=input=dfisheye:output=e:ih_fov=180:iv_fov=180:pitch=90" -y %OUT%

pause
```

Make a double equirectangular test image, consisting of two equirectangular test images vertically stacked together. The top half is yellow and the bottom half is magenta. This can be used for wormhole simulations:

```
set "IN=1024.png"          :: Test pattern from http://www.paulbourke.net/dome/testpattern/1024.png
set "OUT=double_equirect.png"  :: Double equirectangular test image

ffmpeg -i %IN% -i %IN% -lavfi "[0]transpose=1[left];[1]transpose=2,negate[right];[left]
[right]hstack,v360=input=dfisheye:output=e:ih_fov=180:iv_fov=180:split[e1][e2];
[e1]colorchannelmixer=.33:.33:.33:0:.33:.33:.33:0:0:0:0:0[yellow];
[e2]colorchannelmixer=.33:.33:.33:0:0:0:0:0.33:.33:.33:0[magenta];[yellow][magenta]vstack" -y %OUT%

pause
```

Make a double equirectangular test image, consisting of two equirectangular test images vertically stacked together. The top half is yellow and the bottom half is magenta. Each image contains a grid and is labelled "SOUTH WEST NORTH EAST UP DOWN". This can be used for wormhole simulations:

```
ffmpeg -lavfi color=c=black@0:s=720x360,format=rgba,^
drawtext=text="DOWN":x=145:y=173:fontsize=24,^
drawtext=text="UP":x=523:y=173:fontsize=24,^
v360=e:e:roll=90:interp=near,^
drawtext=text="NORTH":x=322:y=173:fontsize=24,^
scroll=hpos=0.5,^
drawtext=text="SOUTH":x=322:y=173:fontsize=24,^
drawtext=text="WEST":x=510:y=173:fontsize=24,^
drawtext=text="EAST":x=154:y=173:fontsize=24[text];^
color=c=yellow:s=720x360,format=rgba,drawgrid=w=45:h=45:c=gray[grid];^
[grid][text]overlay -frames 1 -y yellow.png

ffmpeg -lavfi color=c=black@0:s=720x360,format=rgba,^
drawtext=text="DOWN":x=145:y=173:fontsize=24,^
drawtext=text="UP":x=523:y=173:fontsize=24,^
v360=e:e:roll=90:interp=near,^
drawtext=text="NORTH":x=322:y=173:fontsize=24,^
scroll=hpos=0.5,^
drawtext=text="SOUTH":x=322:y=173:fontsize=24,^
drawtext=text="WEST":x=510:y=173:fontsize=24,^
drawtext=text="EAST":x=154:y=173:fontsize=24[text];^
color=c=magenta:s=720x360,format=rgba,drawgrid=w=45:h=45:c=gray[grid];^
[grid][text]overlay -frames 1 -y magenta.png

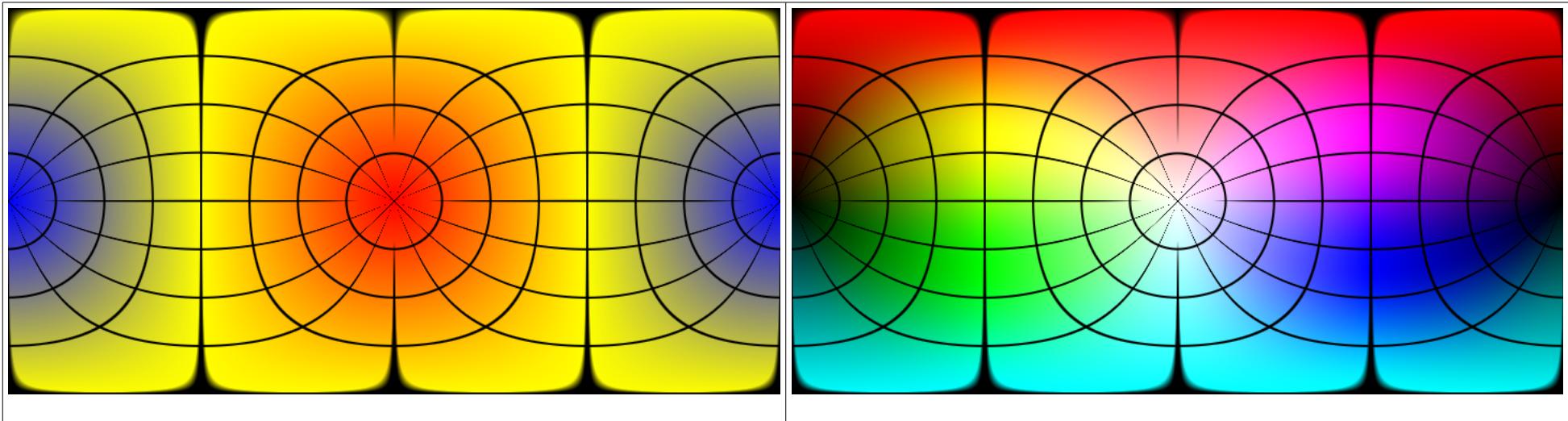
ffmpeg -i yellow.png -i magenta.png -lavfi [0][1]vstack -y double_equirectangular.png

pause
```

Make equirectangular north-south pole images, useful for testing of wormhole simulations:

```
ffmpeg -lavfi gradients=s=720x360:c0=red:c1=yellow:c2=blue:nb_colors=3:x0=0:y0=0:x1=0:y1=359,format=rgba,drawgrid=w=45:h=45:t=2:x=-1:y=-1:c=black,v360=e:e:pitch=90 -frames 1 -y poles1.png  
ffmpeg -lavfi colorspectrum=s=720x360:type=all,format=rgba,drawgrid=w=45:h=45:t=2:x=-1:y=-1:c=black,v360=e:e:pitch=90 -frames 1 -y poles2.png  
  
pause
```

These are the output images:



26 Canon 5D-Mark4

Resolution: 6.720 x 4.480, RAW 14-bit

26.1 All Canon 5D-Mark4 video modes for PAL video system

MOV / MP4	Movie rec. size	Size	Frame rate	Bit rate	YUV/bit	Image compression
MOV	4K 25.00P MJPG	4096x2160	25	480 Mbps	4:2:2 / 8 bit	MJPG yuvj422p
	4K 24.00P MJPG	4096x2160	24	480 Mbps	4:2:2 / 8 bit	MJPG yuvj422p
	FHD 50.00P ALL-I	1920x1080	50	174 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
	FHD 50.00P IPB	1920x1080	50	59 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 25.00P ALL-I	1920x1080	25	88 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
	FHD 25.00P IPB	1920x1080	25	30 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 24.00P ALL-I	1920x1080	24	88 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
	FHD 24.00P IPB	1920x1080	24	30 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	HD 100.0P ALL-I	1280x720	100	154 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
MP4	FHD 50.00P IPB	1920x1080	50	58 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 25.00P IPB	1920x1080	25	29 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 25.00P IPB	1920x1080	25	12 Mbps	4:2:0 / 8 bit	IPB ("Light", this is a stronger compression) h264 yuvj420p
	FHD 24.00P IPB	1920x1080	24	29 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p

26.2 All Canon 5D-Mark4 video modes for NTSC video system

MOV / MP4	Movie rec. size	Size	Frame rate	Bit rate	YUV/bit	Image compression
MOV	4K 29.97P MJPG	4096x2160	29.97	480 Mbps	4:2:2 / 8 bit	MJPEG yuvj422p
	4K 23.98P MJPG	4096x2160	23.98	480 Mbps	4:2:2 / 8 bit	MJPEG yuvj422p
	4K 24.00P MJPG	4096x2160	24	480 Mbps	4:2:2 / 8 bit	MJPEG yuvj422p
	FHD 59.94P ALL-I	1920x1080	59.94	174 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
	FHD 59.94P IPB	1920x1080	59.94	59 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 29.97P ALL-I	1920x1080	29.97	88 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
	FHD 29.97P IPB	1920x1080	29.97	30 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 23.98P ALL-I	1920x1080	23.98	88 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
	FHD 23.98P IPB	1920x1080	23.98	30 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 24.00P ALL-I	1920x1080	24	88 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
	FHD 24.00P IPB	1920x1080	24	30 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	HD 119.9P ALL-I	1280x720	119.9	154 Mbps	4:2:0 / 8 bit	ALL-I h264 yuvj420p
MP4	FHD 59.94P IPB	1920x1080	59.94	58 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 29.97P IPB	1920x1080	29.97	29 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 29.97P IPB	1920x1080	29.97	12 Mbps	4:2:0 / 8 bit	IPB ("Light", this is a stronger compression) h264 yuvj420p
	FHD 23.98P IPB	1920x1080	23.98	29 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p
	FHD 24.00P IPB	1920x1080	24.00	29 Mbps	4:2:0 / 8 bit	IPB h264 yuvj420p

Important note: If the size of a video exceeds 4GB, it can only be downloaded to the computer with "EOS Utility" software.

26.3 Canon 5D-Mark4 Field of view

Lens	Autofocus	4:3 Full-frame field of view 6.720 x 4.480 (36mm x 24mm)	16:9 FHD Field of view 1920x1080 (36mm x 20.25mm)	17:9 C4K Field of view 4096x2160 (21.94mm x 11.57mm)
Sigma EX DG 4.5mm f/2.8	no	180.0° x 180.0°	180.0° x 180.0°	180.0° x 169.3°
Meike 6-11mm f/3.5	no	180.0° x 180.0° @ 9.6mm	180.0° x 180.0° @ 8.3mm	180.0° x 137.9° @ 6mm
Nippon Kogaku 8mm f/2.8	no	180.0° x 180.0°	180.0° x 158.5°	171.7° x 90.5°
Sigma EX DG 8mm f/3.5	no	180.0° x 180.0°	180.0° x 160.6°	174.0° x 91.7°
Canon EF 8-15mm f/4.0	no	180.0° x 180.0° @ 8mm	180.0° x 159.2° @ 8mm	172.5° x 90.9° @ 8mm
Canon EF 11-24mm f/4.0	yes	117.1° x 95.0° - 73.7° x 53.1°	117.1° x 85.3° - 73.7° x 45.7°	89.8° x 55.5° - 49.1° x 27.1°
Sigma 14mm f/1.8	yes	104.3° x 81.2°	104.3° x 71.7°	76.1° x 44.9°
Canon CN-E 24mm T1.5 L F	no	73.7° x 53.1°	73.7° x 45.7°	49.1° x 27.1°
Sigma 24mm f/1.4	yes	73.7° x 53.1°	73.7° x 45.7°	49.1° x 27.1°
Laowa 24mm f/14	no	73.7° x 53.1°	73.7° x 45.7°	49.1° x 27.1°
Canon EF 24-70mm f/4.0	yes	73.7° x 53.1° - 28.8° x 19.5°	73.7° x 45.7° - 28.8° x 16.5°	49.1° x 27.1° - 17.8° x 9.45°
Sigma 50mm f/1.4	yes	39.6° x 27.0°	39.6° x 22.9°	14.7° x 13.2°
Canon CN-E 50mm T1.3 L F	no	39.6° x 27.0°	39.6° x 22.9°	14.7° x 13.2°
Canon CN-E 85mm T1.3 L F	no	23.9° x 16.1°	23.9° x 13.6°	11.6° x 7.79°
Canon EF 100mm f/2.8	yes	20.4° x 13.7°	20.4° x 11.6°	12.5° x 6.62°
Canon EF 100-400mm f/4.5-5.6	yes	20.4° x 13.7° - 5.15° x 3.44°	20.4° x 11.6° - 5.15° x 2.90°	12.5° x 6.62° - 3.14° x 1.66°
Canon EF 200mm f/2.0	yes	10.3° x 6.87°	10.3° x 5.80°	6.28° x 3.31°
Canon EF 400mm f/2.8	yes	5.15° x 3.44°	5.15° x 2.90°	3.14° x 1.66°
+ 1.4x Teleconverter 560mm f/4.0	yes	3.68° x 2.46°	3.68° x 2.07°	2.24° x 1.18°
+ 2x Teleconverter 800mm f/5.6	yes	2.58° x 1.72°	2.58° x 1.45°	1.57° x 0.83°
Canon EF 500mm f/4.0	yes	4.12° x 2.75°	4.12° x 2.32°	2.51° x 1.33°
+ 1.4x Teleconverter 700mm f/5.6	yes	2.95° x 1.96°	2.95° x 1.66°	1.80° x 0.95°
+ 2x Teleconverter 1000mm f/8.0	yes	2.06° x 1.38°	2.06° x 1.16°	1.26° x 0.66°
Takahashi FS-128 1040mm f/8.1	no	1.98° x 1.32°	1.98° x 1.12°	1.21° x 0.64°
TMB Refractor 2057mm f/9	no	1.00° x 0.67°	1.00° x 0.56°	0.61° x 0.32°

Fisheye lenses: $\text{Field_of_view_in_degrees} = 180^\circ * \pi / \text{Image_circle_diameter}$

Normal lenses: $\text{Field_of_view_in_degrees} = 2 * \arctan(x / 2f)$

with x = image width or height in mm

26.4 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>

AF Operation	Notes
ONE SHOT	For non-moving objects
AI FOCUS	This decides automatically if the object is moving or not. Not recommended.
AI SERVO	For moving objects, recommended as default.

Drive Mode	Notes
-	Single shot
H	7 Pictures per second
-	3 Pictures per second
S	3 Pictures per second, silent mode
Clock Symbol	10 Seconds self timer
Clock Symbol 2	2 Seconds self timer

Orange Menu (second from right) --> 3 --> Custom Controls

Shutter Button	leave as-is
AF_ON Button	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.
* Button	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.
Multi_Controller	Set to "Direct AF point selection"
AF Area Selection Button	Set to "Direct AF area selection"
SET Button	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=4IPrb5w1Zw>

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.

(1) Spot AF (Square with Point)	Very small, good choice if you take pictures through branches
(2) Single Point AF	This is the default setting, very precise and fast, if you manage to hold the point on the object.
(3) Expand AF area (5 Points)	Recommended method for moving objects. The center point is prioritized and if this point loses focus, then one of the neighbor points is used. Place the center point on the eye of the object.
(4) AF Expand Surround (9 Points)	Same as (3), but 8 neighbor points.
(5) Zone AF (9 or 12 Points)	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.
(6) Large Zone	Same as (5), but more points.
(7) Auto AF Selection (all 61 Points)	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.

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=VOiQs1UEI8>

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>

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".

Tracking Sensitivity	This is by far the most important parameter! It describes how easily the focus can move away from the previously found focus.
Accel/decel tracking	This is difficult to understand. Best if you leave it at 0.
AF pt auto switching	This describes, how fast the camera switches from one AF point to a neighbor AF point. He leaves it at 0, which means deactivated.

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)

Page 1 --> Lens aberration correction	He switches all options off, so that the pictures can be saved faster.
Page 2 --> Auto Lighting Optimizer	OFF
Page 3 --> High ISO speed NR	OFF
Page 2 --> ISO speed settings	AUTO 100 - 12800 for both ranges
Page 1 --> Release Shutter without card	Disable

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

Birds in front of trees or water	Use a single AF point and hold it on the object.
Birds in the sky	It's easier to use all AF points.

27 Panasonic LUMIX GH5S

Manual: ftp://ftp.panasonic.com/camera/om/dc-gh5s_adv_en_om.pdf

27.1 GH5S Autofocus

There are several focusing modes:

Focus mode	[Continuous AF]	
AFS/AFF/AFC	MODE 1	Continuous AF only works while recording video.
	MODE 2	Continuous AF works all the time.
	OFF	The camera maintains the focus position at the start of recording. While recording video it's possible to focus by pressing the shutter button halfway.
MF	ON or OFF	You can focus manually.

AFS	"Auto Focus Single". The focus stays locked while the shutter button is pressed halfway, so you can perform recording while changing the composition.
AFF	"Auto Focus Flexible". If the subject moves while the shutter button is pressed halfway, the focus is readjusted automatically according to the movement of the subject.
AFC	"Auto Focus Continuous". In this mode, while the shutter button is pressed halfway, focusing is constantly performed to match the movement of the subject.

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 Menue --> Oeration --> 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.

27.2 GH5S Record formats

Record Format	File	Bits	Video Codec	Pixel Format	Audio Codec	Anamorphic	VFR	H LG	Notes
AVCHD	*.mts	8	h264 (High)	yuv420p progressive	ac3 48000Hz, stereo, fltp, 192 kb/s	no	some	no	This data format is suitable for when playing back on a high-definition TV, etc.
MP4	*.mp4	8	h264 (High)	yuv420p tv, bt709, progressive	aac 48000Hz, stereo, fltp, 124 kb/s	no	no	no	This data format is suitable for when playing back on a PC, etc.
MP4 HEVC (High Efficiency Video Coding, h265)	*.mp4	10	hevc (Main 10)	yuv420p10le tv, bt2020nc / bt2020 / arib-std-b67	aac 48000Hz, stereo, fltp, 124 kb/s	no	no	yes	This data format is for HDR motion picture and suitable for playback on a HDR (H LG format)-compatible TV.
MP4 (LPCM) 420 / 8bit / LongGOP	*.mp4	8	h264 (High)	Luminance Level 0-255: yuvj420p, pc, bt709, progressive Luminance Level 16-235 or 16-255: yuv420p, tv, bt709, progressive	pcm_s16be 48000Hz, stereo, s16, 1536 kb/s	possible	some	no	The MP4 data format for image editing. LPCM is an uncompressed audio format: 48kHz * 16bit * 2 channels = 1536 kb/s
MP4 (LPCM) 422 / 10bit / LongGOP	*.mp4	10	h264 (High 4:2:2)	yuv422p10le tv or pc, bt709, progressive		no	no	yes	
MP4 (LPCM) 422 / 10bit / ALL-I	*.mp4	10	h264 (High 4:2:2 Intra)	yuv422p10le tv or pc, bt709, progressive		possible	no	yes	
MOV 420 / 8bit / LongGOP	*.mov	8	h264 (High)	Luminance Level 0-255: yuvj420p, pc, bt709, progressive Luminance Level 16-235 or 16-255: yuv420p, tv, bt709, progressive	pcm_s16be 48000Hz, stereo, s16, 1536 kb/s	possible	some	no	Data format for image editing.
MOV 422 / 10bit / LongGOP	*.mov	10	h264 (High 4:2:2)	yuv422p10le tv or pc, bt709, progressive		no	no	yes	
MOV 422 / 10bit / ALL-I	*.mov	10	h264 (High 4:2:2 Intra)	yuv422p10le tv or pc, bt709, progressive		possible	no	yes	

Note: All VFR (Variable Frame Rate) modes are 8-bit, all H LG (Hybrid Log Gamma) modes are 10-bit.

Note: There aren't any significant differences between "MP4 (LPCM)" and "MOV".

27.3 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 zebras 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.

See also <https://xtremestuff.net/where-in-the-world-is-middle-gray/>

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:

Reflectance	IRE	10-bit code value
0% (black)	7.3	128
18% (middle gray)	42	433
90% (white)	61	602
absolute clipped superwhite	80 IRE	

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 recommended 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:

https://pro-av.panasonic.net/en/cinema_camera_varicam_eva/support/pdf/VARICAM_V-Log_V-Gamut.pdf

VLog table:

$\text{Stops} = \log_2(\text{in} / 0.18)$	$\text{In} = 0.18 * 2^{\text{Stops}}$	out	$\text{out} * 1023$	$\text{IRE} = -7.24 + 116.33 * \text{out}$	curve
$-\infty$	0 (0% black)	0.1250	128.0	7.3	linear
-8	0.000703125	0.1290	131.9	7.8	
-7	0.00140625	0.1329	135.9	8.2	
-6	0.0028125	0.1407	144.0	9.1	
-5	0.005625	0.1565	160.1	11.0	
-4.17	0.01	0.1810	185.2	13.8	
-4.17	0.01	0.1810	185.2	13.8	
-4	0.01125	0.1878	192.1	14.6	
-3	0.0225	0.2346	240.0	20.1	
-2	0.045	0.2915	298.2	26.7	
-1	0.09	0.3554	363.5	34.1	logarithmic
0	0.18 (18% middle gray)	0.4233	433.0	42	
1	0.36	0.4936	504.9	50.2	
2	0.72	0.5650	578.0	58.5	
2.32	0.90 (90% white)	0.5882	601.7	61	
3	1.44	0.6371	651.7	66.9	
4	2.88	0.7095	725.8	75.3	
4.559	4.24246	0.7500	767.2	80 (maximum value for VLog-L)	
5	5.76	0.7820	800.0	83.7 (only available in VLog)	
6	11.52	0.8546	874.3	92.2 (only available in VLog)	
7	23.04	0.9273	948.6	100.6 (only available in VLog)	
8	46.08	1.0000	1023	109.1 (only available in VLog)	

Note: $\log_2(x) = \ln(x) / \ln(2)$

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",

$out = 5.6 * in + 0.125 \quad (in < cut1)$

$out = c * \log_{10}(in + b) + d \quad (in \geq cut1) \quad \text{with } cut1 = 0.01, b = 0.00873, c = 0.241514, d = 0.598206, 0 \leq out \leq 1$

The function for reverting compressed V-Log data to linear refection is as follows.

With V-Log data as "in" and linear reflection as "out",

$in = (out - 0.125) / 5.6 \quad (out < cut2)$

$in = \text{pow}(10.0, ((out - d) / c)) - b \quad (out \geq cut2) \quad \text{with } cut2 = 0.181, 0 \leq out \leq 1$

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=clum='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

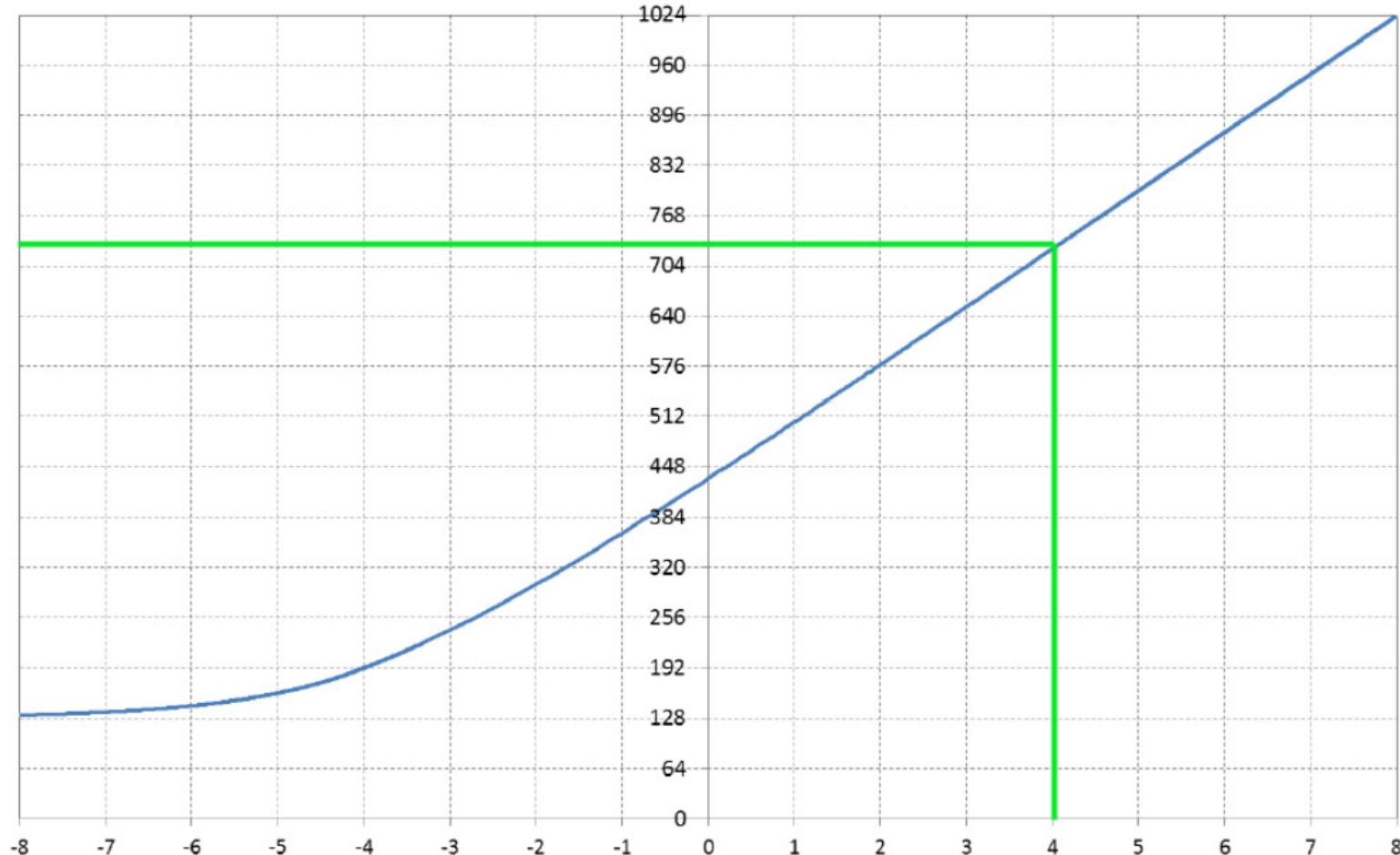
pause
```

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).

27.4 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.

$$E' = r * \sqrt{E} \text{ for } 0 \leq E \leq 1$$

$$E' = a * \ln(E - b) + c \text{ for } 1 < E$$

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

and 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 digits, however 3 or 4 digits would be sufficient.

Another LUT (not free) for converting from BT.2020 color space (HLG) to Rec.709 is available here: <https://cc-lut.hotglue.me/>

HLG Table:

$\text{Stops} = \log_2(\text{in} / 0.18)$	$\text{In} = 0.18 * 2^{\text{Stops}} = 0.9 * E$	$E = 0.2 * 2^{\text{Stops}}$	out	$\text{out} * 1023$	curve
$-\infty$	0 (0% black)	0 (0% black)	0	0	square root
-8	0.000703125	0.00078125	0.0140	14.3	
-7	0.00140625	0.0015625	0.0198	20.2	
-6	0.0028125	0.003125	0.0280	28.6	
-5	0.005625	0.00625	0.0395	40.4	
-4	0.01125	0.0125	0.0559	57.2	
-3	0.0225	0.025	0.0791	80.9	
-2	0.045	0.05	0.1118	114.4	
-1	0.09	0.1	0.158	161.8	
0	0.18 (18% middle gray)	0.2	0.2121	217.0	
1	0.36	0.4	0.3162	323.5	
2	0.72	0.8	0.4472	457.5	
2.32	0.90 (90% white)	1.0 (reference white level)	0.5	511.5	logarithmic
2.32	0.90 (90% white)	1.0 (reference white level)	0.5	511.5	
3	1.44	1.6	0.6089	622.9	
4	2.88	3.2	0.7513	768.6	
5	5.76	6.4	0.8837	904.0	
5.91	10.8	12	1	1023	

Note: $\log_2(x) = \ln(x) / \ln(2)$

See also <https://xtremestuff.net/recording-editing-with-hybrid-log-gamma-part-1/>

and <https://xtremestuff.net/recording-editing-with-hybrid-log-gamma-part-2/>

27.5 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.

27.6 GH5S Recommended settings

	Cinelike-D	VLOG-L	HLG (Hybrid Log Gamma)
"Phot Style" in Exif Data		Unknown (10)	Unknown (13)
Contrast	0	[NA]	[NA]
Sharpness (1)	-5 ?	-5 ?	-5 ?
Noise Reduction (2)	-5	-5	-5
Saturation	-5	[NA]	-5
Hue	0	[NA]	0
Luminance Level	0-1023 (0-255 for 8 bit)	fixed at 32-200 (128-800 for 10 bit) (3)	fixed at 0-1023
Zebras	100%	75%	90%
Exposure compensation		+1	
Possible ISO range for "Dual Native ISO Settings" = Low	80 - 800	320 - 1600	320 - 1600
Possible ISO range for "Dual Native ISO Settings" = High	800 - 204800	1600 - 25600	1600 - 204800
Dynamic range [F-Stops]	10.5	11.58	11.5
Notes		Best choice for video post processing!	Best choice for night sky!

(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 greater 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

27.7 GH5S Custom settings C1, C2, C3-1, C3-2, C3-2

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:

	Rec Format	Pixel	fps	ISO, Photo Style	Exposure Mode, Exposure time	System Frequency	Application
C1	[4K/10bit/150M/25p] 422 / 10Bit / Long GOP	3840x2160 4K	25	400, STD	M, 1/50s	50.00Hz (PAL)	For 4K videos
C2	[C4K/10bit/150M/25p] 422 / 10Bit / Long GOP	4096x2160 C4K	25	Auto, STD	M, 1/50s	50.00Hz (PAL)	For C4K videos
C3-1	[4K/A/150M/25p] 422 / 10bit / Long GOP	3328x2496 Anamorphic	25	51200 HLG, NR=-5	M, 1/25s	50.00Hz (PAL)	For meteor astronomy with SpeedBooster and Nippon Kogaku 8mm f/2.8 fisheye lens
C3-2	[FHD/8bit/100M/25p] 420 / 8Bit / Long GOP	1920x1080 FHD	125	51200, STD	M, 1/125s	50.00Hz (PAL)	For video astronomy: Variable framerate: 125 Ex.Tele Conv is OFF, but can be set to ON
C3-3	[FHD/8bit/100M/25p] 420 / 8Bit / Long GOP	1920x1080 FHD	25	51200, STD	M, 1/25s (1/2s - 1/25s)	50.00Hz (PAL)	For video astronomy: Variable framerate: off Ex.Tele Conv is OFF, but can be set to ON

My function key settings:

Fn1 Sound Rec Level Adj.

Fn2 Histogram

Fn3 Waveform Monitor

Fn4 Zebras

Fn5 Ex. Tele Conv.

27.8 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.

27.9 GH5S Master pedestal level

Creative Video > Master Pedestal Level

-	This side creates a high contrast image with a crisp atmosphere.
0	Standard
+	This side creates a slightly misty atmosphere.

This function is not available when Photo Style is set to V-Log L

27.10 GH5S Video size

Mode	Resolution	Read-out Chip Size	Diagonal Size	Number of Pixels
4K	3820 x 2160 (16:9)	18.8mm x 10.6mm	21.6mm	8251200
C4K	4096 x 2160 (17:9)	19.2mm x 10.12mm	21.7mm	8847360
Anamorphic	3328 x 2496 (4:3)	17.3mm x 13.0mm	21.6mm	8306688
FHD	1920 x 1080 (16:9)	18.8mm x 10.6mm (1)	21.6mm	2062800
FHD with 2.1x Extra Tele Conversion	1920 x 1080 (16:9)	8.95mm x 5.05mm	10.3mm	2062800

(1) Read-out chip size is smaller when frame rate is greater than 200

27.11 GH5S Mechanical / electronic shutter

Rec > Shutter Type

Shutter Type	ISO	Exposure Time Range
Mechanical shutter	100-204800	60s - 1/8000s
Electronic shutter	204800	1/30s - 1/16000s
	102400	1/15s - 1/16000s
	51200	1/8s - 1/16000s
	25600	1/4s - 1/16000s
	12800	1/2s - 1/16000s
	100 - 6400	1s - 1/16000s

27.12 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. These longer exposure times are suitable for filming stars in the night sky.

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

27.13 GH5S Variable frame rate

System Frequency	Rec Quality	Available Framerates
59.94Hz (NTSC)	[4K/8bit/100M/30p] [FHD/24M/30p]	2 15 26 28 30 32 34 45 60
	[FHD/8bit/100M/60p]	2 30 56 58 60 62 64 90 120 150 180 210 240
	[FHD/8bit/100M/30p]	2 15 26 28 30 32 34 45 60 75 90 105 120 135 150 165 180 195 210 225 240
	[4K/8bit/100M/24p] [FHD/24M/24p]	2 12 20 22 24 26 28 36 48 60
	[FHD/8bit/100M/24p]	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
50.00Hz (PAL)	[4K/8bit/100M/25p] [FHD/24M/25p]	2 12 21 23 25 27 30 37 60
	[FHD/8bit/100M/50p]	2 25 46 48 50 52 54 75 100 125 150 200 240
	[FHD/8bit/100M/25p]	2 12 21 23 25 27 30 37 50 62 75 87 100 112 125 137 150 175 200 225 240
24.00Hz (CINEMA)	[4K/8bit/100M/24p] [C4K/8bit/100M/24p]	2 12 20 22 24 26 28 36 48 60
	[FHD/8bit/100M/24p]	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

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.

Note: Variable frame rate isn't possible with 10-bit modes.

27.14 Recording duration on SD cards

Mbps	MB/s	MB/min	128GB card	256GB card	512GB card	640GB card (128GB + 512GB)	1024GB card (512GB + 512GB)
400	50	3000	43 min = 0.7 h	87 min = 1.4 h	174 min = 2.9 h	218 min = 3.6 h	349 min = 5.8 h
200	25	1500	87 min = 1.4 h	174 min = 2.9 h	349 min = 5.8 h	436 min = 7.2 h	699 min = 11.6 h
150	18.75	1125	116 min = 1.9 h	233 min = 3.9 h	466 min = 7.8 h	582 min = 9.7 h	932 min = 15.5 h
100	12.5	750	174 min = 2.9 h	349 min = 5.8 h	699 min = 11.6 h	873 min = 14.5 h	1398 min = 23.3 h
72	9	540	242 min = 4.0 h	485 min = 8.1 h	970 min = 16.2 h	1213 min = 20.2 h	1941 min = 32.3 h
28	3.5	210	624 min = 10.4 h	1248 min = 20.8 h	2496 min = 41.6 h	3120 min = 52.0 h	4993 min = 83.2 h
24	3	180	728 min = 12.1 h	1456 min = 24.3 h	2912 min = 48.5 h	3640 min = 60.6 h	5825 min = 97.0 h
20	2.5	150	873 min = 14.5 h	1747 min = 29.1 h	3495 min = 58.2 h	4369 min = 72.8 h	6990 min = 116.5 h
17	2.125	127.5	1028 min = 17.1 h	2056 min = 34.3 h	4112 min = 68.5 h	5397 min = 89.9 h	8224 min = 137.0 h

27.15 GH5S Cable remote trigger

The cable remote trigger has a 2.5mm connector with 4 contacts:

Tip contact	not connected
2nd contact	not connected
3rd contact	not pressed: 38.5 kΩ to ground (Difference to "half pressed" is 33 kΩ) half pressed: 5.5 kΩ to ground (Difference to "full pressed" is 3.3 kΩ) full pressed: 2.2 kΩ to ground
Outer contact	ground

27.16 GH5S Cheap chinese battery adapters

If a $10k\Omega$ 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.

27.17 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].

27.18 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

27.19 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>

27.20 GH5S Field of view with and without SpeedBooster 0.64x

Lens	Effective focal length and f/ratio with SpeedBooster 0.64x	Field of view GH5S 4K (18.8mm x 10.6mm)	Field of view GH5S 4K (18.8mm x 10.6mm) with SpeedBooster 0.64x	Field of view GH5S Anamorphic (4:3) 4K (17.3mm x 13.0mm)	Field of view GH5S Anamorphic (4:3) 4K (17.3mm x 13.0mm) with SpeedBooster 0.64x	Field of view Full Frame (36mm x 24mm)
Sigma EX DG 4.5mm f/2.8	2.9mm f/1.8	180.0° x 155.1°	180.0° x 180.0°	180.0° x 180.0°	180.0° x 180.0°	180.0° x 180.0°
Meike 6-11mm f/3.5	3.8mm-7.0mm f/2.2	180.0° x 126.4° @ 6mm	180.0° x 180.0° @ 6mm	180.0° x 155.0° @ 6mm	180.0° x 180.0° @ 6mm	180.0° x 180.0° @ 6mm
Nippon Kogaku 8mm f/2.8	5.1mm f/1.8	147.1° x 83.0°	180.0° x 129.6°	135.4° x 101.7°	180.0° x 159.0°	180.0° x 180.0°
Sigma EX DG 8mm f/3.5	5.1mm f/2.2	149.1° x 84.1°	180.0° x 131.3°	137.2° x 103.1°	180.0° x 161.1°	180.0° x 180.0°
Olympus M.Zuiko 8mm f/1.8	--	153.8° x 86.7°	--	141.5° x 106.4°	--	--
Canon EF 8-15mm f/4.0	5.1mm f/2.5	147.8° x 83.3° @ 8mm	180.0° x 130.2° @ 8mm	136.0° x 102.2° @ 8mm	180.0° x 160.0° @ 8mm	180.0° x 180.0° @ 8mm
Canon EF 11-24mm f/4.0	7.0mm-15.4mm f/2.5	81.0° x 24.9° - 42.8° x 24.9°	106.3° x 73.9° - 62.9° x 38.1°	76.4° x 61.2° - 39.6° x 30.3°	101.7° x 85.4° - 58.8° x 45.9°	117.1° x 95.0° - 73.7° x 53.1°
Olympus M.Zuiko 8-25mm f/4.0	--	99.2° x 67.0° - 41.2° x 23.9°	--	94.5° x 78.2° - 38.2° x 29.1°	--	--
Leica DG 12-60mm f/2.8-4.0	--	76.1° x 47.7° - 17.8° x 10.1°	--	71.6° x 56.9° - 16.4° x 12.4°	--	--
Sigma 14mm f/1.8	9.0mm f/1.1	67.8° x 41.5°	92.7° x 61.2°	63.4° x 49.8°	88.0° x 71.9°	104.3° x 81.2°
Canon CN-E 24mm T1.5 L F	15.4mm T 0.96	42.8° x 24.9°	62.9° x 38.1°	39.6° x 30.3°	58.8° x 45.9°	73.7° x 53.1°
Sigma 24mm f/1.4	15.4mm f/0.9	42.8° x 24.9°	62.9° x 38.1°	39.6° x 30.3°	58.8° x 45.9°	73.7° x 53.1°
Laowa 24mm f/14	15.4mm f/9.0	42.8° x 24.9°	62.9° x 38.1°	39.6° x 30.3°	58.8° x 45.9°	73.7° x 53.1°
Canon EF 24-70mm f/4.0	15.4mm-44.8mm f/2.5	42.8° x 24.9° - 15.3° x 8.66°	62.9° x 38.1° - 23.7° x 13.5°	39.6° x 30.3° - 14.1° x 10.6°	58.8° x 45.9° - 21.9° x 16.5°	73.7° x 53.1° - 28.8° x 19.5°
Sigma 50mm f/1.4	32mm f/0.9	21.3° x 12.1°	32.7° x 18.8°	19.6° x 14.8°	30.3° x 23.0°	39.6° x 27.0°
Canon CN-E 50mm T1.3 L F	32mm T 0.83	21.3° x 12.1°	32.7° x 18.8°	19.6° x 14.8°	30.3° x 23.0°	39.9° x 27.0°
Canon CN-E 85mm T1.3 L F	54.4mm T 0.83	12.6° x 7.14°	19.6° x 11.1°	11.6° x 8.75°	18.1° x 13.6°	23.9° x 16.1°
Canon EF 100mm f/2.8	64mm f/1.8	10.7° x 6.07°	16.7° x 9.47°	9.89° x 7.44°	15.4° x 11.6°	20.4° x 13.7°
Canon EF 100-400mm f/4.5-5.6	64mm-256mm f/2.8-3.5	10.7° x 6.07° - 2.69° x 1.52°	16.7° x 9.47° - 4.21° x 2.37°	9.89° x 7.44° - 2.48° x 1.86°	15.4° x 11.6° - 3.87° x 2.91°	20.4° x 13.7° - 5.15° x 3.44°
Canon EF 200mm f/2.0	128mm f/1.2	5.38° x 3.04°	8.40° x 4.74°	4.95° x 3.72°	7.73° x 5.81°	10.3° x 6.87°

Canon EF 400mm f/2.8	256mm f/1.8	2.69° x 1.52°	4.21° x 2.37°	2.48° x 1.86°	3.87° x 2.91°	5.15° x 3.44°
+ 1.4x Teleconverter 560mm f/4.0	--	1.92° x 1.08°	--	1.77° x 1.33°	--	3.68° x 2.46°
+ 2x Teleconverter 800mm f75.6	--	1.35° x 0.76°	--	1.24° x 0.93°	--	2.58° x 1.72°
Canon EF 500mm f/4.0	320mm f/2.5	2.15° x 1.21°	3.37° x 1.90°	1.98° x 1.49°	3.10° x 2.33°	4.12° x 2.75°
+ 1.4x Teleconverter 700mm f/5.6	--	1.54° x 0.87°	--	1.42° x 1.06°	--	2.95° x 1.96°
+ 2x Teleconverter 1000mm f/8.8	--	1.08° x 0.61°	--	0.99° x 0.74°	--	2.06° x 1.38°
Takahashi FS-128 1040mm f/8.1	666mm f/5.2	1.04° x 0.58°	1.62° x 0.91°	0.95° x 0.72°	1.49° x 1.12°	1.98° x 1.32°
TMB Refractor 2057mm f/9	1316mm f/5.8	0.52° x 0.30°	0.82° x 0.46°	0.48° x 0.36°	0.75° x 0.57°	1.00° x 0.67°

Fisheye lenses: $\text{Field_of_view_in_degrees} = 180^\circ \times x / \text{Image_circle_diameter}$

Normal lenses: $\text{Field_of_view_in_degrees} = 2 \times \arctan(x / 2f)$

with x = image width or height in mm

MTF data for 0.64x SpeedBooster: https://www.dpreview.com/files/p/articles/9958618251/metabones_tables_1.jpeg

27.21 GH5S, all 77 video modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[C4K/8bit/150M/60p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	4096x2160	59.94p	150 Mbps	4:2:0/8 bit	Long GOP
[C4K/10bit/150M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	4096x2160	29.97p	150 Mbps	4:2:2/10 bit	Long GOP
[C4K/8bit/100M/30p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	4096x2160	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[C4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	4096x2160	23.98p	400 Mbps	4:2:2/10 bit	ALL-Intra
[C4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	4096x2160	23.98p	150 Mbps	4:2:2/10 bit	Long GOP
[C4K/8bit/100M/24p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	4096x2160	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/8bit/150M/60p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3840x2160	59.94p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/ALL-I/400M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	29.97p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	29.97p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/8bit/100M/30p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	3840x2160	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	23.98p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	23.98p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	3840x2160	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/72M/30p]	MP4 HEVC	no	yes	59.94Hz (NTSC)	3840x2160	29.97p	72 Mbps	4:2:0/10 bit	Long GOP
[4K/72M/24p]	MP4 HEVC	no	yes	59.94Hz (NTSC)	3840x2160	23.98p	72 Mbps	4:2:0/10 bit	Long GOP
[4K/100M/30p]	MP4	no	no	59.94Hz (NTSC)	3840x2160	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/100M/24p]	MP4	no	no	59.94Hz (NTSC)	3840x2160	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/A/150M/60p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3328x2496	59.94p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/A/400M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	29.97p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/A/150M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	29.97p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/A/100M/30p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3328x2496	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/A/400M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	23.98p	400 Mbps	4:2:2/10 bit	ALL-Intra

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[4K/A/150M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	23.98p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/A/100M/24p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3328x2496	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/28M/60p]	AVCHD	no	no	59.94Hz (NTSC)	1920x1080	59.94p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/17M/60i]	AVCHD	no	no	59.94Hz (NTSC)	1920x1080	59.94i	17 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/30p]	AVCHD	yes	no	59.94Hz (NTSC)	1920x1080	59.94i	24 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/24p]	AVCHD	yes	no	59.94Hz (NTSC)	1920x1080	23.98p	24 Mbps	4:2:0/8 bit	Long GOP
[FHD/ALL-I/200M/60p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	59.94p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/60p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	59.94p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/8bit/100M/60p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	1920x1080	59.94p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/ALL-I/200M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	29.97p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	29.97p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/8bit/100M/30p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	1920x1080	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/ALL-I/200M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	23.98p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	23.98p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	1920x1080	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/28M/60p]	MP4	no	no	59.94Hz (NTSC)	1920x1080	59.94p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/20M/30p]	MP4	no	no	59.94Hz (NTSC)	1920x1080	29.97p	20 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/24p]	MP4	no	no	59.94Hz (NTSC)	1920x1080	23.98p	24 Mbps	4:2:0/8 bit	Long GOP
[C4K/8bit/150M/50p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	4096x2160	50.00p	150 Mbps	4:2:0/8 bit	Long GOP
[C4K/10bit/150M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	4096x2160	25.00p	150 Mbps	4:2:2/10 bit	Long GOP
[C4K/8bit/100M/25p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	4096x2160	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/8bit/150M/50p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	3840x2160	50.00p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/ALL-I/400M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3840x2160	25.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3840x2160	25.00p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/8bit/100M/25p]	MP4 (LPCM) / MOV	yes	no	50.00Hz (PAL)	3840x2160	25.00p	100 Mbps	4:2:0/8 bit	Long GOP

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[4K/72M/25p]	MP4 HEVC	no	yes	50.00Hz (PAL)	3840x2160	25.00p	72 Mbps	4:2:0/10 bit	Long GOP
[4K/100M/25p]	MP4	no	no	50.00Hz (PAL)	3840x2160	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/A/150M/50p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	3328x2496	50.00p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/A/400M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3328x2496	25.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/A/150M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3328x2496	25.00p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/A/100M/25p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	3328x2496	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/28M/50p]	AVCHD	no	no	50.00Hz (PAL)	1920x1080	50.00p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/17M/50i]	AVCHD	no	no	50.00Hz (PAL)	1920x1080	50.00i	17 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/25p]	AVCHD	yes	no	50.00Hz (PAL)	1920x1080	50.00i	24 Mbps	4:2:0/8 bit	Long GOP
[FHD/ALL-I/200M/50p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	50.00p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/50p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	50.00p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/8bit/100M/50p]	MP4 (LPCM) / MOV	yes	no	50.00Hz (PAL)	1920x1080	50.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/ALL-I/200M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	25.00p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	25.00p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/8bit/100M/25p]	MP4 (LPCM) / MOV	yes	no	50.00Hz (PAL)	1920x1080	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/28M/50p]	MP4	no	no	50.00Hz (PAL)	1920x1080	50.00p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/20M/25p]	MP4	no	no	50.00Hz (PAL)	1920x1080	25.00p	20 Mbps	4:2:0/8 bit	Long GOP
[C4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	4096x2160	24.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[C4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	4096x2160	24.00p	150 Mbps	4:2:2/10 bit	Long GOP
[C4K/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	24.00Hz (CINEMA)	4096x2160	24.00p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3840x2160	24.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3840x2160	24.00p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	24.00Hz (CINEMA)	3840x2160	24.00p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/A/400M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3328x2496	24.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/A/150M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3328x2496	24.00p	150 Mbps	4:2:2/10 bit	Long GOP

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[4K/A/100M/24p]	MP4 (LPCM) / MOV	no	no	24.00Hz (CINEMA)	3328x2496	24.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/ALL-I/200M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	1920x1080	24.00p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	1920x1080	24.00p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	24.00Hz (CINEMA)	1920x1080	24.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/24p]	MP4	no	no	24.00Hz (CINEMA)	1920x1080	24.00p	24 Mbps	4:2:0/8 bit	Long GOP

27.22 GH5S, all C4K 8 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[C4K/8bit/150M/60p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	4096x2160	59.94p	150 Mbps	4:2:0/8 bit	Long GOP
[C4K/8bit/100M/30p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	4096x2160	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[C4K/8bit/100M/24p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	4096x2160	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[C4K/8bit/150M/50p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	4096x2160	50.00p	150 Mbps	4:2:0/8 bit	Long GOP
[C4K/8bit/100M/25p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	4096x2160	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[C4K/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	24.00Hz (CINEMA)	4096x2160	24.00p	100 Mbps	4:2:0/8 bit	Long GOP

27.23 GH5S, all C4K 10 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[C4K/10bit/150M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	4096x2160	29.97p	150 Mbps	4:2:2/10 bit	Long GOP
[C4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	4096x2160	23.98p	400 Mbps	4:2:2/10 bit	ALL-Intra
[C4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	4096x2160	23.98p	150 Mbps	4:2:2/10 bit	Long GOP
[C4K/10bit/150M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	4096x2160	25.00p	150 Mbps	4:2:2/10 bit	Long GOP
[C4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	4096x2160	24.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[C4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	4096x2160	24.00p	150 Mbps	4:2:2/10 bit	Long GOP

27.24 GH5S, all 4K 8 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[4K/8bit/150M/60p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3840x2160	59.94p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/8bit/100M/30p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	3840x2160	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	3840x2160	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/100M/30p]	MP4	no	no	59.94Hz (NTSC)	3840x2160	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/100M/24p]	MP4	no	no	59.94Hz (NTSC)	3840x2160	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/8bit/150M/50p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	3840x2160	50.00p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/8bit/100M/25p]	MP4 (LPCM) / MOV	yes	no	50.00Hz (PAL)	3840x2160	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/100M/25p]	MP4	no	no	50.00Hz (PAL)	3840x2160	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	24.00Hz (CINEMA)	3840x2160	24.00p	100 Mbps	4:2:0/8 bit	Long GOP

27.25 GH5S, all 4K 10 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[4K/ALL-I/400M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	29.97p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	29.97p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	23.98p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3840x2160	23.98p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/72M/30p]	MP4 HEVC	no	yes	59.94Hz (NTSC)	3840x2160	29.97p	72 Mbps	4:2:0/10 bit	Long GOP
[4K/72M/24p]	MP4 HEVC	no	yes	59.94Hz (NTSC)	3840x2160	23.98p	72 Mbps	4:2:0/10 bit	Long GOP
[4K/ALL-I/400M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3840x2160	25.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3840x2160	25.00p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/72M/25p]	MP4 HEVC	no	yes	50.00Hz (PAL)	3840x2160	25.00p	72 Mbps	4:2:0/10 bit	Long GOP
[4K/ALL-I/400M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3840x2160	24.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/10bit/150M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3840x2160	24.00p	150 Mbps	4:2:2/10 bit	Long GOP

27.26 GH5S, all anamorphic 8 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[4K/A/150M/60p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3328x2496	59.94p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/A/100M/30p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3328x2496	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/A/100M/24p]	MP4 (LPCM) / MOV	no	no	59.94Hz (NTSC)	3328x2496	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/A/150M/50p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	3328x2496	50.00p	150 Mbps	4:2:0/8 bit	Long GOP
[4K/A/100M/25p]	MP4 (LPCM) / MOV	no	no	50.00Hz (PAL)	3328x2496	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[4K/A/100M/24p]	MP4 (LPCM) / MOV	no	no	24.00Hz (CINEMA)	3328x2496	24.00p	100 Mbps	4:2:0/8 bit	Long GOP

27.27 GH5S, all anamorphic 10 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[4K/A/400M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	29.97p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/A/150M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	29.97p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/A/400M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	23.98p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/A/150M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	3328x2496	23.98p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/A/400M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3328x2496	25.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/A/150M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	3328x2496	25.00p	150 Mbps	4:2:2/10 bit	Long GOP
[4K/A/400M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3328x2496	24.00p	400 Mbps	4:2:2/10 bit	ALL-Intra
[4K/A/150M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	3328x2496	24.00p	150 Mbps	4:2:2/10 bit	Long GOP

27.28 GH5S, all FHD 8 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[FHD/28M/60p]	AVCHD	no	no	59.94Hz (NTSC)	1920x1080	59.94p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/17M/60i]	AVCHD	no	no	59.94Hz (NTSC)	1920x1080	59.94i	17 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/30p]	AVCHD	yes	no	59.94Hz (NTSC)	1920x1080	59.94i	24 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/24p]	AVCHD	yes	no	59.94Hz (NTSC)	1920x1080	23.98p	24 Mbps	4:2:0/8 bit	Long GOP
[FHD/8bit/100M/60p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	1920x1080	59.94p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/8bit/100M/30p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	1920x1080	29.97p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	59.94Hz (NTSC)	1920x1080	23.98p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/28M/60p]	MP4	no	no	59.94Hz (NTSC)	1920x1080	59.94p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/20M/30p]	MP4	no	no	59.94Hz (NTSC)	1920x1080	29.97p	20 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/24p]	MP4	no	no	59.94Hz (NTSC)	1920x1080	23.98p	24 Mbps	4:2:0/8 bit	Long GOP
[FHD/28M/50p]	AVCHD	no	no	50.00Hz (PAL)	1920x1080	50.00p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/17M/50i]	AVCHD	no	no	50.00Hz (PAL)	1920x1080	50.00i	17 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/25p]	AVCHD	yes	no	50.00Hz (PAL)	1920x1080	50.00i	24 Mbps	4:2:0/8 bit	Long GOP
[FHD/8bit/100M/50p]	MP4 (LPCM) / MOV	yes	no	50.00Hz (PAL)	1920x1080	50.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/8bit/100M/25p]	MP4 (LPCM) / MOV	yes	no	50.00Hz (PAL)	1920x1080	25.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/28M/50p]	MP4	no	no	50.00Hz (PAL)	1920x1080	50.00p	28 Mbps	4:2:0/8 bit	Long GOP
[FHD/20M/25p]	MP4	no	no	50.00Hz (PAL)	1920x1080	25.00p	20 Mbps	4:2:0/8 bit	Long GOP
[FHD/8bit/100M/24p]	MP4 (LPCM) / MOV	yes	no	24.00Hz (CINEMA)	1920x1080	24.00p	100 Mbps	4:2:0/8 bit	Long GOP
[FHD/24M/24p]	MP4	no	no	24.00Hz (CINEMA)	1920x1080	24.00p	24 Mbps	4:2:0/8 bit	Long GOP

27.29 GH5S, all FHD 10 bit modes

Rec Quality	Rec Format	VFR	HLG	System frequency	Size	Frame rate	Bit rate	YUV/bit	Image compression
[FHD/ALL-I/200M/60p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	59.94p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/60p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	59.94p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/ALL-I/200M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	29.97p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/30p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	29.97p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/ALL-I/200M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	23.98p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/24p]	MP4 (LPCM) / MOV	no	yes	59.94Hz (NTSC)	1920x1080	23.98p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/ALL-I/200M/50p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	50.00p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/50p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	50.00p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/ALL-I/200M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	25.00p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/25p]	MP4 (LPCM) / MOV	no	yes	50.00Hz (PAL)	1920x1080	25.00p	100 Mbps	4:2:2/10 bit	Long GOP
[FHD/ALL-I/200M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	1920x1080	24.00p	200 Mbps	4:2:2/10 bit	ALL-Intra
[FHD/10bit/100M/24p]	MP4 (LPCM) / MOV	no	yes	24.00Hz (CINEMA)	1920x1080	24.00p	100 Mbps	4:2:2/10 bit	Long GOP

28 Immersive 360° videos

See also: Hugh Hou, How to Film & Edit 360° Drone Video - Insta360 ONE X2 + DaVinci Resolve 17 In-Depth Tutorial
https://www.youtube.com/watch?v=ypB2J_p1Ayk&t=0s

Instructions for Insta360 ONE X2: <http://onlinemanual.insta360.com/onex2/en-us/camera/basic>

29 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 with crop values for different field of view:

Field of View	Top and left border	Width and height
180°	176	2104
185°	144	2168
190°	116	2224
195°	88	2280
200°	60	2336

30 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.

Unfortunately the camera has no setting for exposure correction.

English instruction manual: <https://kodakpixpro.com/resources/cameras/360-vr/sp360-4k/docs/sp360-4k-usermanual-en-v03.pdf>

Possible video resolutions in fisheye mode:

2880x2880	2048x2048	1440x1440	1072x1072	720x720
30fps	30fps	30fps or 60fps	30fps or 60fps	120fps

Possible still image resolutions:

2M	4M	8M
1920x1080 (16:9)	2304x1728 (4:3)	2880x2880 (1:1)

Crop values for different field of view, for 2880x2880 resolution:

Field of View	Top and left border	Width and height
180°	210	2456
185°	174	2528
190°	142	2592
195°	106	2664
200°	74	2728

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! Most probably it has also lost all settings. Check the video mode!

When recording, the LED is blinking red.

Test report (german): https://www.digitaleyes.de/Tests/Testbericht_Kodak_Pixpro_SP360_4K_360-Grad-Actioncam/9819

30.1 Remote control

Remote control via Bluetooth is possible with the PIXPRO 360 VR SUITE, which can be downloaded here:

<https://kodakpixpro.com/Europe/de/support/downloads/?id=a04>

But it's much easier to use the RR-BK01 remote control.

When recording a 2880x2880 fisheye image or video, it's important that the left switch of the remote control is in the lower position!

The right switch selects between still image or video.

Important note: If the left switch is in the upper position, the recording format would be 3840x2160 and not the whole 235° fisheye circle is visible!

30.2 Using the PIXPRO SP360 4K Camera as a Webcam

<https://kodakpixpro.com/Europe/support/downloads.php>

Download and install the driver: "PIXPRO SP360 4K UVC Driver for Skype"

If you want to use the camera as a webcam, the battery must be full. In webcam mode the camera isn't powered over the USB cable.

The camera has two USB modes: "Mass Storage" and "Webcam". The active mode is shown in the display when the USB cable is plugged in. A USB2 port is sufficient. Warning: Live streaming doesn't work with an active USB2 extension cable.

The USB mode can be changed as follows:

- Switch on the camera (with no USB cable connected)
- Press the "down" button until the "gearwheel" symbol is shown in the display, then press the red button.
- Press the "down" button until "Mass Storage" or "Webcam" is shown, then press the red button.
- Now toggle the mode with the "down" button, then press the red button for confirmation.
- Find the "Exit" icon and press the red button.

Search the camera name with FFmpeg:

```
ffmpeg -list_devices 1 -f dshow -i dummy  
pause
```

Output:

```
[dshow @ 000001b35549efc0] "PIXPRO SP360 4K" (video)  
[dshow @ 000001f42d740040] "Mikrofon (PIXPRO SP360 4K)" (audio)
```

List the available video modes:

```
ffmpeg -list_options 1 -f dshow -i video="PIXPRO SP360 4K"  
pause
```

Output:

```
[dshow @ 0000027514cb0040] vcodec=mjpeg min s=3840x2160 fps=5 max s=3840x2160 fps=5
[dshow @ 0000027514cb0040] vcodec=mjpeg min s=2880x2880 fps=5 max s=2880x2880 fps=5
[dshow @ 0000027514cb0040] vcodec=mjpeg min s=2048x2048 fps=5 max s=2048x2048 fps=5
[dshow @ 0000027514cb0040] vcodec=mjpeg min s=1440x1440 fps=5 max s=1440x1440 fps=15
[dshow @ 0000027514cb0040] vcodec=mjpeg min s=1920x1080 fps=5 max s=1920x1080 fps=15
[dshow @ 0000027514cb0040] vcodec=mjpeg min s=1280x720 fps=5 max s=1280x720 fps=30
[dshow @ 0000027514cb0040] vcodec=mjpeg min s=640x360 fps=5 max s=640x360 fps=30
```

List the available audio modes:

```
ffmpeg -list_options 1 -f dshow -i audio="Mikrofon (PIXPRO SP360 4K)"
```

Output:

```
[dshow @ 00000219c8c5f400] ch= 2, bits=16, rate= 44100
[dshow @ 00000219c8c5f400] ch= 1, bits=16, rate= 44100
[dshow @ 00000219c8c5f400] ch= 2, bits=16, rate= 32000
[dshow @ 00000219c8c5f400] ch= 1, bits=16, rate= 32000
[dshow @ 00000219c8c5f400] ch= 2, bits=16, rate= 22050
[dshow @ 00000219c8c5f400] ch= 1, bits=16, rate= 22050
[dshow @ 00000219c8c5f400] ch= 2, bits=16, rate= 11025
[dshow @ 00000219c8c5f400] ch= 1, bits=16, rate= 11025
[dshow @ 00000219c8c5f400] ch= 2, bits=16, rate= 8000
[dshow @ 00000219c8c5f400] ch= 1, bits=16, rate= 8000
[dshow @ 00000219c8c5f400] ch= 2, bits= 8, rate= 44100
[dshow @ 00000219c8c5f400] ch= 1, bits= 8, rate= 44100
[dshow @ 00000219c8c5f400] ch= 2, bits= 8, rate= 22050
[dshow @ 00000219c8c5f400] ch= 1, bits= 8, rate= 22050
[dshow @ 00000219c8c5f400] ch= 2, bits= 8, rate= 11025
[dshow @ 00000219c8c5f400] ch= 1, bits= 8, rate= 11025
[dshow @ 00000219c8c5f400] ch= 2, bits= 8, rate= 8000
[dshow @ 00000219c8c5f400] ch= 1, bits= 8, rate= 8000
[dshow @ 00000219c8c5f400] ch= 2, bits=16, rate= 48000
[dshow @ 00000219c8c5f400] ch= 1, bits=16, rate= 48000
[dshow @ 00000219c8c5f400] ch= 2, bits=16, rate= 96000
[dshow @ 00000219c8c5f400] ch= 1, bits=16, rate= 96000
```

Capture 235° fisheye content from the webcam, using the "sdl2" output device:

```
ffmpeg -f dshow -video_size 1440x1440 -framerate 15 -vcodec mjpeg -i video="PIXPRO SP360 4K" -vf format=bgra -f sdl2 -
pause
```

Note: SDL2 can't play audio.

Capture 235° fisheye content from the webcam and pipe the output to FFplay. The "-b:v" option is used to set the bitrate:

```
ffmpeg -f dshow -video_size 1440x1440 -framerate 15 -vcodec mjpeg -i video="PIXPRO SP360 4K" -b:v 10M -f nut - | ffplay
-fs -autoexit -
pause
```

Same as before, but with audio:

```
ffmpeg -f dshow -video_size 1440x1440 -framerate 5 -vcodec mjpeg -i video="PIXPRO SP360 4K" -f dshow -sample_rate 48000
-sample_size 16 -channels 2 -i audio="Mikrofon (PIXPRO SP360 4K)" -b:v 10M -f nut - | ffplay -fs -autoexit -
pause
```

Capture 235° fisheye content from the webcam and pipe the output to VLC. The "-b:v" option is used to set the bitrate:

```
ffmpeg -f dshow -video_size 1440x1440 -framerate 15 -vcodec mjpeg -i video="PIXPRO SP360 4K" -b:v 10M -f mpegts - |
"C:\Program Files\VideoLAN\VLC\vlc.exe" -
pause
```

Same as before, but with audio:

```
ffmpeg -f dshow -video_size 1440x1440 -framerate 15 -vcodec mjpeg -i video="PIXPRO SP360 4K" -f dshow -sample_rate 48000
-sample_size 16 -channels 2 -i audio="Mikrofon (PIXPRO SP360 4K)" -b:v 10M -f mpegts - | "C:\Program
Files\VideoLAN\VLC\vlc.exe" -
pause
```

Capture 235° fisheye content from the webcam, convert it to equirectangular and pipe the output to VLC. The "-b:v" option is used to set the bitrate:

```
ffmpeg -f dshow -video_size 1440x1440 -framerate 15 -vcodec mjpeg -i video="PIXPRO SP360 4K" -vf v360=fisheye:e:ih_fov=235:iv_fov=235:pitch=-90 -b:v 10M -f mpegs - | "C:\Program Files\VideoLAN\VLC\vlc.exe" - pause
```

Same as before, but with audio:

```
ffmpeg -f dshow -video_size 1440x1440 -framerate 15 -vcodec mjpeg -i video="PIXPRO SP360 4K" -f dshow -sample_rate 48000 -sample_size 16 -channels 2 -i audio="Mikrofon (PIXPRO SP360 4K)" -vf v360=fisheye:e:ih_fov=235:iv_fov=235:pitch=-90 -b:v 10M -f mpegs - | "C:\Program Files\VideoLAN\VLC\vlc.exe" - pause
```

30.3 Convert 235° fisheye image to equirectangular panorama

```
set "IN=114_0020.jpg"          :: Input video
set "FOV=235"                  :: Input field of view in degrees
set "YAW=135"                  :: Yaw rotation angle in degrees
set "OUT=out.png"              :: Equirectangular output image or video

ffmpeg -i %IN% -vf v360=input=fisheye:ih_fov=%FOV%:iv_fov=%FOV%:output=equirect:rorder=pyr:yaw=%YAW%:pitch=-90,crop=iw:ih*%FOV%/360:y=0 -y %OUT%

pause
```

31 Chinese 360° Panoramic camera



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 its 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:

- Is the battery fully charged?
- Is the 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.

32 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)

For still images the size is 5376x2688.

Instruction manual: <https://support.theta360.com/en/manual/v/>

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. If you tick the "top/bottom correction" box, the video will automatically be rotated so that the ground is at the bottom. This is possible because the camera has a built-in 3-axis gravity sensor. 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>

32.1 Plugins

Plugins are available here: <https://pluginstore.theta360.com/#official>

32.2 Bluetooth remote control

It's possible to use a "Joby Impulse" Bluetooth remote controller for taking pictures.

1. Power on the camera.
2. If the white light is blinking, continue with step 8.
3. Press the mode button longer than 3s. If the white light is blinking, continue with step 8.
4. Run the "Ricoh Theta Basic app" and click on Start --> Plugin_Management.
5. Connect the camera with USB cable.
6. Select the "Remote App" and click "ok".
7. Disconnect the USB cable. Continue with step 1.
8. Hold the button of the remote control until the remote starts blinking red. Then release the button.
9. After 10-20 seconds, you will hear a sound from the camera. Now the remote is paired with the camera.
10. It will stay paired if the camera goes into sleep mode. You can wake it up with the remote. Sleep time can only be set via smartphone to 3, 5, 7, 10 minutes or off.
11. After power down / power up, it must be paired again.

For instructions see also: <https://www.thetalab.ricoh/en/howto/tips/remotecontroller/>

32.3 Change the viewing direction

```
ffmpeg -i R0010079.jpg -lavfi v360=e:e:yaw=180 -y out.jpg  
  
exiftool -ProjectionType="equirectangular" out.jpg  
  
pause
```

Note: ProjectionType=equirectangular is sufficient for Facebook to recognize the image as spherical 360°.

See also: <http://echeng.com/articles/editing-360-photos-injecting-metadata/>

32.4 How to hide something from a 360° image

Method 1: Edit the image manually with IrfanView:

- IrfanView --> Edit --> Show Paint Dialog --> "Clone Tool"
- Select a suitable brush diameter, then right-click at the replacement area, then left-click (or draw) at the place that shall be covered.

Method 2: Rotate the spot to the center, apply "delogo" filter, then rotate back:

```
set "X=1926"      :: Delogo x position  
set "Y=830"       :: Delogo y position  
set "W=200"        :: Delogo width  
set "H=160"        :: Delogo height  
  
ffmpeg -ss 10 -i R0010057_er.mp4 -lavfi v360=e:e:pitch=-90,delogo=%X%:%Y%:%W%:%H%,v360=e:e:pitch=90 -frames 1 -y out.png  
  
pause
```

Note: The drawback is that there is some loss of resolution, because each v360 filter uses interpolation.

Method 3: Cover a circular area at the bottom with a uniform color:

```
set "IN=R0010080.jpg"    :: Input image
set "DIA=30"              :: Diameter of covered circular area in degrees
set "C=brown"             :: Color
set "OUT=out.jpg"          :: Output image

ffmpeg -i %IN% -lavfi drawbox=c=%C%:t=fill:y=ih*(1-%DIA%/360):w=iw:h=ih*%DIA%/360 -y %OUT%

exiftool -ProjectionType="equirectangular" %OUT%

pause
```

Method 4: Cover a circular area at the bottom with a uniform color, using an unsharp edge:

```
set "IN=R0010080.jpg"    :: Input image
set "SIZE=5376x2688"      :: Input and output size
set "DIA=30"              :: Diameter of covered circular area in degrees
set "EDGE=10"              :: Width of unsharp edge in degrees
set "C=brown"             :: Color
set "OUT=out.jpg"          :: Output image

ffmpeg -f lavfi -i color=brown:size=%SIZE% -frames 1 -y bg.png

ffmpeg -f lavfi -i nullsrc=size=%SIZE% -lavfi format=gray8,geq='clip(128+256/(H*%EDGE%/180)*(Y-H*(1-%DIA%/360)),0,255)' -frames 1 -y mask.png

ffmpeg -i %IN% -i bg.png -i mask.png -lavfi [0]format=gbrp[fg];[2]format=gbrp[mask];[fg][1][mask]maskedmerge -y %OUT%

pause
```

Method 5: Cover a circular area at the bottom with data from a different part of the same image, using an unsharp edge:

```
set "IN=R0010186_neu.jpg"      :: Input image
set "SIZE=5376x2688"          :: Input and output size
set "DIA=23"                  :: Diameter of covered circular area in degrees,
                                :: must be larger than diameter of photograph plus EDGE
set "EDGE=5"                  :: Width of unsharp edge in degrees
set "Y=45"                    :: Yaw angle where cover area is taken from
set "P=23"                    :: Pitch angle where cover area is taken from,
                                :: must not be smaller than DIA
set "OUT=out.jpg"             :: Output image

ffmpeg -f lavfi -i nullsrc=size=%SIZE% -lavfi format=gray8,geq='clip(128+256/(H*%EDGE%/180)*(Y-H*(1-%DIA%/360)),0,255)'
-frames 1 -y mask.png

ffmpeg -i %IN% -i mask.png -lavfi [0]format=gbrp,split[fg][bg];[bg]v360=e:e:yaw=%Y%:pitch=%P%[bg];[1]format=gbrp[mask];
[fg][bg]maskedmerge -y %OUT%

exiftool -ProjectionType="equirectangular" %OUT%
pause
```

A good viewer for 360° images is the "Ricoh Theta Basic app".

360° videos can be viewed with VLC Player.

I can't recommend 5KPlayer. It uses CPU power even when not in use.

32.5 Make an equirectangular 360° image or video

This batch file extracts an image from a video and converts it into an equirectangular panorama image, which is 360° wide and 180° high:

```
set "IN=R0010032.mp4"          :: Input video from Ricoh Theta V
set "H=1920"                   :: Height of input video
set "FOV=192.2"                :: Horizontal and vertical field of view of the fisheye lenses in degrees
set "C=5"                      :: Width of interpolation band in degrees, must be smaller or equal than (FOV-180°)
set "T=100"                     :: Time in seconds where the image is extracted from the input video
set "OUT=equi.png"              :: Output image

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, and convert to equirectangular projection

ffmpeg -ss %T% -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,v360=input=e:output=e:yaw=90:pitch=90" -frames 1 -y %OUT%

pause
```

Note: The "Ricoh Theta Basic app" gives a better stitching result than the above FFmpeg batch file, because it also corrects alignment errors of the two fisheye objectives. Use this batch file to extract an image:

```
set "IN=R0010032_er.mp4"        :: Input video from Ricoh Theta V
set "T=100"                      :: Time in seconds where the image is extracted from the input video
set "OUT=out.png"                 :: Output image

ffmpeg -ss %T% -i %IN% -frames 1 -y %OUT%

pause
```

This batch file extracts an image from the equirectangular video and lets the camera make a horizontal 360° rotation:

```
set "IN=R0010032_er.mp4"          :: Input video from Ricoh Theta V
set "T=100"                      :: Time in seconds where the image is extracted from the input video
set "D=20"                        :: Duration for one 360° revolution
set "FR=50"                       :: Output framerate
set "H_FOV=80"                    :: Horizontal field of view
set "V_FOV=45"                    :: Vertical field of view
set "W=1920"                      :: Width of output video
set "H=1080"                      :: Height of output video
set "OUT=out.mp4"                 :: Output video

ffmpeg -ss %T% -i %IN% -frames 1 -y image.png

ffmpeg -loop 1 -i image.png -vf fps=%FR%,scroll=h='1/(%FR%*%D%)',v360=e:flat:h_fov=%H_FOV%:v_fov=%V_FOV%:w=%W%:h=%H% -t %D% -y %OUT%

pause
```

32.6 Make a "Little-Planet" picture or video

This batch file extracts an image from a video and converts it into a "Little Planet" image with 270° fisheye projection:

```
set "IN=R0010032.mp4"          :: Input video from Ricoh Theta V
set "H=1920"                   :: Height of input video
set "FOV=192.2"                :: Horizontal and vertical field of view of the fisheye lenses in degrees
set "C=5"                      :: Width of interpolation band in degrees, must be smaller or equal than (FOV-180°)
set "LP=270"                    :: Little planet output field of view in degrees
set "T=100"                     :: Time in seconds where the image is extracted from the input video
set "OUT=out.png"               :: Output image

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, and convert to little planet projection

ffmpeg -ss %T% -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,v360=input=e:output=fisheye:h_fov=%LP%:v_fov=%LP%:yaw=90" -frames 1 -y %OUT%

pause
```

Note: The "Ricoh Theta Basic app" gives a better stitching result than the above FFmpeg batch file. Use this batch file to make the "Little Planet" image:

```
set "IN=R0010032_er.mp4"        :: Input video from Ricoh Theta V
set "LP=270"                     :: Little planet output field of view in degrees
set "T=100"                      :: Time in seconds where the image is extracted from the input video
set "OUT=out.png"                :: Output image

ffmpeg -ss %T% -i %IN% -lavfi "v360=input=e:output=fisheye:h_fov=%LP%:v_fov=%LP%:pitch=-90" -frames 1 -y %OUT%

pause
```

I found it useful to use an additional distortion that shrinks the center of the image and enlarges the horizon in radial direction. This distortion is applied to the equirectangular image, before transforming it to 270° fisheye projection. The resulting image seems to have the camera higher above the ground:

```
set "IN=R0010032_er.mp4"      :: Input video from Ricoh Theta V
set "W=3840"                  :: Width of input video
set "H=1920"                  :: Height of input video
set "LP=270"                   :: Little planet output field of view in degrees
set "S=0.6"                    :: Strength of additional distortion (can be disabled with s=0)
set "T=100"                    :: Time in seconds where the image is extracted from the input video
set "OUT=out2.png"            :: Output image

rem Create the remap_x file (this is a simple identity map)

ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16,geq='X' -frames 1 -y remap_x.pgm

rem Create the remap_y file, using the function y = (1 - s) * y + s * y^3 where -1 < y < 1

ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16,geq='st(0,2*Y/%H%-1);%H%/2+%H%/2*((1-%S%)*ld(0)+%S%*pow(ld(0),3))' -frames 1 -y remap_y.pgm

rem Apply the remap filter and transform to "Little Planet" projection

ffmpeg -ss %T% -i %IN% -i remap_x.pgm -i remap_y.pgm -lavfi
"format=pix_fmts=rgb24,remap,v360=input=e:output=fisheye:h_fov=%LP%:v_fov=%LP%:pitch=-90" -frames 1 -y %OUT%
pause
```

This is the same for still pictures (with higher resolution 5376x2688):

```
set "IN=R0010175.jpg"          :: Input image from Ricoh Theta V
set "W=5376"                  :: Width of input video
set "H=2688"                  :: Height of input video
set "LP=270"                   :: Little planet output field of view in degrees
set "S=0.6"                    :: Strength of additional distortion (can be disabled with s=0)
set "OUT=out.png"              :: Output image

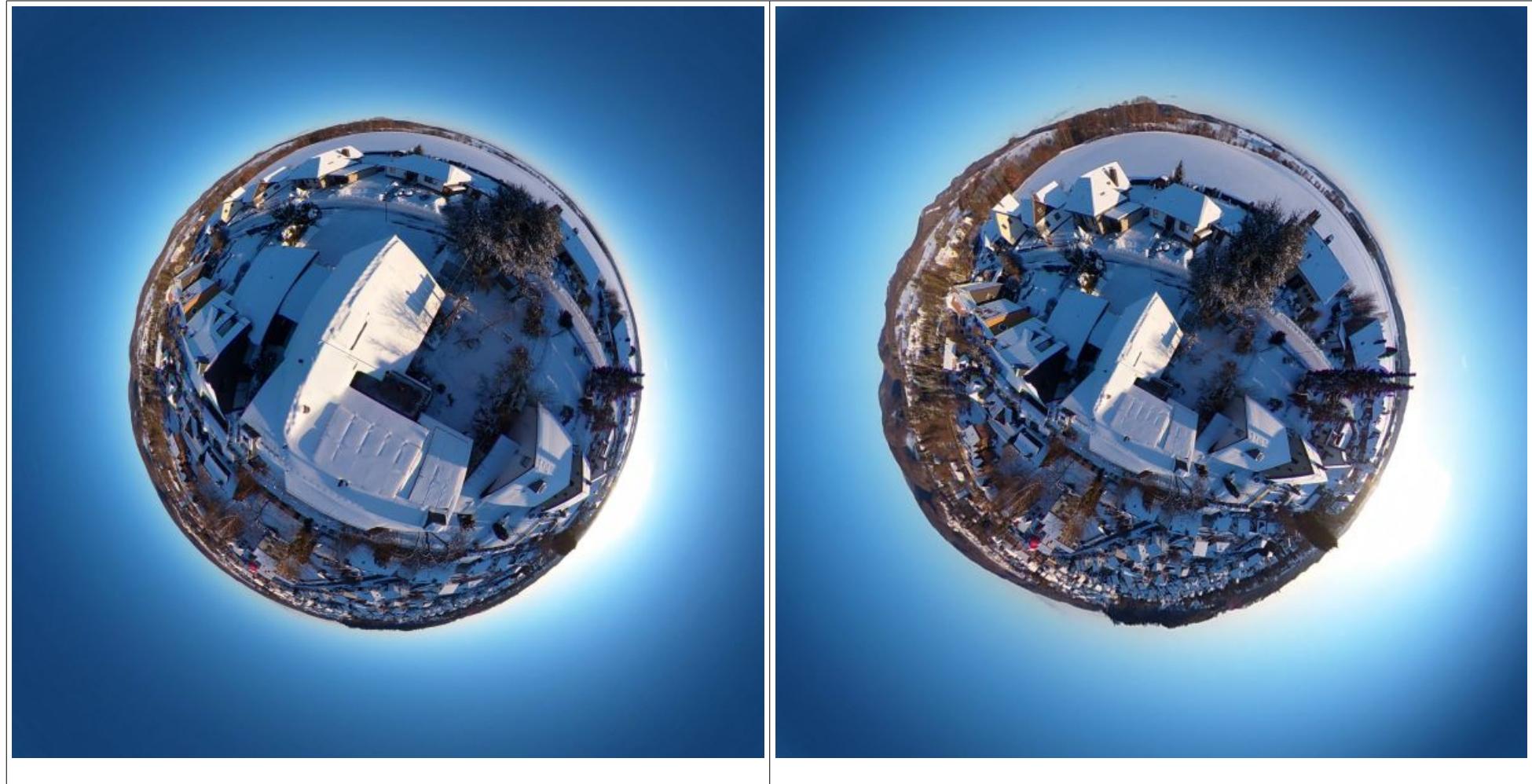
rem Create the remap_x file (this is a simple identity map)
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16,geq='X' -frames 1 -y remap_x.pgm

rem Create the remap_y file, using the function y = (1 - s) * y + s * y^3 where -1 < y < 1
ffmpeg -f lavfi -i nullsrc=size=%W%x%H% -vf format=gray16,geq='st(0,2*Y/%H%-1);%H%/2+%H%/2*((1-%S%)*ld(0)+%S%*pow(ld(0),3))' -frames 1 -y remap_y.pgm

rem Apply the remap filter and transform to "Little Planet" projection
ffmpeg -i %IN% -i remap_x.pgm -i remap_y.pgm -lavfi "format=pix_fmts=rgb24,remap,v360=input=e:output=fisheye:h_fov=%LP%:v_fov=%LP%:pitch=-90" -frames 1 -y %OUT%

pause
```

Here are the results, left side with $s=0$ and right side with $s=0.6$



32.7 Live Streaming with Ricoh Theta V

See the instructions here: https://support.theta360.com/uk/manual/v/content/streaming/streaming_02.html

But one very important thing is missing in these instructions. For Windows 10 you need a special streaming driver "RICOH THETA V/Z1 2.0 Live Streaming Driver V2 for Windows 10" which is available here: <https://theta360.guide/special/>

Press the mode button at the camera until the blue "Live" LED is on.

Connect the camera to a USB3 port. It doesn't work with a USB2 port!

Note: The streaming driver generates a test image when the camera is not connected. You can use the "RICOH THETA V/Z1 FullHD" and "RICOH THETA V/Z1 4K" devices even when the camera is not connected.

Then for example in OBS Studio there are three cameras available:

- RICOH THETA V (this one doesn't work)
- RICOH THETA V/Z1 FullHD
- RICOH THETA V/Z1 4K

When the stream is used by OBS Studio, the blue "Live" LED is blinking.

Note: If only the first camera (RICOH THETA V) is listed, then probably you forgot to install the special streaming driver.

Note: In VLC Player, none of the three cameras is working. Please correct me if I'm wrong.

See also: <https://topics.theta360.com/en/news/2022-02-15/>

See also: <https://devs.theta360.guide/ricoh-theta-live-streaming-driver-updated/>

See also: <https://www.youtube.com/watch?v=dp5BLbtKVwk>

<https://community.theta360.guide/t/rtsp-streaming-plug-in/4556/11?page=2>

The same three cameras are also listed in FFmpeg:

```
ffmpeg -list_devices 1 -f dshow -i dummy  
  
pause
```

Output (among others):

```
[dshow @ 0000022c95acefc0] "RICOH THETA V" (video)  
[dshow @ 0000022c95acefc0] "RICOH THETA V/Z1 FullHD" (video)  
[dshow @ 0000022c95acefc0] "RICOH THETA V/Z1 4K" (video)  
[dshow @ 0000022c95acefc0] "Mikrofon (RICOH THETA V)" (audio)
```

Note: The first camera "RICOH THETA V" doesn't work in FFmpeg.

List the available video modes:

```
ffmpeg -list_options 1 -f dshow -i video="RICOH THETA V/Z1 FullHD"  
  
ffmpeg -list_options 1 -f dshow -i video="RICOH THETA V/Z1 4K"  
  
pause
```

Output:

```
[dshow @ 000001b87e5f4d40] pixel_format=nv12 min s=1920x960 fps=30 max s=1920x960 fps=30  
[dshow @ 0000017d23094d40] pixel_format=nv12 min s=3840x1920 fps=30 max s=3840x1920 fps=30
```

List all available audio modes:

```
ffmpeg -list_options 1 -f dshow -i audio="Mikrofon (RICOH THETA V)"  
  
pause
```

Output:

```
[dshow @ 000001eb787bf400] ch= 2, bits=16, rate= 44100
[dshow @ 000001eb787bf400] ch= 1, bits=16, rate= 44100
[dshow @ 000001eb787bf400] ch= 2, bits=16, rate= 32000
[dshow @ 000001eb787bf400] ch= 1, bits=16, rate= 32000
[dshow @ 000001eb787bf400] ch= 2, bits=16, rate= 22050
[dshow @ 000001eb787bf400] ch= 1, bits=16, rate= 22050
[dshow @ 000001eb787bf400] ch= 2, bits=16, rate= 11025
[dshow @ 000001eb787bf400] ch= 1, bits=16, rate= 11025
[dshow @ 000001eb787bf400] ch= 2, bits=16, rate= 8000
[dshow @ 000001eb787bf400] ch= 1, bits=16, rate= 8000
[dshow @ 000001eb787bf400] ch= 2, bits= 8, rate= 44100
[dshow @ 000001eb787bf400] ch= 1, bits= 8, rate= 44100
[dshow @ 000001eb787bf400] ch= 2, bits= 8, rate= 22050
[dshow @ 000001eb787bf400] ch= 1, bits= 8, rate= 22050
[dshow @ 000001eb787bf400] ch= 2, bits= 8, rate= 11025
[dshow @ 000001eb787bf400] ch= 1, bits= 8, rate= 11025
[dshow @ 000001eb787bf400] ch= 2, bits= 8, rate= 8000
[dshow @ 000001eb787bf400] ch= 1, bits= 8, rate= 8000
[dshow @ 000001eb787bf400] ch= 2, bits=16, rate= 48000
[dshow @ 000001eb787bf400] ch= 1, bits=16, rate= 48000
[dshow @ 000001eb787bf400] ch= 2, bits=16, rate= 96000
[dshow @ 000001eb787bf400] ch= 1, bits=16, rate= 96000
```

Capture live video and show it on the screen, three different solutions, but they all have the problem that the content isn't shown as 360° so that the user can't change the viewing direction. "sdl2" and FFplay don't support 360° output. VLC does support 360° output only if spherical metadata is inserted in MP4 or Matroska containers (this isn't possible for -f mpegts). Also VLC doesn't have a command line option for forcing 360° output.

```
ffmpeg -f dshow -i video="RICOH THETA V/Z1 FullHD" -vf format=bgra -f sdl2 -
ffmpeg -f dshow -i video="RICOH THETA V/Z1 FullHD" -f nut - | ffplay -fs -autoexit -
ffmpeg -f dshow -i video="RICOH THETA V/Z1 FullHD" -b:v 10M -f mpegts - | "C:\Program Files\VideoLAN\VLC\vlc.exe" -
```

Note: In this case it's not necessary to specify the resolution and framerate, because there is only one mode possible.

Capture live video with audio, this works with FFplay or VLC, but not with SDL output device:

```
ffmpeg -f dshow -i video="RICOH THETA V/Z1 FullHD" -f dshow -sample_rate 48000 -sample_size 16 -channels 2 -i  
audio="Mikrofon (RICOH THETA V)" -f nut - | ffplay -fs -autoexit -
```

```
ffmpeg -f dshow -i video="RICOH THETA V/Z1 FullHD" -f dshow -sample_rate 48000 -sample_size 16 -channels 2 -i  
audio="Mikrofon (RICOH THETA V)" -b:v 10M -f mpegts - | "C:\Program Files\VideoLAN\VLC\vlc.exe" -
```

Note: In this case it's not necessary to specify the resolution and framerate, because there is only one mode possible.

PotPlayer can be used for playing a 360° stream in realtime: <https://potplayer.daum.net/> or <http://potplayer.tv/>

Open the menu, click Video --> 360° Video Mode --> tick "Enable/Disable 360° Video Mode".

The command line options are listed in About --> Command Line.

In PotPlayer you can navigate in the 360° video if "Numlock" is enabled:

Numpad 4	left
Numpad 6	right
Numpad 8	up
Numpad 2	down
Numpad 1	zoom out
Numpad 9	zoom in
Numpad -	zoom out, this does also work without Numlock
Numpad +	zoom in, this does also work without Numlock

With PotPlayer it's possible to play live 360° content from Ricoh Theta V, and it's also possible to change the viewing direction in the player in realtime:

```
ffmpeg -f dshow -i video="RICOH THETA V/Z1 FullHD" -b:v 30M -f mpegts - | "C:\Program  
Files\DAUM\PotPlayer\PotPlayerMini64.exe" -
```

```
pause
```

Same as before, but with audio:

```
ffmpeg -f dshow -i video="RICOH THETA V/Z1 FullHD" -f dshow -sample_rate 48000 -sample_size 16 -channels 2 -i  
audio="Mikrofon (RICOH THETA V)" -b:v 30M -f mpegs - | "C:\Program Files\DAUM\PotPlayer\PotPlayerMini64.exe" -  
pause
```

Another method for live streaming is described here, but it seems complicated and I haven't tested it:

http://theta360developers.github.io/community-document/live-streaming.html#_ethernet_rtsp_streaming_direct_to_computer

Note: The GoPro VR Player 3 is used for playing the live stream, but in the above link is written it works only with MAC, not with Windows.

Some notes about "Spherical sidedata":

- Use the "Spatial Media Metadata Injector" to add the required metadata to a *.mp4 video. Then it's recognized by VLC as a 360° video and you can change the viewing direction in the player.
- If such a video (with spatial metadata already injected) is re-encoded by FFmpeg, then the spatial metadata is lost:
`ffmpeg -i video_with_spatial_metadata.mp4 out.mp4`
- However if "-strict experimental" or "-strict unofficial" is used, then spatial metadata is preserved:
`ffmpeg -i video_with_spatial_metadata.mp4 -strict experimental out.mp4`
- It might be helpful also to add "-map_metadata 0":
`ffmpeg -i video_with_spatial_metadata.mp4 -map_metadata 0 -strict experimental out.mp4`
- This might be another method for adding spherical metadata, but it didn't work when I tested it:
`exiftool -XMP-GSpherical:Spherical="true" file.mp4`
- This is another method, but seems to work only for images:
`exiftool -ProjectionType="equirectangular" out.jpg`

<https://github.com/mifi/lossless-cut/issues/169>

<https://www.trekview.org/blog/2021/turn-360-photos-into-360-video/>

<https://github.com/google/spatial-media/blob/master/docs/spherical-video-rfc.md>

Amateras Dome Player: <https://www.orihalcon.co.jp/amateras/domeplayer/en/>

32.8 Ambisonics

For an introduction to Ambisonics see <https://en.wikipedia.org/wiki/Ambisonics>

This is a very good introduction to Ambisonics: https://www.researchgate.net/publication/280010078_Introduction_to_Ambisonics

This book (Zotter, Franz, Frank, Matthias: Ambisonics) is available for free download: <https://www.springer.com/de/book/9783030172060>

See also <https://community.theta360.guide/t/hi-guys-may-i-ask-two-simple-questions-about-the-audio-of-ricoh-theta-v/4705>

You can use this batch file for extracting the four R, Y, Z, and X audio channels from the *.mov file:

```
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:

```
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)

For more examples for "showwavespic" see: <https://trac.ffmpeg.org/wiki/Waveform>

32.9 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%

pause
```

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":

```
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% -f 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%
pause
```

Note: Don't forget to inject the spatial metadata with the "Spatial Media Metadata Injector".

This batch file makes an ambisonic test video with a moving sound source. A mosquito is flying in the horizontal plane around the observer:

```
set "IN=eqirectangular_test.png"    :: Eqirectangular 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"              :: Eqirectangular 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%

pause
```

Note: Don't forget to inject the spatial metadata with the "Spatial Media Metadata Injector".

32.10 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:

front	(W + X) / 2
right	(W - Y) / 2
back	(W - X) / 2
left	(W + Y) / 2

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.

```
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:

left front	$(W + 0.707 * X + 0.707 * Y) / 2$
right front	$(W + 0.707 * X - 0.707 * Y) / 2$
left back	$(W - 0.707 * X + 0.707 * Y) / 2$
right back	$(W - 0.707 * X - 0.707 * Y) / 2$

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),

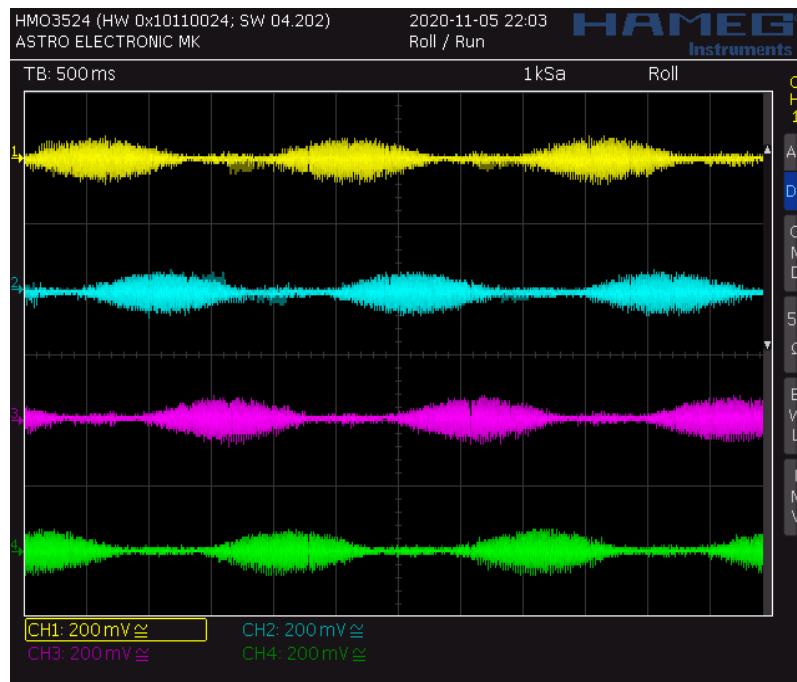
where FC and LFE are silent:

```
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 -

pause
```

This is the output of the USB 5.1 device on an oscilloscope, CH1 = FL, CH2 = FR, CH3 = BR, CH4 = BL:



This is the same as before, but with an additional subbooster channel which is using the W channel as input for the "asubboost" filter:

```
set "IN=R0010013_er.mov"    :: Equirectangular input video with ambisonic sound
set "FOV=200"                :: Field of view of fisheye projection
set "S=1080"                  :: Size of output video
set "DRY=0"                   :: asubboost: Set how much of original signal is kept. Allowed range is from 0 to 1. Default value is 0.5.
set "WET=1"                   :: asubboost: Set how much of filtered signal is kept. Allowed range is from 0 to 1. Default value is 0.8.
set "DECAY=0.7"               :: asubboost: Set delay line decay gain value. Allowed range is from 0 to 1. Default value is 0.7.
set "FEEDBACK=0.5"            :: asubboost: Set delay line feedback gain value. Allowed range is from 0 to 1. Default value is 0.5.
set "CUTOFF=100"               :: asubboost: Set cutoff frequency in Hz. Allowed range is 50 to 900. Default value is 100.
set "SLOPE=0.5"                :: asubboost: Set slope steepness in [1/octave]. Allowed range is 0.0001 to 1. Default value is 0.5.
set "DELAY=20"                 :: asubboost: Set delay in Milliseconds. Allowed range is from 1 to 100. Default value is 20.

ffmpeg -i %IN% -lavfi [0:v]v360=e:fisheye:pitch=90:h_fov=%FOV%:v_fov=%FOV%:w=%S%:h=%S%;[0:a]pan="FL+FR+FC+BL+BR|
c0=c0+0.707*c1+0.707*c3|c1=c0+0.707*c1-0.707*c3|c2=0*c0|c3=c0-0.707*c1+0.707*c3|c4=c0-0.707*c1-0.707*c3[4CH]";[0:a:0]asubboost=dry=
%DRY%:wet=%WET%:decay=%DECAY%:feedback=%FEEDBACK%:cutoff=%CUTOFF%:slope=%SLOPE%:delay=%DELAY%[LFE];[4CH]
[LFE]join=channel_layout=5.1:map="0.0-FL|0.1-FR|0.2-FC|0.3-BL|0.4-BR|1.0-LFE" -q:v 2 -f nut - | ffplay -fs -autoexit -

pause
```

Note: I didn't understand the meaning of the "decay" and "feedback" values. If you can explain it, please let me know. For this filter a block diagram would be very helpful.

Note: The unit of the "slope" parameter can also be set in dB / octave, for example -20dB.

Because it's unclear how "asubboost" actually works, I decided it's better to use instead a "lowpass" filter and enhance the volume:

```
set "IN=R0010013_er.mov"    :: Equirectangular input video with ambisonic sound
set "FOV=200"                :: Field of view of fisheye projection
set "S=1080"                  :: Size of output video
set "CUTOFF=100"               :: Subbooster cutoff frequency in Hz
set "BOOST=10"                 :: Subbooster amplification factor

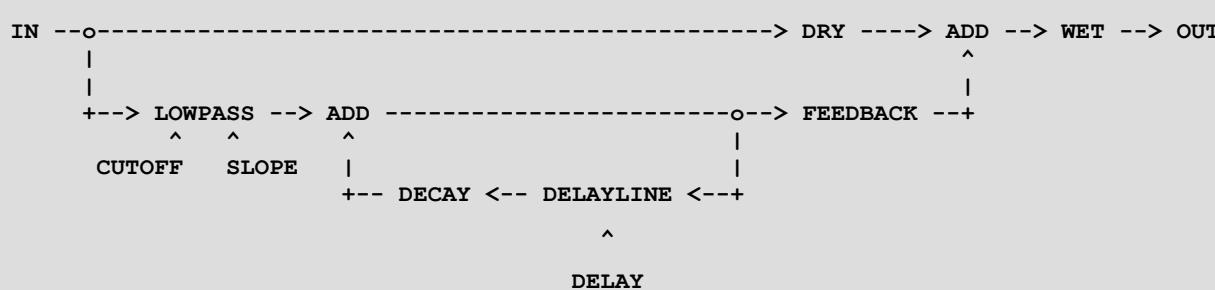
ffmpeg -i %IN% -lavfi [0:v]v360=e:fisheye:pitch=90:h_fov=%FOV%:v_fov=%FOV%:w=%S%:h=%S%;[0:a]pan="FL+FR+FC+BL+BR|
c0=c0+0.707*c1+0.707*c3|c1=c0+0.707*c1-0.707*c3|c2=0*c0|c3=c0-0.707*c1+0.707*c3|c4=c0-0.707*c1-0.707*c3[4CH]";[0:a:0]lowpass=%CUTOFF%
,volume=%BOOST%[LFE];[4CH][LFE]join=channel_layout=5.1:map="0.0-FL|0.1-FR|0.2-FC|0.3-BL|0.4-BR|1.0-LFE" -q:v 2 -f nut - | ffplay -fs
-autoexit -

pause
```

Let's try to reverse engineer the "asubboost" filter. This command makes a curve with a step at t=0.1s and shows the step response of the "asubboost" filter:

```
ffmpeg -f lavfi -i aevalsrc='0.5*gt(t,0.1)':d=1 -lavfi asplit[a][b];[b]asubboost=dry=0:wet=1:decay=0.5:feedback=0.5:delay=100[c],[a][c]join,showwaves=draw=full:s=1920x1080:r=1 -frames 1 -y out.png  
pause
```

This is the reverse engineered block diagram of the "asubboost" filter:



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:

```
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
```

For more examples for "showwavespic" see: <https://trac.ffmpeg.org/wiki/Waveform>

33 Insta360 GO 2

Help page: <https://www.insta360.com/support/supportdetail?name=go2>

It seems there exists no manual for this camera, besides the "QuickStart Guide" which leaves many questions unanswered.

Switch on the camera: Press the camera button >1s.

Switch off the camera: Press the camera button >2s.

It's not recommended to start recording with the camera button, because four different actions are possible, depending on single / double press and depending on the camera standby state. Better use the "Record" button of the remote control.

LED color:	Meaning:
Solid blue	Camera / Charge Case is powering on
Solid cyan	Camera / Charge Case (with GO 2 connected) is in standby mode
Solid red	Camera / Charge Case is off and charging
off	Camera is off and fully charged
Solid green	Charge Case is off and fully charged
Slowly flashing white	Camera / Charge Case is recording
Rapidly flashing blue	Camera / Charge Case is in U-disk mode (or storage error)
Rapidly flashing cyan	Camera / Charge Case is updating firmware
Rapidly flashing yellow	GO 2 needs to cool down (when it's on)
Rapidly flashing red	GO 2 needs to cool down (when it's off)

Video resolution in "Pro" mode: 2688x2688, about 80Mbit/s, Codec h264.

The horizontal and vertical field of view is 127° and the diagonal field of view is 150° (in "Ultrawide" mode).

The stabilization is done in postprocessing with "Insta360 Studio 2022".

The size of the stabilized file is 2560x1440 and the field of view is 108° x 64° (in "Ultrawide" mode).

Left button (with red dot) is "Record", right button is "Mode".

Press "Mode" two times, and then one time to scroll through the modes:

- Video (this is with standard stabilization)
- Photo
- Pro Video (this is with stabilization, and the horizon is always horizontal)
- Timeshift (is like timelapse, but for moving camera)
- Timelapse (for non-moving camera on a tripod)
- HDR Video (not for fast moving scenes, only 25fps)
- Slow-Mo (120fps)
- Settings

Select "Settings" and enter the menu by pressing the "Record" button. These are the menu points:

- (Microphone icon) No Wind Reduction / Wind Reduction
- ANTI FLICKER 50Hz / 60Hz / auto
- (ON/OFF icon) Timeout 80s / 120s / 180s
- START icon H.264 (nothing else can be selected here)
- (Histogram icon) Standard Bitrate / High Bitrate
- (Lamp icon) Switch the camera LED on / off
- (Camera icon) Lock or unlock? (I don't understand the purpose)
- Language English
- (WLAN icon) Auto
- Format
- (Circle with arrow icon) Reset to default settings
- i Display the firmware version
- Done

Go to "Pro Video" mode and press the "Record" button >2s to get into the menu:

- Format 16:9 or 9:16
- Resolution 1080P or 1440P
- Framerate 24, 25, 30 or 50
- Field of view Linear, Narrow, UltraWide, ActionView,
- Color Standard, Log, Vivid
- Timer 15s, 30s, 1min, 5min, 10min, 15min, 30min

See also <https://www.youtube.com/watch?v=RdwysEWAx24>

34 JJRC X17 Drone

Checklist for start:

1.	Insert SD card with FAT32	It's best to delete all old files, because the drone has no real time clock and all files are dated 1970-1-1
2.	Power on	First the drone, then the remote.
3.	Check the battery voltages	Drone: >12.0V Remote: >4.4V Drone voltage isn't shown? Put the drone flat on the ground.
4.	Gyro calibration	Push both joysticks to bottom right position.
5.	Magnetic compass calibration	Drone in normal position, press button, 3 rotations around camera axis, beep, 3 rotations around vertical axis, beep.
6.	Wait for GPS mode 2	It's better to wait 1-2 minutes longer.
7.	Adjust camera pitch	-10° is good
8.	Start video recording	Press camera button >2s
9.	Press start button, then lift off	Use the left joystick. The ▼▲ button doesn't work correctly.

The Ⓜ button works only with GPS. Short press: Return to home Long press: Headless mode

Checklist for landing:

1.	Stop video recording	If you forget it, the video file won't be readable!
2.	Catching the drone by hand	To stop the motors, turn the drone upside down.
3.	Wait 15s before switching off the drone	To make sure that the video is fully written to the SD card.

Note: The calibration procedure for the magnetic compass is described totally wrong in the manufacturer's manual. The error is up to 70°. If you follow the instructions, the drone will do "toilet bowling" and crash. Use this alternative calibration procedure instead: Hold the drone in normal position, the propellers up and the camera pointing to the horizon. Press the calibration button on the remote control. Now rotate the drone around the camera axis. That means the camera is always pointing to the same point on the horizon, and the propellers are facing up, left, down, right and so on. After 3 rotations you hear a beep. Now rotate the drone around the vertical axis. That means the propellers are always facing up, and the camera is facing north, east, south, west and so on. After 3 rotations you hear a beep. It doesn't care if you make the rotations left-hand or right-hand. That's all and you will never experience the "toilet bowling" problem again.

Properties of video from JJRC X17 drone:

Codec: h264 (Main)
Pixel format: yuv420p(progressive)
Size: 2688x1512 (= FHD * 1.4)
Bitrate: about 32000 kb/s
Framerate: about 25.26 fps
no audio stream

35 Lightning

Lightning guide for portrait photography: <https://www.digitalcameraworld.com/tutorials/cheat-sheet-pro-portrait-lighting-setups>

36 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.

Color Temperature	Illuminance [lux] @ 1m	R	G	B	S = R + G + B	R / S	G / S	B / S
3200 K	426 (Minimum)	140.4	132.5	137.3	410.2 (Minimum)	34.2% (Minimum)	32.3% (Maximum)	33.5%
3400 K	456	149.1	138.1	145.9	433.1	34.4%	31.9%	33.7% (Maximum)
3600 K	474	156.5	143.5	152.7	452.7	34.6%	31.7%	33.7% (Maximum)
3800 K	494	161.9	145.9	155.1	462.9	35.0%	31.5%	33.5%
4000 K	508	165.8	149.2	158.6	473.6	35.0%	31.5%	33.5%
4200 K	518	169.8	151.5	160.6	481.9	35.2%	31.4% (Minimum)	33.3%
4400 K	525	172.5	153.9	162.7	489.1	35.3%	31.5%	33.3%
4600 K	531	174.1	154.9	163.1	492.1	35.4%	31.5%	33.1%
4800 K	535 (Maximum)	175.9	156.5	163.3	495.7	35.5% (Maximum)	31.6%	32.9%
5000 K	535 (Maximum)	175.9	156.9	162.6	495.4	35.5%	31.7%	32.8% (Minimum)
5200 K	534	175.4	157.8	163.9	497.1 (Maximum)	35.3%	31.7%	33.0%
5400 K	530	174.5	158.0	164.5	497.0	35.1%	31.8%	33.1%
5600 K	502	169.8	156.1	164.5	490.4	34.6%	31.8%	33.5%

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

Same test for Neewer RGB660 lamp with diffusor installed, brightness is set to 100%:

Color Temperature	Illuminance [lux] @ 1m	R	G	B	$S = R + G + B$	R / S	G / S	B / S
3200 K	574							
3400 K	539 ???							
3600 K	597							
3800 K	662							
4000 K	734							
4200 K	810							
4400 K	892 (Maximum)							
4600 K	805							
4800 K	730							
5000 K	661							
5200 K	597							
5400 K	540							
5600 K	468 (Minimum)							
Hue = 0°, red	373							
Hue = 60°, yellow	684							
Hue = 120°, green	568							
Hue = 180°, cyan	799							
Hue = 240°, blue	410							
Hue = 300°, magenta	650							

Main Menu	Sub Menu	Recommended Setting	Notes
BASIC	RECORD	ON / OFF	Choose the channels you want to use
	PAN		Balance for monitoring the inputs, this doesn't affect the record
	GAIN	LOW / MID / HIGH / HI+PLUS	Choose the input gain
	INPUT	XLR/TRS	Choose the input
MONITOR	MIX		Dont't care, this is only for the monitor output
INPUT	INPUT GAIN	MIC+PHANTOM	
	LIMITER	OFF	
	LOWCUT	OFF	High pass filter
	DELAY	0	Delay time to channel 1
	PHASE	OFF	Reverses the polarity
RECORD	FILE TYPE	STEREO	One or two stereo files will be written
	FORMAT	WAV 24bit	Best quality
	SAMPLE	44.1kHz / 48kHz / 96kHz / 192kHz	Use 96kHz for ultrasound conversion
	DUAL REC	OFF or -1db to -12dB	This is only possible if channels 3 and 4 are deactivated
SLATE			Slate signal
MIC	MS MODE 1/2	OFF	
	MS MODE 3/4	OFF	
	PHANTOM VOLT	48V	Phantom voltage for Rode MT-1 microphones
OTHERS	SYSTEM --> FORMAT		Formatting the SD card
	BATTERY	NIMH / ALKAL	Battery type
	DATE / TIME		Setting date and time

I typically make either 4-channel records, or 2-channel records with DUAL REC -10dB. These settings must be changed:

Application:	BASIC --> RECORD CH3+4	RECORD --> DUAL RECORD
4-Channel Recording	ON	OFF
2-Channel Recording with DUAL REC -10dB	OFF	-10dB

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.

Menue	Recommended Setting	Notes
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

14/19 HDMI AUDIO ASSIGN	OFF	
15/19 AMBISONICS	OFF	
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.

38.1 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):

OUTPUT LEVEL / CAMERA	Output voltage (OSCILLATOR = -18dB)	Output voltage (OSCILLATOR = -20dB)	Maximum 1kHz sine output voltage, just before clipping occurs in the output signal.
0dB	7.5 mV_rms	6.2 mV_rms	62 mV_rms
12dB	30.3 mV_rms	24.0 mV_rms	240 mV_rms
20dB	79.0 mV_rms	62.0 mV_rms	620 mV_rms
30dB	249.6 mV_rms	200.0 mV_rms	2.00 V_rms
40dB	795 mV_rms	622 mV_rms	3.35 V_rms
42dB	993 mV_rms	795 mV_rms	3.35 V_rms

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):

OUTPUT LEVEL / LINE	Output voltage (OSCILLATOR = -18dB)	Output voltage (OSCILLATOR = -20dB)	Maximum 1kHz sine output voltage, just before clipping occurs in the output signal.
-12dB	62 mV_rms	49 mV_rms	0.5 V_rms
-3dB	175 mV_rms	139 mV_rms	1.41 V_rms
0dB	248 mV_rms	197 mV_rms	2.0 V_rms
12dB	990 mV_rms	785 mV_rms	3.27 V_rms

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.

Sound Rec Level Adj.	Input voltage when level indicator is at -12dB mark	Maximum sine voltage before clipping occurs	Maximum peak voltage before clipping occurs
-12dB	1050 mV_rms	4.88 V_rms	+/- 6.90 V
-6dB	525 mV_rms	2.44 V_rms	+/- 3.45 V
0dB	262 mV_rms	1.22 V_rms	+/- 1.73 V
+6dB	131 mV_rms	0.61 V_rms	+/- 0.86 V

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.

39 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/>

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>

The Apprehension Engine - First Look Part 02 (Horror Machine) <https://www.youtube.com/watch?v=61Cw5vApw-o>

The Apprehension Engine - First Look Part 03 (Horror Machine) <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>

The Apprehension Engine - Horror Suite Part 2 <https://www.youtube.com/watch?v=K9xE1UHDoLU>

Apprehension Engine: Sound check in Chicago <https://www.youtube.com/watch?v=cgp76wROgxY>

Horror Musical Instrument - The Apprehension Engine <https://www.youtube.com/watch?v=lzk-l8Gm0MY>

DIY Apprehension Engine 1 - Metallic Bones (Rulers) https://www.youtube.com/watch?v=_q-yMKs1NYg

DIY Apprehension Engine 2 - Glass Shards (Wine Glass) <https://www.youtube.com/watch?v=1arnzEoAuAk>

DIY Apprehension Engine 3 - Gates of Hell (Spring Reverb) <https://www.youtube.com/watch?v=vV7ygTQu1Eo>

DIY Apprehension Engine 4 - Heavy Metal (Guitar) <https://www.youtube.com/watch?v=gh7TbusFy-4>

Latest build: "Horror Box 1.0" - (demo) - spring box w/ piezo microphone <https://www.youtube.com/watch?v=tEgUXJiDEIg>

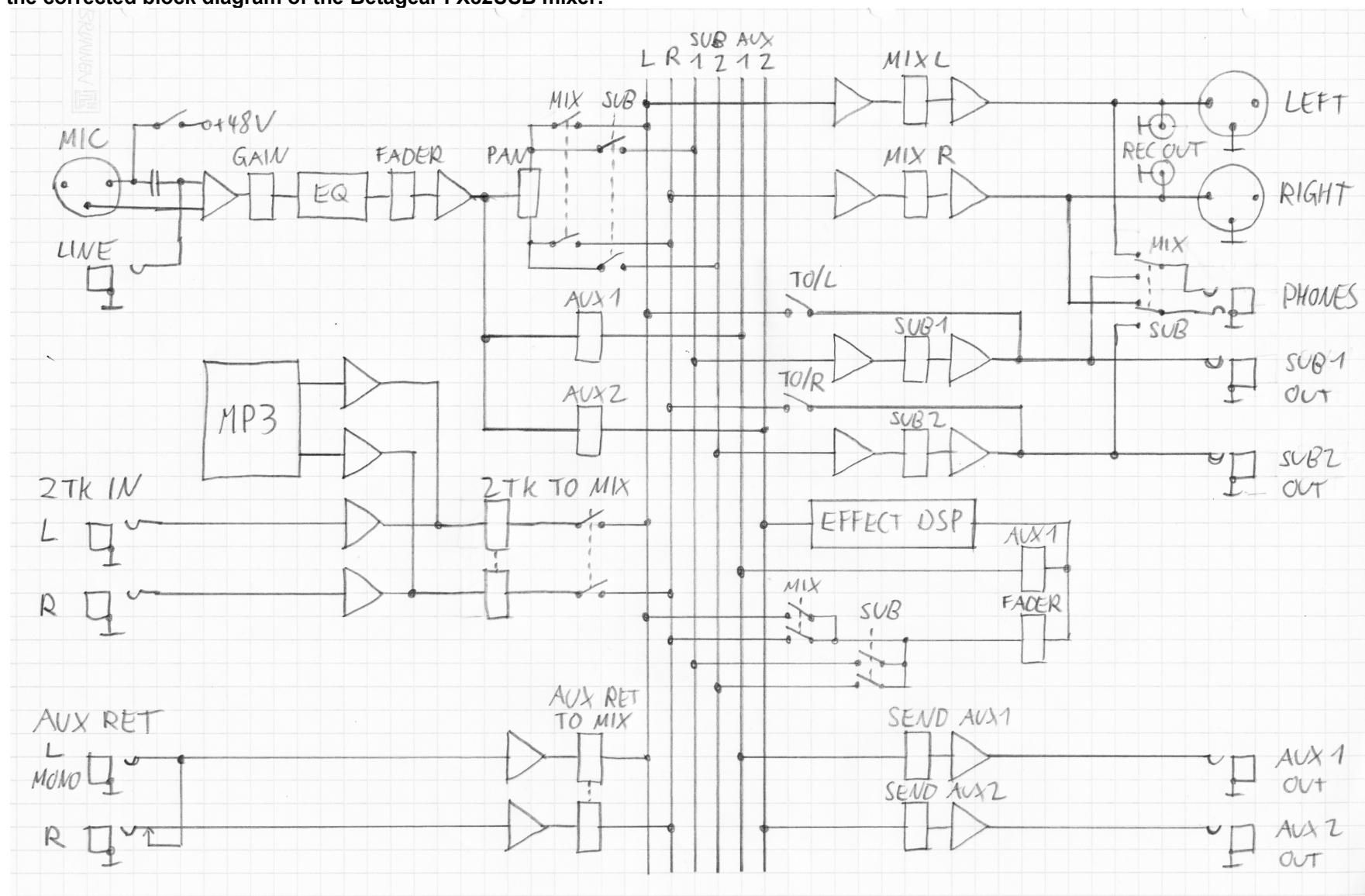
Here is a series of "How to Build The Apprehension Engine" videos by Michael Freudenberg on Youtube:

Chapter	Youtube Link	Material needed
#1 - HISTORY	https://www.youtube.com/watch?v=xHXUEycuAMY	
#2 - The Base	https://www.youtube.com/watch?v=uwZmU4l4P10	5 Ply board: (for the base) Width 40cm x Length 1.2 meters Pine wood: Width 4.2 cm x Depth 1.9 cm x Length 1.2 meters
#3 - The Sides	https://www.youtube.com/watch?v=eZXM-Dj7IQw	
#4 - Finishing the Sides	https://www.youtube.com/watch?v=JfEjMSTJ_Ts	
#5 - Attaching rear support	https://www.youtube.com/watch?v=SkLuiXLvTgw	
#6 - The Hurdy Gurdy Wheel	https://www.youtube.com/watch?v=cLI_YBDax5s	MDF 16mm thick
#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=XWddPilneco	
#10 - Guitar Neck Supports	https://www.youtube.com/watch?v=az5uFla6gBg	
#11 - Left side Soundboard	https://www.youtube.com/watch?v=HhuUxrRji5E	
#12 - Front and Right side Soundboard	https://www.youtube.com/watch?v=wIzUkMgLDhM	
#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=Kel4onBniTs	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-Pickup-HUMBUCKER-3-way-switch-Electric-Guitar/332103308294
#22 - Installing the Piezo Contact Pickup	https://www.youtube.com/watch?v=N1BU6MSp8Xs	

#23 - Installing the Reverb Tank	https://www.youtube.com/watch?v=keRG7eUaOww	Contact Piezo Pickups with volume and balance knobs, Reverb Tank (Long 2 Spring Long Decay Reverb tank model 4FB3A1A). Input 1475 Ω, Output 2250 Ω, Long decay 2.75s - 4s, Input grounded, Output grounded, \$23.50 USD https://www.amplifiedparts.com/products/reverb-tank-mod-4fb3a1a-long-decay-2-spring
#24 - The Final Tutorial	https://www.youtube.com/watch?v=QSJOtxwgd8Y	
Learn how to add the cotton to the Hurdy Gurdy strings here:	https://www.youtube.com/watch?v=0TTi5FoNKw8	Also it's important to apply rosin to the hurdy gurdy wheel.

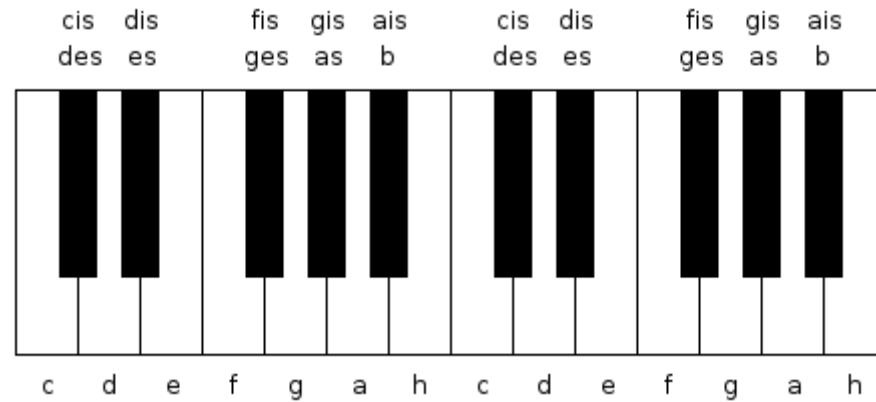
This is the corrected block diagram of the Betagear FX82USB mixer:



40 Synthesizers and Midi

See also: "Build an analog POLYPHONIC SYNTHESIZER from scratch": <https://www.youtube.com/watch?v=vj-DSh6yfM0>

40.1 The keyboard



40.2 Frequencies and control voltages of the keys

Frequency factor between two notes: $2^{(1/12)} = 1.05946$

Octave	c	cis/des	d	dis/es	e	f	fis/ges	g	gis/as	a	ais/b	h
	C	CSharp	D	DSharp	E	F	FSharp	G	GSharp	A	ASharp	B
-2	4.0879 Hz	4.3310 Hz	4.5885 Hz	4.8614 Hz	5.1504 Hz	5.4567 Hz	5.7812 Hz	6.1249 Hz	6.4891 Hz	6.8750 Hz	7.2838 Hz	7.7169 Hz
-1	8.1768 Hz	8.6620 Hz	9.1770 Hz	9.7227 Hz	10.301 Hz	10.913 Hz	11.562 Hz	12.250 Hz	12.978 Hz	13.750 Hz	14.568 Hz	15.434 Hz
0	16.352 Hz 0.0 V	17.324 Hz 0.083 V	18.354 Hz 0.167 V	19.445 Hz 0.25 V	20.602 Hz 0.333 V	21.827 Hz 0.417 V	23.125 Hz 0.5 V	24.500 Hz 0.583 V	25.957 Hz 0.667 V	27.500 Hz 0.75 V	29.135 Hz 0.833 V	30.868 Hz 0.917 V
1	32.703 Hz 1.00 V	34.648 Hz 1.083 V	36.708 Hz 1.167 V	38.891 Hz 1.25 V	41.203 Hz 1.333 V	43.654 Hz 1.417 V	46.249 Hz 1.5 V	48.999 Hz 1.583 V	51.913 Hz 1.667 V	55.000 Hz 1.75 V	58.270 Hz 1.833 V	61.735 Hz 1.917 V
2	65.406 Hz 2.00 V	69.296 Hz 2.083 V	73.416 Hz 2.167 V	77.782 Hz 2.25 V	82.407 Hz 2.333 V	87.307 Hz 2.417 V	92.499 Hz 2.5 V	97.999 Hz 2.583 V	103.83 Hz 2.667 V	110.00 Hz 2.75 V	116.54 Hz 2.833 V	123.47 Hz 2.917 V
3	130.81 Hz 3.00 V	138.59 Hz 3.083 V	146.83 Hz 3.167 V	155.56 Hz 3.25 V	164.81 Hz 3.333 V	174.61 Hz 3.417 V	185.00 Hz 3.5 V	196.00 Hz 3.583 V	207.65 Hz 3.667 V	220.00 Hz 3.75 V	233.08 Hz 3.833 V	246.94 Hz 3.917 V
4	261.63 Hz 4.00 V	277.18 Hz 4.083 V	293.66 Hz 4.167 V	311.13 Hz 4.25 V	329.63 Hz 4.333 V	349.23 Hz 4.417 V	369.99 Hz 4.5 V	392.00 Hz 4.583 V	415.30 Hz 4.667 V	440.00 Hz 4.75 V	466.14 Hz 4.833 V	493.88 Hz 4.917 V
5	523.25 Hz 5.00 V	554.37 Hz 5.083 V	587.33 Hz 5.167 V	622.25 Hz 5.25 V	659.26 Hz 5.333 V	698.46 Hz 5.417 V	739.99 Hz 5.5 V	783.99 Hz 5.583 V	830.61 Hz 5.667 V	880.00 Hz 5.75 V	932.33 Hz 5.833 V	987.77 Hz 5.917 V
6	1046.5 Hz 6.00 V	1108.7 Hz 6.083 V	1174.7 Hz 6.167 V	1244.5 Hz 6.25 V	1318.5 Hz 6.333 V	1396.9 Hz 6.417 V	1480.0 Hz 6.5 V	1568.0 Hz 6.583 V	1661.2 Hz 6.667 V	1760.0 Hz 6.75 V	1864.7 Hz 6.833 V	1975.5 Hz 6.917 V
7	2093.0 Hz 7.00 V	2217.5 Hz 7.083 V	2349.3 Hz 7.167 V	2489.0 Hz 7.25 V	2637.0 Hz 7.333 V	2793.8 Hz 7.417 V	2960.0 Hz 7.5 V	3136.0 Hz 7.583 V	3322.4 Hz 7.667 V	3520.0 Hz 7.75 V	3727.3 Hz 7.833 V	3951.1 Hz 7.917 V
8	4186.0 Hz 8.00 V	4434.9 Hz	4698.6 Hz	4978.0 Hz	5274.0 Hz	5587.7 Hz	5919.9 Hz	6271.9 Hz	6644.9 Hz	7040.0 Hz	7458.6 Hz	7902.1 Hz
9	8372.0 Hz	8869.8 Hz	9397.3 Hz	9956.1 Hz	10548 Hz	11175 Hz	11840 Hz	12544 Hz	13290 Hz	14080 Hz	14917 Hz	15804 Hz
10	16744 Hz											

Matrixbrute: 4.0879 Hz – 16744 Hz (including +2 octaves with the coarse tuning knob)

Subsequent 37: 8.1768 Hz – 8372.0 Hz

Microbrute: 32.703 Hz – 2093.0 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:

$$m = 12 \cdot \log_2(f_m/440 \text{ Hz}) + 69 \quad f_m = 440 \text{ Hz} \cdot 2^{(m-69)/12} \quad \log_2(x) = \ln(x) / \ln(2)$$

Matrixbrute control voltage:

$$U = 4.75 + (m - 69) / 12 = (m / 12) - 1.0 \quad U = 4.75 + \log_2(f / 440 \text{ Hz}) \quad f = 440 \text{ Hz} \cdot 2^{(U - 4.75)}$$

The voltage step from one to the next semitone is $1/12 \text{ V} = 0.0833 \text{ V}$.

40.3 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.

40.4 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.

40.5 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

40.6 Presets / Patches

	Matrixbrute	Subsequent 37
Initialize a preset with a basic sound	Hold PANEL and then press PRESET.	
Load a preset	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.

40.7 Potentiometer behaviour

Behaviour	Matrixbrute	Subsequent 37	Notes
Absolute mode, the value changes immediately when you turn the potentiometer.	In MIDI Control Center, device tab: Set 2pot mode2 to "Jump". On MatrixBrute: Press PRESET+SEQ+MOD, then press G1.	Press "GLOBAL", scroll down to "POT MODE", then select "ABS".	Recommended for creating new sounds.

Value changes, when you turn the potentiometer past the current setting.	In MIDI Control Center, device tab: Set 2pot mode2 to “Hook”. On MatrixBrute: Press PRESET+SEQ+MOD, then press G2.	Press “GLOBAL”, scroll down to “POT MODE”, then select “THRU”.	Recommended for live performance.
Value changes relative to potentiometer adjustment. x Then you must first turn it to the other extreme position.	In MIDI Control Center, device tab: Set 2pot mode2 to “Scaled”. On MatrixBrute: Press PRESET+SEQ+MOD, then press G3. Drawback: If the potentiometer is at the extreme position, you cannot move it further.	Press “GLOBAL”, scroll down to “POT MODE”, then select “RLTV”.	Recommended for live performance.

40.8 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.

40.9 MatrixBrute power saving mode

In the system settings editor (PRESET + SEQ + MOD), row “P”, column “4” is “OFF” and column “5” is “ON”.

40.10 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.

40.11 MatrixBrute Control Voltages I/O

VCO1 Pitch	0 - 10V
------------	---------

VCO1 Ultra Saw	+ - 5V
VCO1 Pulse Width	+ - 5V
VCO1 Metalizer	+ - 5V
VCO2 Pitch	0 - 10V
VCO2 Ultra Saw	+ - 5V
VCO2 Pulse Width	+ - 5V
VCO2 Metalizer	+ - 5V
Steiner Cutoff	0 - 10V
Ladder Cutoff	0 - 10V
LFO1 Amount	0 - 10V
VCA	0 - 10V

40.12 MatrixBrute: Which Modules are unused?

Module ...	is unused, if ...
VCO1	([VCO1>VCO2] == 0 and ([MIXER_VCO1] == 0 or [MIXER_VCO1_Filter] == none)) or ([VCO1_Sub] == 0 and [VCO1_Saw] == 0 and [VCO1_Square] == 0 and [VCO1_Tri] == 0)
VCO2	[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)
VCO3-LFO3	[VCO1<VCO3>VCO2] == 0 and [VCF1<VCO3>VCF2] == 0 and (Row_G has no active elements) and ([MIXER_VCO3] == 0 or [MIXER_VCO3_Filter] == none)
NOISE	[VCO1<Noise>VCF1] == 0 and ([MIXER_Noise] == 0 or [MIXER_Noise_Filter] == none)
VCF1_STEINER	[Steiner_Out] == 0
VCF2_LADDER	[Ladder_Out] == 0
LFO1	Row_E has no active elements
LFO2	Row_F has no active elements
ENV1	[VCF1_STEINER_Env_Amt] == 0 and [VCF2_LADDER_Env_Amt] == 0 and (Row_A has no active elements)
ENV2	(ENV2 is always used)
ENV3	Row_C has no active elements
ANALOG_EFFECTS	[Dry/Wet] == 0

41 Timelapse duration table

This table lists the number of pictures and the video duration at 30fps, depending on interval and recording time.

Interval	Recording time in hours																	
	1h		2h		3h		4h		5h		6h		8h		12h		24h	
2s	1800	60s	3600	120s	5400	150s	7200	240s	9400	300s	10800	360s	14400	480s	21600	720s	43200	1440s
3s	1200	40s	2400	80s	3600	120s	4800	160s	6000	200s	7200	240s	9600	320s	14400	480s	28800	960s
4s	900	30s	1800	60s	2700	90s	3600	120s	4500	150s	5400	180s	7200	240s	10800	360s	21600	720s
5s	720	24s	1440	48s	2160	72s	2880	96s	3600	120s	4320	144s	5760	192s	8640	288s	17280	576s
6s	600	20s	1200	40s	1800	60s	2400	80s	3000	100s	3600	120s	4800	160s	7200	240s	14400	480s
8s	450	15s	900	30s	1350	45s	1800	60s	2350	75s	2700	90s	3600	120s	5400	180s	10800	360s
10s	360	12s	720	24s	1080	36s	1440	48s	1800	60s	2160	72s	2880	96s	4320	144s	8640	288s
12s	300	10s	600	20s	900	30s	1200	40s	1500	50s	1800	60s	2400	80s	3600	120s	7200	240s
15s	240	8s	480	16s	720	24s	960	32s	1200	40s	1440	48s	1920	64s	2880	96s	5760	192s
20s	180	6s	360	12s	540	18s	720	24s	900	30s	1080	36s	1440	48s	2160	72s	4320	144s
24s	150	5s	300	10s	450	15s	600	20s	750	25s	900	30s	1200	40s	1800	60s	3600	120s
30s	120	4s	240	8s	360	12s	480	16s	600	20s	720	24s	960	32s	1440	48s	2880	96s
40s	90	3s	180	6s	270	9s	360	12s	450	15s	540	18s	720	24s	1080	36s	2160	72s
60s	60	2s	120	4s	180	6s	240	8s	300	10s	360	12s	480	16s	720	24s	1440	48s
120s	30	1s	60	2s	90	3s	120	4s	150	5s	180	6s	240	8s	360	12s	720	24s

How to re-format SD cards which have multiple partitions (this happens for example if the card comes from a Raspberry Pi) ?

Windows 10: Right-click on the Windows symbol in the lower left of the desktop, and choose "Datenträgerverwaltung" (If you know how it's called in english, please let me know). Find the drive in the list, right-click and format.

42 Zhiyun Crane 3S

See also: <https://www.zhiyun-tech.com/crane3s/en>

Mode	Pan axis (left / right)	Tilt axis (up / down)	Roll axis (rotate image)	How to activate this mode	Joystick	Description
PF	following	stabilized	stabilized	Press MODE button	Pan + Tilt	Pan following mode (This is the default mode after startup) Note: The "MODE" button is only on the main body.
L	stabilized	stabilized	stabilized	Press MODE button	Pan + Tilt	Locking mode Note: The "MODE" button is only on the main body.
POV	following	following	following	Press POV button	--	Point of view mode, press FOV button again to return to previous mode
V	following	following	following	Press POV button twice	Pan	I don't understand this mode. If you can explain it, please let me know.
F	following	following	stabilized	Press F button	Roll	Following mode, press F button again to return to previous mode
GO	fast following	fast following	stabilized	Press GO button	--	Following mode, similar to "F" mode but faster following. Press GO button again to return to previous mode

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 stbilization in the lens is activated.

Balance adjustment table:

Camera + Lens	Camera left/right	Tilt axis	Roll axis	Pan axis
GH5S + Leica DG 12-60mm f/2.8-4.0	22mm	48mm (maximum)	27mm	54mm
GH5S + Canon CN-E 24mm T1.5 L F with 0.64x SpeedBooster and variable ND filter and focus motor	22mm	27mm	8.5mm	43mm

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.

43 Timelapse+ View

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:

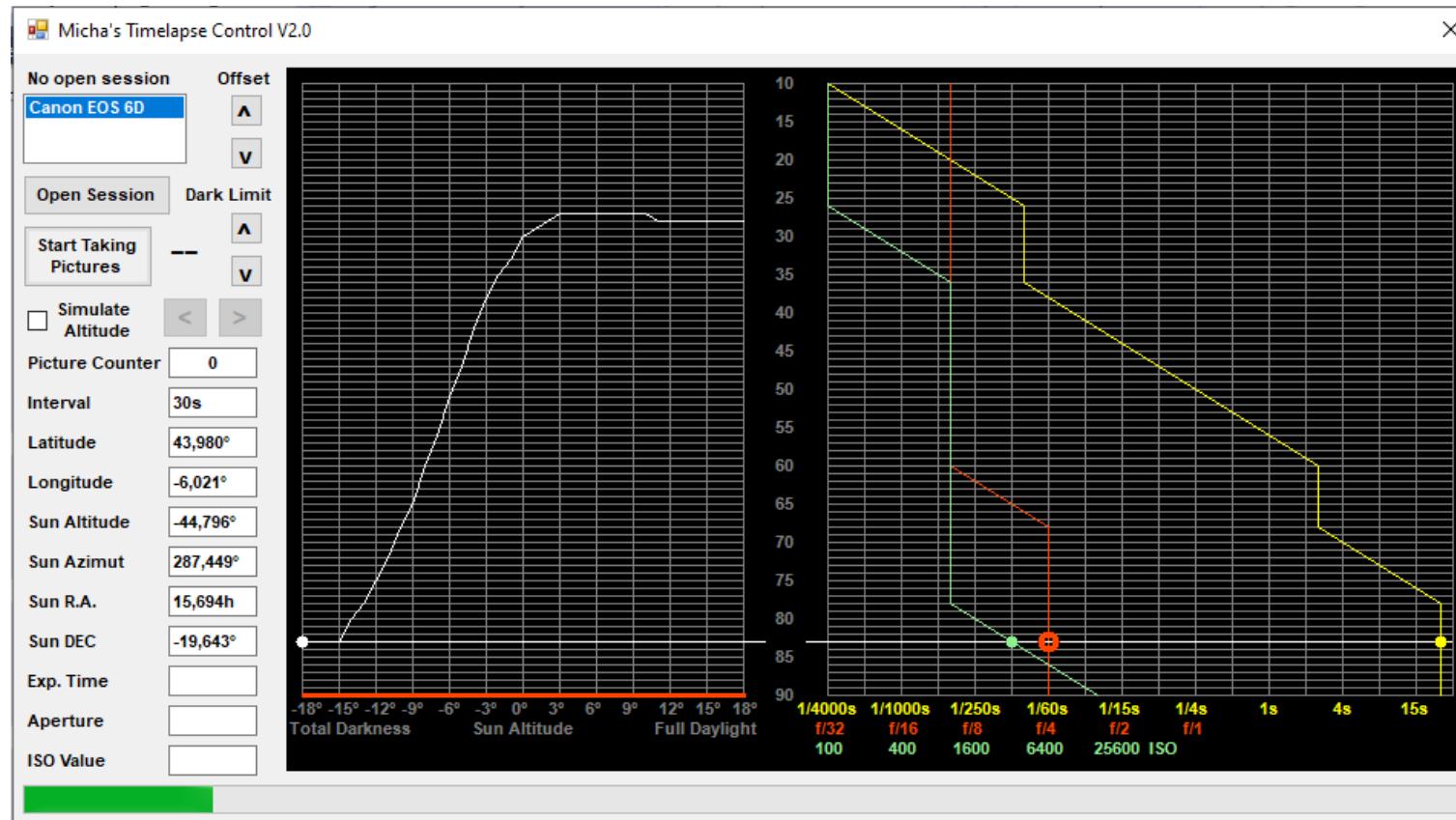
- Set the camera to manual (M) mode
- Use a native ISO setting (not Auto ISO)
- Save as RAW (not RAW + JPG)
- Manual focus (no autofocus)
- Disable image stabilization
- Check all parameters before using
- Don't rely on the internal battery, use an external powerbank
- Save the images in the camera, 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).

Example for HDR timelapse with Timelapse+ View: <https://www.facebook.com/groups/395686464095972/permalink/1413928625605079/>

44 Timelapse_Control_V2

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 "."
43.98021    Geographic latitude, north = positive, decimal separator "."
83           Light_value for altitude -18° (deep night)
83           Light_value for altitude -15°
75           Light_value for altitude -12°
65           Light_value for altitude -9°
51           Light_value for altitude -6°
38           Light_value for altitude -3°
30           Light_value for altitude 0° (sunset)
27           Light_value for altitude +3°
27           Light_value for altitude +6°
27           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
42 22 10 10
43 23 10 10
44 24 10 10
45 25 10 10
46 26 10 10
47 27 10 10
48 28 10 10
49 29 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
84 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
```

```
==== Anything below this line are only comments ====
```

```
-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:
```

0	1/4000s	1	1/3200s	2	1/2500s	3	1/2000s	4	1/1600s	5	1/1250s
6	1/1000s	7	1/800s	8	1/640s	9	1/500s	10	1/400s	11	1/320s
12	1/250s	13	1/200s	14	1/160s	15	1/125s	16	1/100s	17	1/80s
18	1/60s	19	1/50s	20	1/40s	21	1/30s	22	1/25s	23	1/20s
24	1/15s	25	1/13s	26	1/10s	27	1/8s	28	1/6s	29	1/5s
30	1/4s	31	0.3s	32	0.4s	33	0.5s	34	0.6s	35	0.8s
36	1s	37	1.3s	38	1.6s	39	2s	40	2.5s	41	3.2s
42	4s	43	5s	44	6s	45	8s	46	10s	47	13s
48	15s	49	20s	50	25s	51	30s				

```
Aperture codes:
```

0	f/32	1	f/29	2	f/25	3	f/22	4	f/20	5	f/18
6	f/16	7	f/14	8	f/13	9	f/11	10	f/10	11	f/9
12	f/8	13	f/7.1	14	f/6.3	15	f/5.6	16	f/5	17	f/4.5
18	f/4	19	f/3.5	20	f/3.2	21	f/2.8	22	f/2.5	22	f/2.2
24	f/2	25	1.8	26	f/1.6	27	f/1.4	28	f/1.2	29	f/1.1
30	f/1										

```
ISO codes:
```

0	ISO100	1	ISO125	2	ISO160	3	ISO200	4	ISO250	5	ISO320
6	ISO400	7	ISO500	8	ISO640	9	ISO800	10	ISO1000	11	ISO1250
12	ISO1600	13	ISO2000	14	ISO2500	15	ISO3200	16	ISO4000	17	ISO5000
18	ISO6400	19	ISO8000	20	ISO10000	21	ISO12800	22	ISO16000	23	ISO20000
24	ISO25600										

My source code can be downloaded here: http://www.astro-electronic.de/source/Timelapse_Control_V2.zip

Hint: You must allow unsafe code in the project: Click on project → properties → build → allow unsafe code

The following libraries from Canon are required: EdsImage.dll, EDSDK.dll

45 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/>

The manual is available here: https://www.projectpluto.com/user_man/manual.pdf

45.1 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.

45.2 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.

45.3 Add new comets to Guide 9.1

The "Add MPC Comets / Asteroids" function does no longer work. You can use this workaround:

Go to <http://astro.vanbuitenen.nl/cometelements?format=guide>

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)

```
The ^Minor Planet Center (MPC) //xhttps://www.minorplanetcenter.net/cfa/ps/mpc.html^ and the
^IMCCE//xhttp://www.imcce.fr/fr^
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.
```

```
^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".

45.4 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:

2013 02 15 190000 12 01 19.4 -31 18 25 0.00026 0.988 127.0 52.9 8.4 2279.58 003.5 285 -29 -23 0.31 147 +36 44 359.0 / Map / Offsets
2013 02 15 190100 12 01 30.2 -30 40 19 0.00026 0.988 127.5 52.5 8.3 2300.72 003.5 284 -29 -24 0.31 147 +35 44 359.3 / Map / Offsets
2013 02 15 190200 12 01 41.1 -30 01 52 0.00025 0.988 128.0 52.0 8.3 2321.73 003.5 284 -28 -24 0.31 147 +35 44 359.5 / Map / Offsets
2013 02 15 190300 12 01 51.9 -29 23 04 0.00025 0.988 128.5 51.5 8.3 2342.59 003.5 283 -28 -24 0.31 148 +35 44 359.8 / Map / Offsets
2013 02 15 190400 12 02 02.7 -28 43 55 0.00025 0.988 128.9 51.1 8.3 2363.24 003.4 283 -27 -24 0.31 148 +35 44 000.1 / Map / Offsets
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.

45.5 Add an overlay with comments

- Click on "Overlays" and select "Edit Overlay".
- Click on "(Create new overlay)", then click "ok".
- Enter a name for the new overlay, for example "Comments".
- A small new window appears, where you can select "Add Text".
- Right-click somewhere in the star chart and enter your comment.
- The data is saved in a *.uov file in the Guide folder, but the file isn't easily editable.

45.6 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.

The longitude can be found here, for example:

<https://skyandtelescope.org/observing/interactive-sky-watching-tools/transit-times-of-jupiters-great-red-spot/>

45.7 Add a user-defined location

This can be added in the file `user_loc.txt`

45.8 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:

```
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=iw:ih/2:y=0,vflip"
-y %OUT%

pause
```

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
110 20
120 15
130 6
140 10
150 6
160 6
170 6
180 7
190 8
200 8
210 7
220 7
230 9
240 9
250 8
260 14
270 16
280 8
290 12
300 17
310 21
320 24
330 26
340 28
350 29
360 25
hend

i N_for_North 0 .5 .5
i N_for_North 43 .5 .4
i E_for_East 47 .5 .4
i E_for_East 90 .5 .5
i S_for_South 133 .5 .4
i E_for_East 137 .5 .4
```

```
i S_for_South 180 .5 .5  
i S_for_South 223 .5 .4  
i W_for_West 227 .5 .4  
i W_for_West 270 .5 .5  
i N_for_North 313 .5 .4  
i W_for_West 317 .5 .4
```

45.9 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:

```
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.

45.10 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.

45.11 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.

45.12 Find objects from Tycho catalog

For example, if you want to find the star "TYC 5965 965" from the Tycho catalog, use GoTo --> Star --> Guide Star catalog --> 5965 965

45.13 Moon libration

In Tables --> Lunar Data you can generate tables which contains the moon's libration.

"AMT" is the amount of libration in degrees, and "PA" is the position angle, counter-clockwise from the north pole.

Libration in latitude:

- A positive value means the north pole is visible.
- A negative value means the south pole is visible.

Libration in longitude:

- A positive value means more details are visible at the left side (longitude west), that's the side where Mare Procellarum is.
- A negative value means more details are visible at the right side (longitude east), that's the side where Mare Crisium is.

46 Stellarium

www.stellarium.org

How to create equirectangular images of the night sky:

<http://paulbourke.net/dome/stellariumsphere/>

The procedure for running Stellarium on the extended desktop (4K beamer) is as follows:

- Start Stellarium. If it's in fullscreen mode, press F11 to make it a window.
- Move this window to the extended desktop (4K beamer), then press F11 to make it fullscreen again.
- Press F2 to open the configuration window. Click on "Save Settings" (which is in the middle of the screen). **It's not possible to open the configuration window with the mouse, because the lower left corner of the screen isn't visible in the dome).**

That's all and Stellarium does now behave as follows:

- If the 4K beamer is not connected, it will start fullscreen on the default FHD screen.
- If the 4K beamer is connected, it will start on the extended desktop.
- If the beamer is disconnected while Stellarium is running, it does automatically move fullscreen to the default FHD screen.
- If the beamer is connected while Stellarium is running, it doesn't automatically move to the extended desktop. You have to restart Stellarium.

See also:

<https://github.com/Stellarium/stellarium/wiki/Common-Problems-for-the-current-version#mixing-screens-or-projectors-with-high-and-normal-resolutions>

See also: <http://www.astro-electronic.de/planetarium.htm>

This is the path to the config.ini file: C:\Users\user_name\AppData\Roaming\Stellarium\config.ini

Useful parameters in the configuration file for fulldome projection:

Section	Parameter	Meaning
[gui]	<code>flag_mouse_cursor_timeout = true</code>	Auto-hide the mouse pointer ...
	<code>mouse_cursor_timeout = 3</code>	... after 3 seconds
[navigation]	<code>flag_enable_mouse_navigation = false</code>	Mouse navigation makes no sense in a planetarium
	<code>flag_enable_mouse_zooming = false</code>	Mouse zooming makes no sense in a planetarium
	<code>init_fov = 180</code>	Field of view 180°
	<code>init_view_pos = 0,0,1</code>	Zenit
	<code>max_fov = 180</code>	Maximum field of view 180°
[projection]	<code>type = ProjectionFisheye</code>	Fulldome projection
	<code>viewport = disc</code>	Don't show anything below the horizon
[video]	<code>fullscreen = true</code>	Fullscreen mode
	<code>screen_number = 1</code>	Screen number, typically 0 for the default screen and 1 for the extended desktop (beamer)
	<code>screen_w</code>	Screen width, don't care if <code>fullscreen = true</code>
	<code>screen_h</code>	Screen height, don't care if <code>fullscreen = true</code>
	<code>screen_x = 0</code>	X position of top left corner
	<code>screen_y = 0</code>	Y position of top left corner

Useful settings for fulldome projection (if these settings aren't set in the configuration file):

- Configuration --> Tools --> Disc viewport
- Configuration --> Tools --> Gravity labels
- Configuration --> Tools --> Untick "Enable mouse navigation"
- Configuration --> Tools --> Untick "Enable mouse zooming"
- Sky and viewing options --> Projection --> Fish-eye
- Sky and viewing options --> Custom FoV limit --> 180°

47 Space Engine

This is a physically correct simulation program for astronomy and space, covering the whole universe. It was developed by Vladimir Romanyuk. The known part of the universe is simulated with real data, and the unknown part is filled with procedural galaxies, stars and planets.

<http://spaceengine.org/>

- The latest version costs 20.99 EUR. A Nvidia GPU is required. Older versions are free, but don't work with newer graphics cards.
- There is a "SpaceEngine Pro" version that costs additional 52.99 EUR, but it doesn't have any features that I need.
- It's surprising that such a good program has such a bad documentation.
- The informations on the website are several years old and parts of it are outdated. For example, it's written there "Look at the readme_eng.txt file located in the SpaceEngine\docs\ folder, it contains a list of all controls in the program." But this folder doesn't exist. The folder was renamed "license" and still contains the mentioned file. But the informations in this file is also outdated.

Another example: On the website is written "Alternately, you can open the in-game controls menu by the [F8] key to view and modify the controls." This doesn't work either.

- Click in the left menu in "Settings" and then on "Controls" to get a list of all keyboard shortcuts. Unfortunately this list can't be printed out.
- There are some tutorials in Planetarium --> Tutorials, but they cover only a small subset of all functions.
- The resolution of a saved video is the same as the resolution of the desktop. My notebook has a 1920x1080 screen, but the resolution is doubled to 3840x2160 if a 4K beamer is connected. Then the video is saved with 3840x2160 resolution. It's not necessary to power on the beamer, connecting it with a HDMI cable is sufficient.
- In window mode, it's possible to make the window larger than the desktop. The window can be moved with Alt key and left mouse button.
- What's called "Cylindrical" in the display menu is in fact an equirectangular projection. When you read "cylindrical" in SE's documentation, you can always translate it to "equirectangular".

47.1 Keyboard shortcuts

Common controls:

Main Menu	Esc	
Video capture settings	F9	
Video capture	Ctrl-F9	
Screenshot without GUI	F11	
Screenshot with GUI	Ctrl-F11	
Console	~	Ö on german keyboard Type "FPS" to enable the framerate counter. Type "TARDIS" to get the same time warp settings as the planetarium mode. Type "FOV 90" to set the field of view to 90°.
Switch to fullscreen	Alt-Enter	
Switch to next display	Shift-Enter	
Minimize window	Ctrl-Enter	
Stereoscopic 3D	Numpad /	
Reset pose in VR	Enter	
Exit to desktop	Alt-F4	
Orbit Object	Insert	This key is not assigned by default. I did assign the "Insert" key to this function. When you hold the "Insert" key pressed, you can orbit around the object with the Numpad 4, Numpad 6, Numpad 2, and Numpad 8 keys.
Journey log	Tab	
Universe map (selection)	F1 or M	
Universe map (current position)	Ctrl-F1 or Ctrl-M	
Solar system chart	F8 or 0	
Solar system browser (selection root)	Alt-F2	

Solar system browser (selection level)	F2	
Solar system browser (current position)	Ctrl-F2	
Find object	F3	
Star browser	Shift-F3	
Spacecraft manager	Ctrl-F3	
Locations	F6	
Wiki	I	
Settings	F4	
Visual style	Ctrl-Shift-F4	
Filter objects	Shift-F4	
Graphics	Ctrl-F4	
Sound	Ctrl-F12	
Music player	Shift-F12	
Editor	Shift-F2	
Debug mode	Numpad * or Shift-8	
Select previous object	Backspace	
Select home object	Shift-H	
Free binding to object	Shift-D	
Follow object	Shift-F	
Rotate with object	Shift-R	
Reverse time speed	J	
Decelerate time speed	K	
Accelerate time speed	L	
Pause time	Spacebar	
Set normal time speed	\	^ on german keyboard
Set current time	Ctrl-\	Ctrl-^ on german keyboard

Frame timing mode	Shift-\	Shift-^ or ° on german keyboard
Pause scenario	Shift-Spacebar	
Cancel scenario	Shift-Esc	
Zoom out	Page Down	
Zoom in	Page Up	
Reset zoom	Home	
Free mode	1	
Spacecraft mode	2	
Aircraft mode	3	
Take control of spacecraft	4	
Toggle auto horizon	5	
Atmospheres	Shift-A	
Aurora	Ctrl-Shift-A	
Clouds	Shift-C	
Water	Ctrl-C	
Comet tails	Ctrl-Shift-C	
Orbits	O	
Vectors	Ctrl-O	
Selection pointer	:	Ü on german keyboard
Velocity vector	Shift-:	Shift-Ü on german keyboard
Celestial grids	Ctrl-:	Ctrl-Ü on german keyboard
Geographic grid	Alt-:	Alt-Ü on german keyboard
Labels	'	Ä on german keyboard
Markers	Ctrl-'	Ctrl-Ä on german keyboard
Constellations	X	
Constellation lines	Shift-X	

Constellation boundaries	Ctrl-X	
Constellation names	Alt-X	
Interface (Toggle text on / off)	Ctrl-~	Ctrl-Ö on german keyboard
Diffraction spikes	Shift-B	
Photo mode (Toggle autoexposure)	V	
Planet shine	Shift-L	
Eclipse shadows	Ctrl-L	
Reset magnitude limit	P	
Increase magnitude limit]	' on german keyboard
Decrease magnitude limit	[ß on german keyboard
Increase galaxies mag limit	Shift-]	Shift-' on german keyboard
Decrease galaxies mag limit	Shift-[Shift-ß on german keyboard
Increase stars mag limit	Ctrl-]	Ctrl-' on german keyboard
Decrease stars mag limit	Ctrl-[Ctrl-ß on german keyboard
Increase planets mag limit	Ctrl-Shift-]	Ctrl-Shift-' on german keyboard
Decrease planets mag limit	Ctrl-Shift-[Ctrl-Shift-ß on german keyboard
Increase exposure compensation	.	
Decrease exposure compensation	,	
Reset exposure compensation	?	# on german keyboard
Increase ambient light	Shift-.	
Decrease ambient light	Shift-,	
Increase galaxies brightness	Ctrl-.	
Decrease galaxies brightness	Ctrl-,	
Debug screen mode	Shift-Numpad *	(or Numpad-* on on german keyboard
UpdateStdShaders	F5	
UpdateProcShaders	Ctrl-F5	

UpdateUnivTree	Shift-F5	
ReloadObject	Ctrl-Shift-F5	
Cullface	Shift-J	
Wireframe	Shift-K	
Freeze camera	Ctrl-N	
ViewImpostors	Shift-M	
ToggleCatStarOctree	Y	
ToggleStarOctree	Ctrl-Y	
ToggleCatGalOctree	U	
ToggleGalOctree	Ctrl-U	
ToggleGalSysOctree	Shift-U	
ToggleGalModelBoxes	Ctrl-Shift-U	
ToggleClosOctree	Shift-I	
ToggleNebModelBoxes	Ctrl-Shift-O	
ToggleLandOctree	Shift-P	
StarHistogrammColl	Shift-N	
IncStarLimLevel	Ctrl-Home	
DecStarLimLevel	Ctrl-End	
IncGalLimLevel	Shift-Home	
DecGalLimLevel	Shift-End	
IncPlanLimLevel	Alt-Home	
DelPlanLimLevel	Alt-End	
IncLayersMask	Ctrl-Up Arrow	
DecLayersMask	Ctrl-Down Arrow	
IncStarSaturation	Ctrl-Alt-Home	
DecStarSaturation	Ctrl-Alt-End	

IncStarColorShift	Ctrl-Shift-Home	
DecStarColorShift	Ctrl-Shift-End	
Reset stereobase	Shift-Numpad /	

Camera controls:

Function	Mouse	Keyboard	Arrows and numpad	Joystick with mode lamp on	Joystick with mode lamp off
Move forward		W	Up Arrow		
Move back		S	Down Arrow		
Move left		A	Left Arrow		
Move right		D	Right Arrow		
Move up		R	Numpad 1		
Move down		F	Numpad 0		
Turn left	- LeftMouse X		Numpad 4		
Turn right	+ LeftMouse X		Numpad 6		
Turn up	- LeftMouse Y		Numpad 8	Joystick ↓	
Turn down	+ LeftMouse Y		Numpad 2	Joystick ↑	
Turn clockwise		E	Numpad 9	Joystick →	
Turn counterclockwise		Q	Numpad 7	Joystick ←	
Increase velocity	Mouse wheel	+	Numpad +		
Decrease velocity	Mouse wheel	-	Numpad -		
Look back			Ctrl-Numpad 5		
Look left			Ctrl-Numpad 4		Joystick ←
Look right			Ctrl-Numpad 6		Joystick →
Look up			Ctrl-Numpad 8		Joystick ↓
Look down			Ctrl-Numpad 2		Joystick ↑

Horizon object		End			
Center object		C			
Go to object		G		Joystick K3	Joystick K3
Go faster to object		G G			
Land on object		Shift-G			
Go to object's center		Ctrl-G			
Track object		T			
Stop rotation			Numpad 5		
Stop motion		Z			
Next velocity		Ctrl-+	Ctrl-Numpad +	Joystick K12	Joystick K12
Previous velocity		Ctrl--	Ctrl-Numpad -	Joystick K11	Joystick K11
Next stereobase		Shift-+	Shift-Numpad +		
Previous stereobase		Shift--	Shift-Numpad -		
Reset stereobase			Shift-Numpad /		
?				Joystick K1	Joystick K1
Orbit around object left	- RightMouse X		Insert-Numpad 4		
Orbit around object right	+ RightMouse X		Insert-Numpad 6		
Orbit around object up	- RightMouse Y		Insert-Numpad 8		
Orbit around object down	+ RightMouse Y		Insert-Numpad 2		

Note: The keys marked in blue do only work when you have assigned the "Insert" key to the "Orbit Object" function. This is not the default setting.

Online gamepad and joystick tester: <https://gamepad-tester.com/>

Ship controls:

Move forward		W	Up Arrow
Move back		S	Down Arrow
Move left		A	Left Arrow
Move right		D	Right Arrow
Move up		R	Numpad 1
Move down		F	Numpad 0
Turn left	- LeftMouse X		Numpad 4
Turn right	+ LeftMouse X		Numpad 6
Turn up	- LeftMouse Y		Numpad 8
Turn down	+ LeftMouse Y		Numpad 2
Turn clockwise		E	Numpad 9
Turn counterclockwise		Q	Numpad 7
Rudder +			Numpad 6
Rudder -			Numpad 4
Elevator +	+ LeftMouse Y		Numpad 2
Elevator -	- LeftMouse Y		Numpad 8
Ailerons +	+ LeftMouse X	E	Numpad 9
Ailerons -	- LeftMouse X	Q	Numpad 7
Main engines +		+	Numpad +
Main engines -		-	Numpad -
Retro engines +		Shift-+	Shift-Numpad +
Retro engines -		Shift--	Shift-Numpad -
Hover engines +		Ctrl-+	Ctrl-Numpad +
Hover engines -		Ctrl--	Ctrl-Numpad -
Warp drive +		Alt-+	Alt-Numpad +

Warp drive -		Alt--	Alt-Numpad -
Look back			Ctrl-Numpad 5
Look left			Ctrl-Numpad 4
Look right			Ctrl-Numpad 6
Look up			Ctrl-Numpad 8
Look down			Ctrl-Numpad 2
Off HUD (Head-Up-Display)		Ctrl-1	
Horizontal HUD		Ctrl-2	
Orbital HUD		Ctrl-3	
Warp HUD		Ctrl-4	
Docking HUD		Ctrl-5	
Prograde		5	
Retrograde		6	
Radial		7	
Antiradial		8	
Normal Up		9	
Normal Down		0	
Horizon		End	
Hold altitude		H	
Flight assist		U	
Kill rotation			Numpad 5
Stop engines		Z	
Stop hyperdrive		Alt-Z	
Sync velocity		Shift-Z	
Fly to target		Shift-G	
Warp to target		Alt-G	

Fly or warp to target		G	
Rotate to target		C	
Reference body		P	
Target		T	
View ship trajectory		Alt-O	
Center on ship		Alt-C	

Map / Chart control

Move forward		W	Up Arrow
Move back		S	Down Arrow
Move left	- LeftMouse X	A	Left Arrow
Move right	+ LeftMouse X	D	Right Arrow
Move up	- LeftMouse Y	R	Numpad 1
Move down	+ LeftMouse Y	F	Numpad 0
Turn left			Numpad 4
Turn right			Numpad 6
Turn up			Numpad 8
Turn down			Numpad 2
Turn clockwise		E	Numpad 9
Turn counterclockwise		Q	Numpad 7
System architecture		1	
Size		2	
Mass		3	
Density		4	
Temperature		5	
Exploration mode		End	
Center object		C	
Go to object		G	

47.2 Planet classification

This chapter was written by Vladimir Romanyuk, source: <http://spaceengine.org/news/blog170924/>

The “approved” planet classification scheme is the following:

[temperature class] [volatiles class] [mass prefix][bulk composition class]

The temperature class is like in previous versions of SE, but with slightly changed limits:

frigid (90 K) cold (170 K) cool (250 K) temperate (330 K) warm (500 K) hot (1000 K) torrid

This format of notation means that, for example, the temperate class lies between 250 K and 330 K, and torrid is greater than 1000 K. These subdivisions are based on the properties of important substances:

90 K is the limit for the liquefaction of nitrogen, methane and other hydrocarbons found on very cold Titan-like worlds;

170 K is the water snow line in the Solar system and the temperature of outer asteroids in the Main asteroid belt;

250 K is the equilibrium temperature of Earth, and the assumed minimum average temperature of an Earth-like planet which could have temperate zones on its surface (with temperatures above water's freezing point);

330 K is close to the maximum temperature on Earth, and the assumed maximum average temperature of an Earth-like planet which does not fall into a runaway greenhouse effect;

500 K and 1000 K boundaries are arbitrary, and could be adjusted in the future to match some observational or theoretical characteristics of the hottest exoplanets.

The volatiles class is a combined atmosphere + hydrosphere description. It includes:

airless – a planet with an atmospheric pressure less than 1 nanobar;

desertic – a planet with atmospheric pressure greater than 1 nanobar, but with no liquids on its surface;

lacustrine – a planet with a small amount of liquid on its surface (lakes), and, obviously, with an atmosphere (because liquid cannot exist in a vacuum);

marine – a planet with seas of a liquid substance, i.e. a significant amount of it, but not completely covering the surface;

oceanic – a planet with a global ocean, completely covering the surface;

superoceanic – a planet with a very deep ocean (hundreds of kilometers deep), with exotic forms of ice forming on its bottom (ice VI and ice VII).

I will explain this in more detail in the next blog post.

The mass prefix is different for solid and gaseous planets; it is chosen to match the modern astronomical terminology at some points. Here ‘Me’ is Earth mass, and ‘Mj’ is Jupiter mass (318 Earth masses).

Solid planets (ferria, carbonia, terra, aquaria):

micro (0.002 Me) mini (0.02 Me) sub (0.2 Me) no prefix (2 Me) super (10 Me) mega

Ice giants (neptunes):

mini (4 Me) sub (10 Me) no prefix (25 Me) super (62.5 Me) mega

Gas giants (jupiters):

sub (0.2 MJ) no prefix (2 MJ) super (10 MJ) mega

The bulk composition classes define the fundamental substances of which the planet is made:

Feria – metals (iron, nickel) and siderophilic elements such as sulfur. The boundary is > 50% by mass.

Carbonia – carbon and its compounds like carbides, also CO and methane. The boundary is > 25% by mass.

Aquaria – water in the form of exotic ices and liquid (but not vapor or supercritical fluid – that is a minineptune). The boundary is > 25% by mass.

Terra – not ferria, not carbonia and not aquaria. The primary component is silicates (rocks).

Jupiter – hydrogen and helium. The boundary is > 25% by mass.

Neptune – not any of the previous classes. Typically, H/He is less than 25%; other substances are water/ammonia/methane and a rocky core. An aquaria with a supercritical vapor atmosphere is classified as a minineptune/subneptune.

Let me explain why the classification is as such.

The first 4 classes are “terrestrial” or “solid” planets, usually referred as “earths” in astronomy. Currently there is no accepted subdivision by bulk composition like this one, because astronomers can’t detect the bulk composition from observations of exoplanets – they can only estimate it based on a computed bulk density. This is enough to distinguish earths, neptunes and jupiters, but intermediate cases are hard to classify (a large water planet is indistinguishable from a small neptune, and a massive neptune is indistinguishable from a small jupiter). The exception is our Solar system – we can measure the density distribution inside a planet or moon by the precise measurement of the motion of a spacecraft near it. Combining this with knowledge about various substances under pressure (and for some planets, using information about the propagation of seismic waves), one can develop a model of the internal structure of a space body. Mercury with its large iron core (60% of the planet mass) should be classified as ferria in SE. This corresponds to a theoretical [iron planet](#). Venus, Earth, Mars, the Moon, and Io are classified as terra (Earth is a marine terra); other moons of gas giants and all dwarf planets are aquaria. Ceres and Europa are complex cases – they are on the boundary between terra and aquaria (about 25% of water and ice). Uranus and Neptune are neptunes, Jupiter and Saturn are jupiters. No carbonia worlds exist in the Solar system; this class is the theoretical [carbon planet](#). Aquaria corresponds to the theoretical [water or ocean planet](#), if it is warm enough to melt the icy crust; otherwise it corresponds with an [ice planet](#).

The most used mass/size prefixes in modern astronomy are super- and sub- for solid planets. The upper boundary of 10 Earth masses for [super-earths](#) is well defined, while the lower boundary differs in various sources. For SE I chose 2 Earth masses to make the scale more regular. There is speculation about the existence of more massive solid planets called [mega-earths](#). Possible candidate mega-earths are Kepler-10c and K2-3d, although recent re-estimations of their masses have moved them back to the super-earth class. The boundaries of the [sub-earth](#) class (in some sources called mini-earth) are also not well defined in literature; I chose it to be between 0.02 and 0.2 Earth masses so both Mercury and Mars fall into this class, but the Moon does not. Interestingly, Ganymede and Titan are also in the sub-earth class (they are subaquarias). The “mini” class is added to SE for planets less massive than 0.02 Earth masses; this is between the mass of the Moon and Ganymede. And the “micro” class is for the tiniest objects, which still have a round shape and should be classified as planets/dwarf planets/moons, unlike irregularly-shaped asteroids. The boundary of 0.002 Earth masses is arbitrary, just to make the scale uniform.

There is no sharp boundary between asteroids and “planetoids”, because it depends on composition, tidal heating and the history of the body. For example, Ceres has a mass of 0.0001566 Earths and a round shape, while the asteroid Vesta is 0.0000432 and non-round. So we could assume that the boundary for rocky asteroids is near 0.0001 Earth masses. But we have a counter-example: Mimas (Saturn’s moon) is just 0.0000062 Earths and round,

while Proteus (Neptune's moon) is slightly more massive – 0.0000083 Earths, and has an irregular shape. So the boundary for icy bodies cannot be defined by their mass only. For now, SE generates a slightly randomized boundary for each procedural body, about 0.0001 Earth masses for rocky objects and 0.000006 Earth masses for icy ones.

Ice giants, or “neptunes”, are planets formed mainly from water/ammonia/methane ices (in fact as a supercritical fluid) and often with a thick atmosphere of hydrogen and helium, up to 25% of their mass. But they can also be very massive rocky planets (more than 10 Earth masses) that have a supercritical atmosphere (that is, in a state of supercritical fluid with enormous pressures and temperatures of thousands of degrees). Therefore the term “neptune” looks more appropriate than “ice giant”; it also eliminates confusion from names like “hot ice giant”. Neptunes in SE have their own size prefix scale, overlapping with the classes of solid planets. The subdivision by mass is arbitrary, but astronomers often distinguish [mini-neptunes](#) into a separate class – those are small planets with extremely low bulk density (typically, rocky or water planets with a thick supercritical atmosphere). It corresponds to both the mini- and sub- classes in SE, but this additional subdivision is done for clarity (more details in the next blog post). The mega-neptunes are theoretical planets more massive than the smallest gas giants (~60 Earth masses), but composed predominantly of ices, not of hydrogen and helium. The overall scale is logarithmic with a 2.5 step: 4-10-25-62.5

Unlike ice giants, gas giants or “jupiters” are composed predominantly of hydrogen and helium. Hydrogen forms a metallic layer deep inside the planet. Saturn-mass planets are sometimes called “sub-jupiters” or “sub-giants”, or even “saturns”. So 0.2 Jupiter masses is a good choice for this class boundary (the mass of Saturn is 0.3 Jupiter masses). 0.2 Jupiter masses is roughly equal to 60 Earth masses, so the sub-jupiter class overlaps with the mega-neptune class. This emphasizes the difference in the nature of these two types of planets. The mass of 2 Jupiters is around the theoretical limit for the *largest* (by radius) planet. More massive gas giants are smaller because of the compression of gas by their enormous gravity. Such planets are called super-jupiters in this classification. And the most massive giants of more than 10 Jupiter masses are called mega-jupiters – they are close to the brown dwarf mass limit (13 Jupiters). Technically, brown dwarfs which have run out of deuterium fuel could be classified as extremely massive planets, so the mega-jupiter class could be used for them in future updates (for now SE doesn't model the evolution of brown dwarfs, and so can't determine if fusion already stopped). Super-jupiters and mega-jupiters in SE correspond to [super-jupiters](#) in literature, although there are huge differences in the definition of this class in various sources. The size prefix scale of gas giants resembles the scale of solid planets, so one can add the mini-jupiter class by analogy – planets with a mass lower than 0.02 Jupiters (about 6 Earth masses). It is doubtful that such small gas giants could exist in reality. The SE planet generator also don't produce them.

This classification resembles one proposed by the [Planetary Habitability Laboratory](#).

Changes in the engine made to support the new classification system include adding the bulk composition data into planet scripts. SE generates it procedurally if it's missing, but for the Solar system's planets and moons we want to have real data. This data is displayed in the Wiki. Note the new “Hydrosphere” tab – it displays information about seas, including their chemical composition (this is a topic for the next blog post). The procedural planet generator now creates all of the types of planets described above, including mini-neptunes, carbonias and even frozen terras.

Note: Brown dwarfs are considered as main sequence stars with spectral class types M, L, T, and Y.

47.3 Interesting objects

- **S/2009 S1, dwarf moon in Saturn's rings**
- **2010 TK7, Trojan asteroid on Earth's orbit**
- **Planemo = Free planet without a star: They are very dark, use Settings / Camera / Ambient_Lighting to make them visible.**

47.4 Export Textures

Tools --> Export Textures

The files will be saved to this folder: E:\SteamLibrary\steamapps\common\SpaceEngine\export

47.5 Export Skybox

Tools --> Export Skybox

Six cubemap files will be saved to this folder: E:\SteamLibrary\steamapps\common\SpaceEngine\export

The filenames are:

sky_neg_x.jpg sky_pos_x.jpg

sky_neg_y.jpg sky_pos_y.jpg

sky_neg_z.jpg sky_pos_z.jpg

The six cubemap images can be converted to an equirectangular image with this FFmpeg command:

```
ffmpeg -i sky_pos_x.jpg -i sky_neg_x.jpg -i sky_pos_y.jpg -i sky_neg_y.jpg -i sky_pos_z.jpg -i sky_neg_z.jpg -lavfi  
hstack=6,v360=c6x1:e -y out.jpg
```

```
pause
```

47.6 Scripts

This is an example script for Olympus Mons on Mars:

```
// Olympus Mons
// omons.sc
// Put in [SpaceEngine]\addons\scripts (for SE 0.980)
// Open console, type:
// Run omons (without .sc)
Time Current
//Time 2016.05.04 15:03:07
GotoFast Mars
Wait 3.0
```

Put it in 'E:\SteamLibrary\steamapps\common\SpaceEngine\addons\scripts', then open the console (with the tilde '~') and type 'run omons'

This is an example for overlaying a PNG image with alpha channel:

```
// Watermark
// Empress.se
Watermark
{
    Path "D:/Programme/Steam/steamapps/common/SpaceEngine/addons/textures/cockpits/Empress of the Universe.png"
    Opacity 1.0
    Visible true
}
```

See also <https://spaceengine.org/manual/making-addons/introduction/>

47.7 Useful links

Creating a star: <https://spaceengine.org/manual/making-addons/creating-a-star/>

Creating a planet: <https://spaceengine.org/manual/making-addons/creating-a-planet/>

Planet textures: <https://spaceengine.org/manual/making-addons/planet-textures/>

How to create a helium planet: <https://www.guideoui.com/spaceengine-0-990-how-to-create-a-helium-planet/>

Creating a deep space object: <https://spaceengine.org/manual/making-addons/creating-a-dso/>

Scenario scripts: <https://spaceengine.org/manual/making-addons/scenario-scripts/>

Space Engine Tutorial: <https://www.youtube.com/watch?v=Ln3xIU9Vkhg>

How to edit stuff in Space Engine: <https://www.youtube.com/watch?v=xMKGNjWlpjY>

Scott Manley: Space Engine - Seamlessly Explore The Entire Universe <https://www.youtube.com/watch?v=Htyv6m9jaNY>

Screenshots folder location: <https://www.guideoui.com/spaceengine-0-990-screenshots-folder-location/>

47.8 Abbreviations

- RS = Random Star
- RG = Random Galaxy
- RSC = Random Star in a Cluster
- RN = Random nebula

47.9 Suggestions for improvement

- Messier objects are difficult to find because there is a space character between "M" and the number, which is quite unusual.
- Saturn's ring particles are always behind S/2009 S1, never in front of it.
- Add brown dwarfs
- Add mergers of black holes or neutron stars
- Saved locations don't work in flight simulator mode
- Stars close to the black hole in center of the milky way appear too late when the observer gets closer to them
- Nebula RN 8496-8003 Central stars appear dark red, even when the nebula is behind the stars
- From about 2000000km distance, the moon appears brighter than the earth (only in HDR mode)
- Implement a realistic user-defined start location on earth, for example with several equirectangular drone images at height 0m, 10m, 20m, ... 100m? I would really like to start my spaceship from my own location.
- Renaming saved locations: It should be possible to append some text to the location name. But when new text is typed in, the old text is immediately deleted.
- "Display" menu: What's called "Cylindrical" is in fact "Equirectangular". Calling the projection "Cylindrical" isn't wrong because equirectangular projection belongs to the large class of cylindrical projections. "Equirectangular" is just more specific.
- Add the Cirrus nebula = NGC 6960 and NGC 6995
- Add the Blue snowball = NGC 7662
- More details for Ina structure on the moon [https://en.wikipedia.org/wiki/Ina_\(crater\)](https://en.wikipedia.org/wiki/Ina_(crater))
https://wms.lroc.asu.edu/lroc/view_rdr/NAC_DTM_INACALDERA1
- Add NGC1360 Robins Egg Nebula

48 Fitswork

Instruction manual (in german):

<https://www.fitswork.de/anleitung/>

<https://www.fitswork.de/anleitung/rawflow.php>

How to select images for stacking:

- Click on "Anfangsdatei" and select the images while holding CTRL or SHIFT down.
- Or use "Alle Dateien im Ordner" if you want to use all images in the folder.
- Or select the first image and stack until the end of the folder.

After the images have been selected, click on the ">" arrow at the right of the window and specify which processing steps shall be done.

49 DeepSkyStacker 4.2.3

DeepSkyStacker 4.2.3 can be downloaded here: <http://deepskystacker.free.fr/english/index.html>

Support group: <https://groups.io/g/DeepSkyStacker>

Youtube video from Frank Sackenheim (in german): <https://www.youtube.com/watch?v=LrMptU0kLPE>

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!

49.1 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.

49.2 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 its 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: $-0.42^\circ/d * 0.0102d = -0.00428^\circ$, $-0.36^\circ/d * 0.0102d = -0.00367^\circ$
- Calculate the image scale in degrees/pixel. Example for a full frame camera with 36mm image width, 6744 horizontal pixels and a 400mm lens: $\arctan(36mm / (6744 * 400mm)) = 0.000765^\circ/\text{pixel}$
- Now you can calculate how many pixels the comet has moved between the two images. Example: $X = 0.00428^\circ / 0.000765^\circ/\text{pixel} = -5.60 \text{ pixel (to the left)}$, $Y = -0.00367^\circ / 0.000765^\circ/\text{pixel} = -4.80 \text{ pixel (downwards)}$
- 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:

RA_speed = RA comet motion in degrees per day =

DE_speed = DE comet motion in degrees per day =

T1 = Time of first image =

T2 = Time of last image =

dT = Time difference in days = T2 - T1 =

RA_deg = RA comet movement in degrees = RA_speed * dT =

DE_deg = DE comet movement in degrees = DE_speed * dT =

S = Image scale in degrees per pixel =

For Canon 5D-MK4 with 400mm lens: 0.000765°/pixel

RA_pix = RA comet movement in pixels = RA_deg / S =

DE_pix = DE comet movement in pixels = DE_deg / S =

X_old = Original X value in info.txt =

Y_old = Original Y value in info.txt =

X_new = New X value in info.txt = X_old + RA_pix =

Y_new = New Y value in info.txt = Y_old - DE_pix =

50 SharpCap

<https://www.sharpcap.co.uk/>

51 AutoHotKey

AutoHotkey is a free, open-source scripting language for Windows that allows users to easily create scripts for all kinds of tasks such as: redefining function keys, auto-clicking, macros, etc. <https://www.autohotkey.com/>

How to make a remote control for SharpCap, which can be used outside at the telescope in winter with gloves on?

Get the cheapest USB keyboard you can find. Disassemble it and keep only the electronics. Examine the keyboard matrix and find out which contacts must be connected for the F1-F12 keys. Replace the original keyboard matrix by large bushbuttons. I did use 12 buttons, but you can use less or more if you want. Search for "Arcade button" on Aliexpress or Ebay. Buttons are also available with built-in LED for illumination.

This is my remote control for SharpCap:



This is the AutoHotKey script for my SharpCap remote control:

```
#IfWinActive SharpCap

F1::           ;
  MouseClick, left,1051,65      ; Zoom
  MouseClick, left,1030,85      ; Auto
  MouseMove 1600,50            ; Move the mouse out of the picture
  return

F2::           ;
  MouseClick, left,1051,65      ; Zoom
  MouseClick, left,1030,310     ; 300%
  MouseMove 1600,50            ; Move the mouse out of the picture
  return

F3::           ;
  send,{F1}                     ; Exposure -
  return

F4::           ;
  send,{F2}                     ; Exposure +
  return

F5::           ;
  send,!t m                     ; Seeing Monitor on/off
  ; Alt-t m
  return

F6::           ;
  MsgBox,,,F6,2                 ; unused
  return

F7::           ;
  send,{F3}                     ; Gain -
  return

F8::           ;
  send,{F4}                     ; Gain +
  return

F9::           ;
  ; Histogram on/off
```

```
send,!t h          ; Alt-t h
return

F10::               ; Focus Assistant on/off
send,!t f {Enter} ; Alt-t f Enter
return

F11::               ; Quick Start
send, !q           ; Alt-q
SoundBeep,1000,200 ; Beep high
return

F12::               ; Stop Capture
send, {esc}         ; Escape
SoundBeep,500,200  ; Beep low
return
```

Using a joystick as a mouse: <https://www.autohotkey.com/docs/scripts/index.htm#JoystickMouse>

Remapping a joystick to keyboard or mouse: <https://www.autohotkey.com/docs/misc/RemapJoystick.htm>

52 Autostakkert!

<https://www.autostakkert.com/>

Set the window size: ctrl-1 ... ctrl-9

https://www-astrokraai-nl.translate.goog/tut/guide_dennis_put.htm?_x_tr_sl=en&_x_tr_tI=de&_x_tr_hI=de&_x_tr_pto=nui

53 Tycho Tracker

<https://www.tycho-tracker.com/>

54 C# Programming project / Digital maps and elevation data

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.

54.1 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).

Rain sounds: <https://www.youtube.com/watch?v=hKfgHAE7pYU>

54.2 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:

```
n = 2 ^ zoom
x = n * ((lon_deg + 180) / 360)
y = n * (1 - (log(tan(lat_rad)) + sec(lat_rad)) / pi)) / 2
```

Coordinate transformation from X and Y to latitude and longitude:

```
n = 2 ^ zoom
lon_deg = x / n * 360.0 - 180.0
lat_rad = arctan(sinh(pi * (1 - 2 * y / n)))
lat_deg = lat_rad * 180.0 / pi
```

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:

```
resolution = 156543.03 meters/pixel * cos(latitude) / (2 ^ zoom)
```

Note: You can create your own layers in the french version of OpenStreetMap: <umap.openstreetmap.fr>

54.3 Digital elevation data

I found several sources for free digital elevation data:

- Bundesamt für Kartografie und Geodäsie, elevation data on a 200m grid is free, and higher resolution data must be paid (too expensive for my project): <https://gdz.bkg.bund.de/index.php/default/catalog/product/view/id/756/s/digitales-gelandemodell-gitterweite-200-m-dgm200/category/8/>
- NASA Earthdata 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/> 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 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.

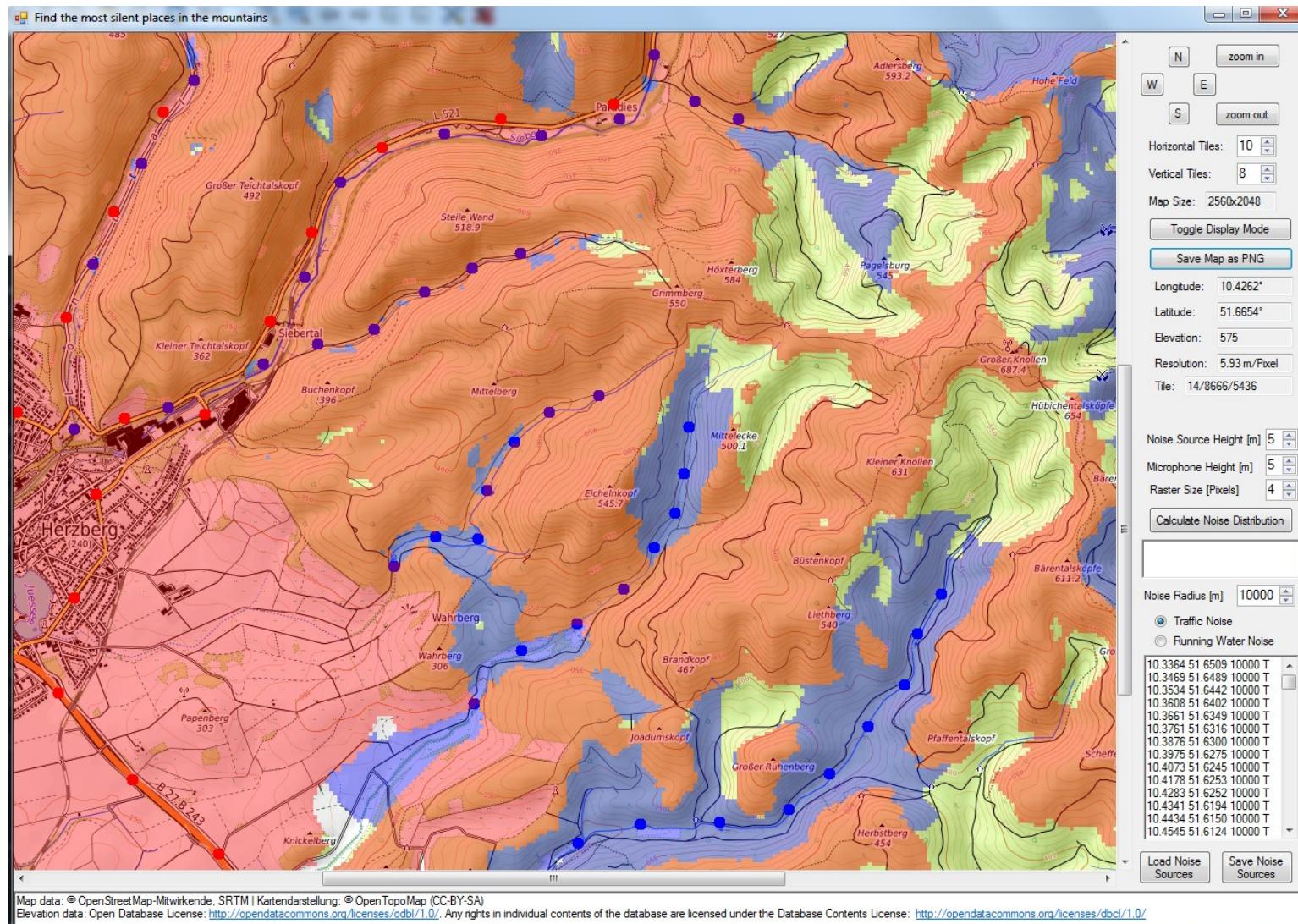
I found useful instructions here: <http://build-failed.blogspot.com/2014/12/processing-geotiff-files-in-net-without.html>

This is the link to the library: <https://bitmiracle.com/libtiff/>

My source code can be downloaded here: <http://www.astro-electronic.de/source/Topographie.zip>

54.4

Noise map (red = traffic noise, blue = running water noise)



55 Astronomy

55.1 Moon observing

What's the best time for moon observing? That depends on the moon's phase as follows:

Moon phase:	Largest altitude:	When has the ecliptic the steepest angle to the horizon?
New Moon	(not during darkness)	
Waxing crescent	(not during darkness)	Spring, March 20, at sunrise
First quarter	Spring, March 20, 18:00 o'clock	
Waxing gibbous	Winter, February 5, 21:00 o'clock	
Full moon	Winter, December 20, 0:00 o'clock	
Waning gibbous	Autumn, November 5, 3:00 o'clock	
Last quarter	Autumn, September 20, 6:00 o'clock	
Waning crescent	(not during darkness)	Autumn, September 20, at sunset

The given dates are a rough estimate plus or minus one month, only valid for observers on the northern hemisphere.

We always see less than 50% of the moon's surface, typically 49.78%.

The visible surface area is: $A = 50\% * (1 - \sin(\theta))$ where theta is half of the angular diameter (about 0.25°).

Interactive map of the moon: <https://quickmap.lroc.asu.edu/>

Irregular mare patches: https://en.wikipedia.org/wiki/Irregular_mare_patch

Crater Ina: [https://en.wikipedia.org/wiki/Ina_\(crater\)](https://en.wikipedia.org/wiki/Ina_(crater))

Digital elevation data for Ina: https://wms.lroc.asu.edu/lroc/view_rdr/NAC_DTM_INACALDERA1

Resolution of the moon surface (in arc seconds):

Distance	1 km	2 km	4 km
350702 km (Perigee)	0.59"	1.18"	2.35"
378692 km (mean distance)	0.54"	1.09"	2.18"
401032 km (Apogee)	0.51"	1.03"	2.06"

Note: The earth's radius for latitude 51.5° and the moon's radius are already subtracted.

55.2 Moon videos

Very good videos of the moon surface by Seán Doran:

- Orbit the Moon in Ultra High Definition 4k <https://www.youtube.com/watch?v=BFNUya4Na6k>
- MOON in Real Time I <https://www.youtube.com/watch?v=ctqXSOJuaRE>
- MOON in Real Time II <https://www.youtube.com/watch?v=lfrQ5dczECY>
- Low Lunar Orbit <https://www.youtube.com/watch?v=XU8zZjLaEjE>
- Orbit : Moon [4K] <https://www.youtube.com/watch?v=UOcroR50808>
- Selene - Orbit the Moon <https://www.youtube.com/watch?v=MamH-Jy3j8s>
 - The beginning is a flight over the south pole.
 - 3:50 Demonax
 - 5:15 Aitken
 - 6:01 Van de Graaff
 - 6:18 Sikorsky and Vallis Schrödinger
 - 7:34 Kugler
 - 8:10 Lebedinskiy
 - 9:50 Jackson
 - 10:20 Byrgius, left D, right A
 - 10:46 Rimae Darwin and Rima Sirsalis
 - 11:11 Crüger
 - 12:13 Grimaldi
 - 12:29 left Karpinskiy, right Roberts
 - 14:48 Plaskett, top right Rozhdestvenskiy (that's near the north pole)
 - 18:33 Segner / Segner H

- 19:13 Zucchius
- 19:52 top right Bettinus, in the center Bettinus A
- 20:30 top left Hohmann Q (in Mare Orientale)
- 21:20 Rimae Focas
- 22:30 left Focas U
- 24:24 Humboldt
- 25:24 Tycho
- 26:40 Mare Ingenii on the lunar far side
- 27:16 In the middle Thomson V, at the right Thomson W
- 28:00 'Albedo Swirls' are thought to be produced by remanent magnetism in the Moon's crust
- 28:18 left Obruchev X
- 29:00 Obruchev

3D Simulation of the Shackleton Crater: <https://www.youtube.com/watch?v=izl9HVo8bjc>

55.3 Limiting magnitude for video astronomy

Lens	Camera	Video mode	ISO	Limiting magnitude	Notes
Canon EF 400mm f/2.8 + SpeedBooster 0.64x	Panasonic GH5S	FHD 25fps, [Ex. Tele Conv.] = 2.1x	25600	about 12.2 mag	Sky wasn't perfectly clear, with 4x contrast enhancement, no noise reduction

Dragonfly project with 400mm f/2.8 lenses: https://en.wikipedia.org/wiki/Dragonfly_Telephoto_Array

<http://www.dunlap.utoronto.ca/instrumentation/dragonfly/>

<https://arxiv.org/pdf/1401.5473.pdf>

55.4 Limiting magnitude for stacked exposures

Lens	Camera	Exposure	ISO	Limiting magnitude	Notes
Canon EF 400mm f/2.8	Canon 5D MK4	173 x 30s = 86.5 min	3200	about 18.5 mag	Stacked with DeepSkyStacker, sky wasn't perfectly clear

55.5 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]:

$$M_v = -14.18 - 2.5 \cdot \log_{10} E_v$$

Convert illuminance E_v [Lux] to irradiance E_E [W/m^2], for wavelength 555nm:

$$E_E = E_v / 683 \text{ lx/W}$$

Convert irradiance E_E [W/m^2] to illuminance E_v [Lux], for wavelength 555nm:

$$E_v = E_E \cdot 683 \text{ lx/W}$$

55.6 Artificial Stars

mag	-1	0	1	2	3	4	5	6	7
Intensity = $10^{-(\text{mag}/2.5)}$	2.5	1	0.4	0.16	0.063	0.025	0.01	0.004	0.0016

Diameter = sqrt(Intensity)	1.6	1	0.63	0.4	0.25	0.16	0.1	0.063	0.04
Diameter * 4	6.3	4	2.5	1.6	1	0.63	0.4	0.25	0.16
Diameter * 2.5	4	2.5	1.6	1	0.63	0.4	0.25	0.16	0.1
Diameter * 1.6	2.5	1.6	1	0.63	0.4	0.25	0.16	0.1	--

Intensity = $10^{-\text{mag}/2.5}$

Note: Available drill sizes are 0.1mm, 0.15mm, 0.25mm, 0.4mm, 0.65mm, 1.0mm, 1.6mm, 2.5mm, 4.0mm, 6.3mm

55.7 Viewing below the horizon

How deep can we look below the horizon from a high mountain?

$$\alpha = \arccos(R / (R + H))$$

with R = radius of earth and H = height above ground

55.8 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 $1.326\mu\text{Hz}$ per hour.
2. The Doppler effect due to earth's rotation. The velocity of the observer is: $V = 2 * \pi * 6370\text{km} / 86400\text{s} * \cos(\text{latitude})$
For 51.5° latitude the velocity is 0.288 km/s towards the east point of the local horizon.
The Doppler frequency shift is $f_D = f * V / c$ where c is the speed of light 300000 km/s.
For $f = 30\text{Hz}$ the Doppler frequency shift is $28.8\mu\text{Hz}$ at the east horizon and $-28.8\mu\text{Hz}$ at the west horizon. The frequency shift is 0 at the meridian.
Near the meridian the Doppler frequency shift decreases by $7.5\mu\text{Hz}$ per hour.
3. The Doppler effect due to earth orbiting around the sun. The velocity of the observer is: $V = 2 * \pi * 149.6e6\text{ km} / 365.25d / 86400\text{s} = 29.786\text{ km/s}$ towards a point on the ecliptic which is about 90° west of the sun.
The Doppler frequency shift is $f_D = f * V / c$ where c is the speed of light 300000 km/s.
For $f = 30\text{Hz}$ the Doppler frequency shift is $99.3\mu\text{Hz}$ on the ecliptic 90° west of the sun and $-99.3\mu\text{Hz}$ 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 $2.13\mu\text{Hz}$ per hour.

Adding these three effects, the observed pulsar frequency decreases by about $11\mu\text{Hz}$ per hour (valid only in winter at midnight, when the pulsar is in the south near the meridian).

An error Δf in the reference frequency will produce after time t a phase error Δp with respect to the pulsar phase as follows: $\Delta p / 360^\circ = t * \Delta f$

Example: If the reference frequency is off by $10\mu\text{Hz}$, after 2 hours there will be a phase error of 25.92° .

The latest exact frequency of the pulsar is from December 15, 2019:

$$f = 29.6122791665 \text{ Hz} - 0.000001326 \text{ Hz / hour}$$

55.9 How to visualize the size of the universe

I found this in the discussion forum for Space Engine:

"If you spread out all the stars over the surface of the earth, each square cm would have 60.000 stars in it. And on average, of the 60.000 stars, around 2600 of them will have Super Earth's within their habitable zone."

55.10 Solar System Model

Scale	Diameter of sun	Distance sun - earth	Diameter of earth	Distance earth - moon	Diameter of moon
1:1	696342km	149.6e6km	12700km	384000km	3474km
1 : 1e10	70mm	15m	1.27mm	38.4mm	0.35mm
1 : 1e11	7mm	1.5m	0.13mm	3.84mm	0.035mm

Sizes of needles:

Beadsmith ZB10512 Diameter 0.36mm / Width at end 0.50mm

Beadsmith ZB10513 Diameter 0.30mm / Width at end 0.40-0.45mm

Beadsmith ZB10515 Diameter 0.25mm / Width at end 0.45mm

55.11 Selfmade Planetariums

- Planetarium with 1.5m dome in Sankt Andreasberg observatory: <http://www.astro-electronic.de/planetarium.htm>
- Tactile planetarium in this video, beginning at 1:03:00 <https://www.youtube.com/watch?v=2422JA9XF6M>

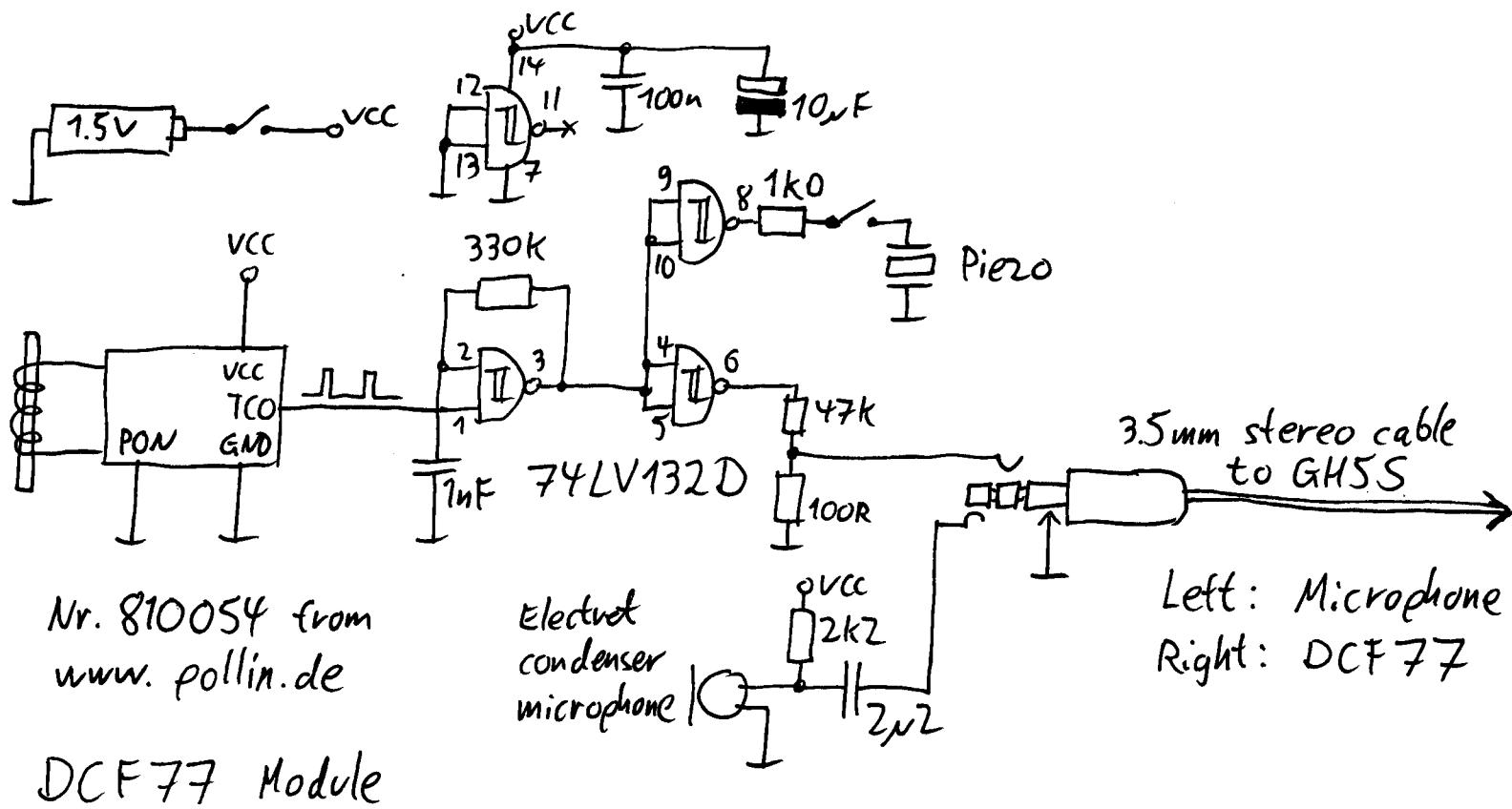
56 DCF77 Decoding

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:

Bit	Description
0	Start of minute, is always 0
1-16	Civil warning bits and other informations
17	CET: 0 CEST: 1
18	CET: 1 CEST: 0
19	Leap second announcement
20	Always 1
21	Minutes 1
22	Minutes 2
23	Minutes 4
24	Minutes 8
25	Minutes 10
26	Minutes 20
27	Minutes 40
28	Even parity over minute bits 21-2

Bit	Description
29	Hours 1
30	Hours 2
31	Hours 4
32	Hours 8
33	Hours 10
34	Hours 20
35	Even parity over hours bits 29-35
36-41	Day of month
42-44	Day of week
45-49	Month number
50-57	Year within century
58	Even parity over the date bits 36-58
59	No impulse
0	Previous defined time is valid at the beginning of this impulse (which is always 0)

Schematic diagram of a DCF77 receiver which produces 2kHz beeps for recording with the GH5S camera:



57 The invisible Lattice Mast

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.

The C# program for calculating the lattice mast can be downloaded here: <http://www.astro-electronic.de/source/Gittermast.zip>

This is the result:

```
Lattice mast calculation:
```

```
Base width: 200.0
```

```
Base angle: 86.5°
```

```
Diagonal angle: 45.0°
```

```
Diagonal: 266.5    Width: 176.9    Height: 188.5
```

```
Diagonal: 235.8    Width: 156.5    Height: 355.2
```

```
Diagonal: 208.6    Width: 138.5    Height: 502.7
```

```
Diagonal: 184.6    Width: 122.5    Height: 633.3
```

```
Diagonal: 163.3    Width: 108.4    Height: 748.7
```

```
Diagonal: 144.5    Width: 95.9    Height: 850.9
```

```
Diagonal: 127.8    Width: 84.9    Height: 941.3
```

```
Diagonal: 113.1    Width: 75.1    Height: 1021.2
```

```
Diagonal: 100.1    Width: 66.4    Height: 1092.0
```

```
Diagonal: 88.5    Width: 58.8    Height: 1154.6
```

```
Diagonal: 78.3    Width: 52.0    Height: 1210.0
```

```
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 :-)

58 Fireflies, glowworms, lightning bugs

See also: <https://en.wikipedia.org/wiki/Firefly>

There are only three firefly species in central Europe:

- Kleiner Leuchtkäfer, Gemeines Glühwürmchen, Johanniskäfer oder Johanniswürmchen (*Lamprohiza splendidula*)
See also (only available in german): https://de.wikipedia.org/wiki/Kleiner_Leuchtk%C3%A4fer
Search for them in warm summer nights in June and July, in riparian forests.
- Großer Leuchtkäfer, Großes Glühwürmchen oder Großes Johannisglühwürmchen (*Lampyris noctiluca*)
See also https://en.wikipedia.org/wiki/Lampyris_noctiluca
- Kurzflügel-Leuchtkäfer (*Phosphaenus hemipterus*)

Source: <https://de.wikipedia.org/wiki/Leuchtk%C3%A4fer>

Interactive map (in german) showing where fireflies have been observed: <https://www.naturgucker.de>

Search for the species "*Lamprohiza splendidula*", "*Lampyris noctiluca*" or "*Phosphaenus hemipterus*".

Larvae of *Lampyris noctiluca*? Found in December 2021 near Bad Lauterberg. Pictures by Sandra Zi.



59 Spherical Trigonometry and Rotation Matrices

See also https://en.wikipedia.org/wiki/Rotation_matrix

See also <http://paulbourke.net/geometry/transformationprojection/>

This is the rotation matrix for yaw rotation:

```
( cos(yaw) -sin(yaw) 0 )
( sin(yaw) cos(yaw) 0 )
( 0 0 1 )
```

This is the rotation matrix for pitch rotation:

```
( cos(pitch) 0 sin(pitch) )
( 0 1 0 )
( -sin(pitch) 0 cos(pitch) )
```

This is the rotation matrix for roll rotation:

```
( 1 0 0 )
( 0 cos(roll) -sin(roll) )
( 0 sin(roll) cos(roll) )
```

This is the rotation matrix for yaw, pitch and roll rotations (in this order):

```
( cos(yaw) * cos(pitch) cos(yaw) * sin(pitch) * sin(roll) - sin(yaw) * cos(roll) cos(yaw) * sin(pitch) * cos(roll) + sin(yaw) * sin(roll) )
( sin(yaw) * cos(pitch) sin(yaw) * sin(pitch) * sin(roll) + cos(yaw) * cos(roll) sin(yaw) * sin(pitch) * cos(roll) - cos(yaw) * sin(roll) )
( -sin(pitch) cos(pitch) * sin(roll) cos(pitch) * cos(roll) )
```

Angular distance between two stars:

```
cos(distance) = cos(alpha1 - alpha2) * cos(delta1) * cos(delta2) + sin(delta1) * sin(delta2)
```

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:

```
Distance^2 = (alpha1 - alpha2)^2 + (delta1 - delta2)^2
```

60 My To Do List

-- Darkframe subtraction with DaVinci Resolve?

-- Stitching 360° videos from two or more cameras

-- Spherical transformations for dlp beamer

-- There seems to be a surprisingly large void of information about these subjects and their interactions:

FFmpeg / HDR / SDR / zscale / tonemapping / DNG / Rec709 / Rec2020 / colorspace / colormatrix

<https://web.archive.org/web/20190722004804/https://stevens.li/guides/video/converting-hdr-to-sdr-with-ffmpeg/>

<https://codecalamity.com/encoding-uhd-4k-hdr10-videos-with-ffmpeg/>

```
-f mp4: file format/container is mp4
-f h264: file format is h264, even if the file extension is mp4
```

61 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

DLP = Digital Light Processing (Beamer)

DR = Dynamic Range or DaVinci Resolve

DTS = Decoder Time Stamp

DVR = Digital Video Recoder, but the abbreviation is also used for "DaVinci Resolve"

ETC = Extra Tele Conversion

ETTL = Expose To The Left

ETTR = Expose To The Right

EV = Environment Variable

FPS = Frames Per Second

FX = Special Effects

GFX = Graphical Effects

GOP = Group of Pictures

HDR = High Dynamic Range

HEVC = High Efficiency Video Coding

HLG = Hybrid Log Gamma

HSB = Hue-Saturation-Brightness

HSV = Hue-Saturation-Value

IPB = Interframe compression method, which saves three types of frames: Intraframes (I-frames), predicted frames (P-frames) and bi-directional predicted frames (B-frames)

IR = Impulse Response

IRE = a measurement of composite video signals, 100 IRE = the difference between black and white level

LHS = Left Hand Side

LUT = Look-up-Table

MTF = Modulation Transfer Function

NR = Noise Reduction

OSC = One Shot Color

POV = Point Of View, which means a scene in which the camera sees what a person sees

PQ = Perceptual Quantization

PTS = Presentation Time Stamp

RHS = Right Hand Side

SAR = Sample Aspect Ratio = the aspect ratio of the pixels

SDR = Standard Dynamic Range

SMPTE = A timecode format, see https://en.wikipedia.org/wiki/SMPTE_timecode

SNR = Signal to Noise Ratio

TBC = Time Base Corrector, corrects/standardizes the timing of analog signals coming from tape, see https://en.wikipedia.org/wiki/Time_base_correction

VFR = Variable Frame Rate

VO = Voice-Over

62 Special characters

- Windows: Click in lower left corner on the Windows symbol, then choose "Windows Accessories" --> "Character Map"
- Unicode character recognition: Shapecatcher <http://shapecatcher.com/index.html>

63 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 Anders Jiras and Andrei B. for a few hints for FFmpeg.

Thanks to Alexander Strasser for helping me with GIT.

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.

Thanks to Gerald Schulze for his DaVinci Resolve course.

Thanks to Hugh Hou for pointing me to the KartaVR plugin for DR, and for several very helpful Youtube videos about 360° content.

Thanks to Andrew Hazelden for the KartaVR plugin for DR.

Thanks to Sandra Zi for pictures of fireflies.

Thanks to Rainer Rosenthal and Andreas Leitgeb for help with a complicated mathematical problem is de.sci.mathematik

Thanks to Gavin Lucas and Bernd Klimm for a great DaVinci Resolve Webinar.

64 Index

0-padded.....	339	2kscope.....	377	7° rule of thumb.....	645
0x07.....	618	2x Teleconverter.....	668, 692	7artisans (Viltrox) 7.5mm.....	655
0xAABBGGRR.....	389	2x2 mosaic.....	112	Abbreviations, GH5S.....	830
0xRRGGBBAA.....	389	3-axis gravity sensor (Ricoh Theta).....	712	abgr.....	319
1.4x Teleconverter.....	668, 692	3.5mm headphone output.....	482	abs.....	250, 420
1/4 wave antenna.....	640	3.5mm input jack.....	482	Absolute angles (v360 filter).....	459
1/90000s clock.....	511	3.5mm stereo connector.....	482, 749	Abstraction (DR).....	563
1>.....	427	360° Drone Video.....	703	ACER P7605.....	661
1> log.csv.....	368	360° Panoramic camera.....	711	ACES Transform (DR).....	562
1> measured.csv.....	283	360° stabilizer (Shotcut).....	612	Acknowledgements.....	832
10 bit video (DR).....	606	360° videos (VLC).....	621	acos.....	420
10 Step (DR).....	561	3D Camera Tracking (DR).....	580	Acoustic feedback.....	588
10-bit accuracy in filtergraphs.....	362	3D Effects (DR).....	578	Acoustic feedback (DR).....	590
10-bit compatibility.....	362	3D Keyer (DR).....	577	acrossfade.....	460, 463
10-bit test video.....	366	3D Look-up table.....	356	Add a new "MediaIn" node (DR).....	538
10-bit video.....	59, 430	3D LUT Creator.....	625	Add borders.....	91
10-bit VLog test video.....	679	3D Plant Models.....	584	Add one or more sound effects.....	466
100mV Steps (DR).....	562	3D Water Effect (DR).....	579	addition (blend filter).....	118
16-Bit.....	370	3x3 neighborhood.....	164	Additive color mixing.....	353
16-bit GeoTiff.....	813	4:2:0.....	318	adecorrelate.....	442
16-bit PNG.....	372	4:2:2.....	318	adelay.....	317, 466, 499, 507
16cif.....	377	4:4:4.....	318	Adjusting telescope optics.....	247
18% gray.....	677	4096x4096.....	111	Adjustment clips (DR).....	572
180° fisheye.....	192	4cif.....	376	Adobe Digital Negative.....	370
1kHz tone.....	359	4k.....	375, 377	Adobe DNG converter.....	370, 616
2.1x Telescopic effect.....	690	4K.....	654	Adobe DNG Converter.....	175
2.5mm connector.....	689	4K 10 bit Modes GH5S.....	699	Adobe Lightroom (DNG export).....	372
2>.....	427	4K 8 bit Modes GH5S.....	698	ADR (DR).....	590
2> out.txt.....	427f.	4kdci.....	375, 377	adrawgraph.....	393
2010 TK7.....	797	4kflat.....	375, 377	aegisub.....	403
2k.....	375, 377	4kscope.....	377	aeval.....	469, 481, 728
2kdci.....	375, 377	5.1 channel layout.....	738	aevals...89, 475, 488, 490, 492, 496, 498, 737f.	
2kflat.....	375, 377	5KPlayer.....	716	AF Icon (GH5S).....	674

afade.....	88, 378, 385, 466
afade curves.....	89
AFC (GH5S).....	674
AFF (GH5S).....	674
afftfilt.....	442, 483, 497
afreqshift.....	442, 476f.
AFS (GH5S).....	674
airless (SE).....	794
alignment error (double fisheye).....	231
ALL-I.....	667, 830
Alpha channel.....	336
Alpha masking (DR).....	565
Alpha Matte Shrink and Grow (DR).....	563
alpha_mask.....	220f., 224, 236, 263, 284
alpha=0 means transparent.....	141, 336, 565
alpha=255 means opaque.....	141, 336, 565
alphaextract.....	147
alphamerge.....	147
Alternating left/right stereo sound.....	508
altitude of the sun.....	767
Amateras Dome Player.....	727
ambisonic sound.....	733
ambisonic test tone.....	730
Ambisonics.....	728
amege.....	469, 472
amix.....	443, 466, 480, 507
amovie.....	513
Amplifier.....	473
amplify.....	75, 443
amultiply.....	475, 482
Analog Damage (DR).....	564
analyze clip lengths (DR).....	529
Anamorphic.....	654
Anamorphic 10 bit Modes GH5S.....	700
Anamorphic 8 bit Modes GH5S.....	699
and (blend filter).....	118
Angular distance.....	201
Angular to Equirectangular (DR).....	593
Animated GIF.....	113, 115
Animated PNG.....	115
animation.....	334
anoisesrc.....	497
ANSI.....	97
anullsrc.....	101, 467f.
apad.....	475, 499
Aperture.....	655
Aperture Diffraction (DR).....	563
Aperture height (DR).....	581
Aperture number.....	662
Aperture width (DR).....	581
aphaseshift.....	443
Apogee.....	817
Append a Fusion composition (DR).....	537
Append at End (DR).....	531
Append text.....	619
Apply a look-up-table (DR).....	560
Apply Grade (DR).....	541
Apply LUT (OBS).....	636
Apprehension Engine.....	752
apt-get build-dep ffmpeg.....	648
apt-get install.....	647
apt-get install build-essential.....	648
apt-get install ffmpeg.....	647
apt-get install yasm.....	648
Aquaria (SE).....	795
Arcade button.....	807
Archive file (DR).....	528
Archive projects (DR).....	528
aresample.....	378, 463, 465
areverse.....	88
argb.....	319
ARIB STD-B67.....	681
ARIB STD-B67 HLG (DR).....	558
Artefacts.....	163
Artificial lightning (DR).....	570
Artificial Stars.....	819
ascent (drawtext filter).....	98
ASCII.....	97
ASCII PGM (P2).....	455
ASCII text file.....	427
asetnsamples.....	393
asetrate.....	465, 475
ASI178MM.....	654, 662
asin.....	218, 420
asplit.....	477, 507, 737
ass.....	402
astats.....	393, 509, 513
Asteroid.....	390, 772
Astronomy.....	357, 816
Astronomy simulation.....	782
asubboost.....	443, 736f.
asubboost block diagram.....	737
atadenoise.....	315, 363, 380f.
atan.....	420
atan2.....	169, 179, 190, 216ff., 223, 249, 262, 420
atempo.....	465, 475
atilt.....	444
Atkinson, Grant.....	671
Atmospheric dispersion correction.....	74
atrim.....	480, 738
Attach Cover Art.....	408
attached_pic.....	407f.
Attaching a preview image to a video.....	407
Audible range.....	475
Audio length.....	465
Audio pitch.....	465
Audio processing.....	462
Audio sample rate.....	463
Audio samples per frame.....	511
Audio samples per pixel.....	501
Audio SweepGen.....	496
Audio volume.....	463
Audio volume (DR).....	536
Audio volume level.....	473

Audio waveform.....	500	bench=stop.....	440
Auto Balance (DR).....	540	Benchmarks.....	440
Auto Track (DR).....	581	best time for moon observing.....	816
Auto track selector (DR).....	522	Betagear FX82USB mixer.....	755
Auto Track Selector (DR).....	535	between.....	227, 258, 420, 422, 449, 497
AutoHotKey.....	807	BGR.....	326
Automated Dialog Replacement (DR).....	590	bgr0.....	192
Automatic Dirt Removal (DR).....	563	bgr24.....	319
Automatic format conversion.....	325	bgr48be.....	319
Autostakkert!.....	810	bgr48le.....	319
AVCHD.....	675	bgra.....	192, 319
average (blend filter).....	118	Bi-directional predicted frames.....	830
averaging.....	380	Bidirectional Tracking (DR).....	581
AVFrame struct.....	374	Big eyes.....	312
avgblur.....	70, 75	BigBlueButton.....	641
avsynctest.....	444, 508	Bigger nose.....	310
avutil-56.dll.....	292	BIMP (Plugin for GIMP).....	372
B-frame.....	830	Bin (DR).....	527
Background (DR).....	554, 570, 601	Binary black hole merger.....	496
Background video.....	146	bind_address (zmq filter).....	292
bad interpreter: No such file or directory.....	648	Binocular view simulation.....	126
ball.....	181, 219, 263, 284	Bird trails.....	135f.
ball_to_xyz.....	198	bitand.....	124, 420
Band-limited noise.....	497	BitMiracle.....	814
Banding artefacts.....	164	bitor.....	420, 426, 497
Bandpass (FFT filter).....	83	Bitrate.....	334
Bandpass filter.....	83	Bitstream filter.....	193
barrel.....	181	BL.....	733
barrelsplt.....	181	black.....	86
Batch file.....	13	Black hole.....	261
Batch files (DOS, Windows 7 and 10).....	617	Black hole (DR).....	564, 600
Batch files (Unix, Linux).....	620	Black hole distortion.....	201
Bats.....	475	Black hole merger.....	493
bayer_rggb16le.....	373	Black hole simulation with remap filter.....	253
Beamers for fulldome projection.....	661	Black point (DR).....	540
Beauty (DR).....	563	black@0.....	104
Beginning of video.....	86	blackdetect.....	369
bench=start.....	440	blackframe.....	369
		blackhat (morpho filter).....	165
		Blackmagic RAW speed test (DR).....	610
		Blanking Fill (DR).....	563
		blend.....	110, 114, 363, 380, 427f., 444
		Blend filter.....	118
		blend=all_expr.....	119
		blend=all_mode=difference.....	159
		blend=all_mode=grainextract.....	66, 119
		blend=all_mode=lighten.....	381
		blend=all_mode=subtract.....	128, 380f.
		Blender.....	584
		Blink comparator.....	113
		Blinn (DR).....	599
		Blocking artefacts.....	163
		Bluescreen.....	146
		Bluescreen / greenscreen (DR).....	577
		Bluetooth remote (Ricoh Theta).....	713
		Blur (DR).....	571
		Blur images.....	75
		bm3d.....	315, 363
		BMP.....	370
		BOM.....	97
		Boring detector (DR).....	529
		Bourke, Paul.....	135, 169, 186, 656, 663
		box.....	301, 731
		Box Blur (DR).....	562
		boxborderw.....	301, 731
		boxcolor.....	301, 731
		BR.....	733
		Bratwurst.....	117
		Breaking down the script.....	646
		brgp16be.....	319
		brgp16le.....	319
		brightest pixel.....	164
		Brightness.....	27
		brightness (hue filter).....	40
		Brightness gradient.....	164
		Brown dwarf.....	796

BT.2020 (DR).....	559
BT.2020 (GH5S).....	681
buffer_size.....	434
Building FFmpeg.....	647
Built-in help.....	395
Built-in microphone.....	478
Bulk composition (SE).....	795
Byte Order Mark.....	97
C-Mount.....	656, 662
C#.....	240, 293, 767, 826
C# programming.....	811
C# Programming.....	295
c1x6.....	181
c3x2.....	181
C4K.....	654
C4K 10 bit Modes GH5S.....	697
C4K 8 bit Modes GH5S.....	697
c6x1.....	181
Cable Remote Trigger GH5S.....	689
Calculate variables in a batch file.....	619
Calculations in Windows batch files.....	23
Calibrite ColorChecker.....	67
call.....	87
Camera.....	654
Camera / fisheye combinations.....	657
Camera profile.....	175
Camera Shake (DR).....	564
Camera3D (DR).....	582, 598
Cameras and lenses for fulldome video.....	652
CameraTracker (DR).....	581
Canon 5D MK4.....	654, 657, 819
Canon 5D-Mark4.....	666
Canon 5D-Mark4 Field of view.....	668
Canon 5D-Mark4 video modes.....	666
Canon 5D-MK4.....	407, 744
Canon 6D.....	652, 654, 657, 767
Canon 7D.....	654
Canon 8-15mm f/4.....	171
Canon CN-E 24mm T1.5 L F.....	645, 668, 691, 765
Canon CN-E 50mm T1.3 L F.....	645, 668, 691
Canon CN-E 85mm T1.3 L F.....	645, 668, 691
Canon EF.....	655, 662
Canon EF 100-400mm f/4.5-5.6.....	668, 691
Canon EF 100mm f/2.8.....	668, 691
Canon EF 11-24mm f/4.0.....	668, 691
Canon EF 200mm f/2.0.....	668, 691
Canon EF 24-70mm f/4.0.....	668, 691
Canon EF 400mm f/2.8.....	390, 668, 692, 818f.
Canon EF 500mm f/4.0.....	668, 692
Canon EF 8-15mm at 8mm.....	655
Canon EF 8-15mm f/4.0.....	668, 691
Canon EF-M.....	662
Canon EF-S.....	662
Canon EOS R.....	654, 657
Canon FD.....	662
Canon R.....	662
Capture a region of the desktop.....	397
Capture a video from a webcam.....	395
Capture a window.....	397
capture device.....	395
Capture the entire desktop.....	396
Capture the entire desktop with audio.....	396
Capturing HDMI output from GH5S.....	398
Carbon fiber.....	826
Carbonia (SE).....	795
cb_expr.....	227
cd	648
cd (Unix command).....	647
CD Rippers.....	624
ceil.....	82, 420
Center coordinates of a circle.....	240
center-aligned.....	98
Centre Weighted.....	683
cga.....	376
Change audio length, sample rate and/or pitch.....	465
Change audio volume.....	463
Change Clip Speed (DR).....	585, 587
Change the container format.....	17
Change the framerate of a clip (DR).....	534
Change the speed of a clip (DR).....	534
Changing options at runtime.....	450
Changing the size of an animated GIF.....	116
Changing the speed.....	92
channel layouts.....	469
channelmap.....	470f.
channelsplit.....	470
Character Map.....	831
Chessboard.....	359
Chinese battery adapters.....	690
Chip size.....	654
Chirp.....	488
Chirp mass.....	493
chmod +x.....	648
CHROMA.....	331
Chroma subsampling.....	319
chromahold.....	73, 363
chromakey.....	138, 143, 146, 149f., 363
Chromakey.....	143
chromanr.....	315, 363, 444
chromashift.....	74, 363
Chromatic aberration.....	172
Chromatic Aberration (DR).....	563
Chromatic Adaption (DR).....	562
cif.....	376
Cinelike-D.....	684
Cinema advertising.....	375
Cinema advertizing.....	642
Circular mask.....	120, 165, 191, 378
clip.....	28, 120f., 162, 229, 258, 344, 420
Clip Attributes (DR).....	534
clipping.....	473
Clipping GH5S audio.....	751
Clipping TASCAM.....	750

clone.....	647
Clone Tool (IrfanView).....	714
close (morpho filter).....	165
Closed GOP.....	335
CLUT.....	59
cmd.....	13
Col Boost (DR).....	540
cold (SE).....	794
color.....	361
Color Chart overlay (DR).....	558
Color Compressor (DR).....	562
color cube.....	327
Color format.....	389
Color Generator (DR).....	563
Color grading.....	59, 61, 625
Color grading with 3D LUT Creator.....	625
Color look-up table.....	59, 61
Color Managed (DR).....	541
Color matching (DR).....	558
Color page (DR).....	540
Color Palette (DR).....	563
Color Science (DR).....	541
Color Space & Transforms (DR).....	541
Color Space Transform (DR).....	562
Color Stabilizer (DR).....	562
Color table.....	331
Color temperature.....	744
Color temperature test with video lights.....	744
color_primaries.....	374
color_range.....	374
color_trc.....	374
color-look-up-table.....	58
color=black@0.....	104, 731
color=black@0.0.....	308
color=c=black.....	101, 474
color=c=black@0.....	664
colorbalance.....	46f., 363
colorchannelmixer.....	32f., 35, 78, 83, 138, 330,
363, 372, 445	
Colorchannelmixer filter.....	32
colorchart.....	67, 69f.
ColorChecker.....	625
colorcontrast.....	48f., 363
colorcorrect.....	36, 363, 445
Colorcorrect filter.....	36
ColorCorrector (DR).....	571
colorhold.....	72, 364, 445
colorkey.....	138, 142, 146f., 364
Colorkey.....	142
colorlevels.....	29, 99, 381, 388f., 393, 445
colormap.....	69ff., 358, 445
colormatrix.....	374
Colors of nodes (DR).....	537
colorspace.....	374
colorspectrum.....	345, 347, 349, 351, 446, 665
colortemperature.....	38, 364
Colortemperature filter.....	38
colr atom.....	606
Combining keys (DR).....	542
Comet.....	772
Command key (Mac).....	582
Command prompt.....	618
Command_Promt.....	13
commands.....	286
comments.....	13
Comments (Guide 9.1).....	774
compensationdelay.....	484
Compiling FFmpeg.....	647
Complementary colors.....	30
concat.....	93f., 106
Concat demuxer.....	106
Concat filter.....	108
Concentric waves (DR).....	564
config.ini (Stellarium).....	780
Console window.....	13, 295
Console window position.....	158
Constant rate factor.....	334
Constant Rate Factor.....	830
Container format.....	17
Continuous spectrum.....	498
Contrast.....	27
Contrast (DR).....	540
Contrast enhancement (DR).....	562
Contrast Pop (DR).....	562
Control Voltage.....	759
Convert.....	17
Convert *.mkv videos for DR.....	605
Convert 10-bit videos from GH5S (DR).....	606
Convert a *.cube LUT to a halclut file.....	66
Convert a clip to black and white (DR).....	567
Convert an audio waveform to a picture.....	500
Convert HSV to RGB.....	332
Convert RGB to HSV or HSL.....	331
Convert ultrasound.....	475
Convert video formats.....	17
convolve.....	446
cool (SE).....	794
Coordinate system (Ambisonics).....	729
CoordinateSpace (DR).....	594, 597
copy (video filter).....	338
copy /l /y.....	294
Copy a Grading (DR).....	541
Copy and paste.....	13
Copy attributes (DR).....	533
copy metadata.....	338
Copy planes.....	34
Copy the color grading (DR).....	541
Copy the red channel to the green and blue channels.....	33
corner pin (DR).....	556
Correcting Lens Distortion (DR).....	602
correction_method.....	44
Corrector nodes in "Color" page (DR).....	542
cos.....	78, 420, 492

Could not open lock file (Ubuntu).....	647
Count the number of frames.....	514
Couple one parameter to another (DR).....	601
Cover a circular area.....	715f.
Cover Art.....	408
cover_rect.....	366
CPU Load (OBS).....	636
cr_expr.....	227
CR2 (DR).....	527
CR2 images.....	372, 744
Crab Pulsar.....	821
create a concat_list.....	107
Create short text files.....	619
Creating animated GIF.....	115
Credit files.....	404
Credits file.....	98
CRF.....	334, 830
crop.....	22, 77, 132, 139, 146, 162, 229, 249, 288, 300, 364, 372, 378, 381, 389, 776
Crop.....	90
Crop factor.....	174, 177
Cross correlation.....	446
Cross Dissolve (DR).....	561
Crossfading.....	19, 89, 462
Crossfading (DR).....	561
Crosshair.....	246
CRT Screen Effect.....	163
CS-Mount.....	656, 662
CTRL G.....	618
cubemap.....	798
cue.....	446
Curved text.....	101
curves.....	29, 380, 446
Cut page (DR).....	529
Cutting on the beat (DR).....	547
CV.....	759
Cyclic swapping of the RGB channels.....	32
cylindrical.....	181, 198, 203
cylindrical_to_xyz.....	198
cylindricalea.....	181
DAR.....	440, 830
Dark video.....	380
darken (blend filter).....	118
Darkframe.....	128
Darktable.....	175, 372
Darktable (Plugin for GIMP).....	372
datascope.....	160
DaVinci Resolve.....	830
DaVinci Resolve	518
DaVinci Resolve Project Server.....	518
DaVinci YRGB Color Managed (DR).....	541, 559
DAW.....	759
Day-for-Night.....	35
Day-for-Night (DR).....	546
day-to-night.....	767
day/night.....	239
dblur.....	76
DCF77.....	824
dcf77noiz.....	364
DCP.....	642
dctdnoiz.....	315
DCTL (DR).....	562
Dead Pixel Fixer (DR).....	563
Deadcat.....	507, 509
deband.....	446
Deband (DR).....	563
Deblock.....	163
Debugging of expressions.....	425
decay.....	129
Decoder Time Stamp.....	830
Deep-Zoom-In.....	179
DeepSkyStacker.....	819
DeepSkyStacker 4.2.3.....	803
Defect film projector.....	162
deflicker.....	129, 131f., 364, 385, 447
Deflicker (DR).....	549, 563
Deflicker a video.....	131
Dehaze (DR).....	562
Deinstance (DR).....	538
del.....	294
DeLaHunt, Jim.....	108
Delay by x frames.....	92
Delay by x seconds.....	92
Delay for RGB planes.....	140
delay frames.....	139
Delete clip, leave gap (DR).....	531
Delete gap (DR).....	532
Deliver only one frame (DR).....	545
Deliver page (DR).....	545
Deliver resolution (DR).....	545
delogo.....	366, 714
Delta Keyer (DR).....	577
DeltaKeyer (DR).....	555
denoising.....	315
Dent (Dnt) (DR).....	600
Dent (DR).....	564, 600
derain.....	447
descent (drawtext filter).....	98
desertic (SE).....	794
deshake.....	298f., 364
despill.....	147, 149f., 364, 447
despill, recommended settings.....	158
Detect black frames.....	369
Deviation angle.....	201
Device Manager.....	636, 641
dfisheye.....	181, 189, 215, 301, 663
dfisheye_to_xyz.....	197
Diagonal plane (RGB cube).....	328
Diagonal Size.....	686
Diameter of covered circular area.....	716
difference (blend filter).....	118
Difference between two images.....	119
different durations.....	21
Digital Audio Workstation.....	759

Digital Cinema Package	642	dnxhr_sq	606	Dual Native ISO Settings GH5S	684
Digital Light Processing	830	Documentation	10	Duggan-Smith, Tony	752
Digital maps	811	Dome	210	Duplicate clip (DR)	532
Digital Projection	660f.	Dome diameter	653	DVD Rippers	624
dilate (morpho filter)	165	Doppler effect	821	DVR	830
dilation	164, 364	Doran, Seán	817	DVX200	680
Directional blur	76	double backslashes	608	Dwarf moon	797
Directional Blur (DR)	562	Double equirectangular test image	663f.	Dynamic keyframe (DR)	542
Dirt removal (DR)	563	double fisheye	197, 203	Dynamic Range	830
Disable the overlays (GH5S)	690	Double fisheye test image	663	Dynamic trim mode (DR)	533
Disc viewport (Stellarium)	781	Double quotation mark	291	Dynamic Trim Mode (DR)	522
displace	310ff., 364	double quotes	481	Dynamic Zoom (DR)	546
Displace (DR)	571	Double quotes	291	Dynamic Zoom Ease (DR)	546
Display Aspect Ratio	830	Double speed	92	Dynamically change commands	286
Display Capture (OBS)	634	Double-circular mask	215	dynamically changing size	105
Distance (Beamer)	660	Double-clicking on script files (Unix)	648	e	181
Distorted screen	308	double-fisheye	215	E-Vision Laser 11000 4K-UHD	661
Distortion (Little planet)	720	Double-fisheye	231	eac	181
DLP	830	Download FFmpeg	10	EBU Color Bar (DR)	562
DLP Beamer output	214	Downloading videos from Facebook	429	echo (Windows batch file)	618f.
DMD chip	661	Downloading videos from Youtube	430	Ecliptic	816
DMD size	661	DR	830	Edge Attenuation	125
DMW-AC10E	690	drag-and-drop	13	Edge Detect (DR)	563
DNG	370, 447, 616	Dragonfly project	818	Edit page (DR)	530
DNG (DR)	527	draw (option of showwaves filter)	501	Editor	98
DNG Images	372	drawbox	125, 223f., 235f., 245, 304, 364, 366, 389, 393, 447, 715		
DNG test image	374	drawellipse	448	Effects Library (DR)	561
dnxhd	606	drawgraph	389, 393, 448	EFhex BBhex BFhex	97
DNxHD	298, 606f.	drawgrid	77, 359, 664f.	ega	376
DNxHR 444	607	drawtext	.96, 98ff., 103f., 301, 354, 388, 664, 731	Eingabeaufforderung	13
DNxHR HQ	607	drawtext reinit	301	Einstein-Rosen Bridge	259
DNxHR HQX	607	Drone	742	Electronic shutter GH5S	687
DNxHR LB	607	Drone camera (DR)	581	Elevation data	811
DNxHR SQ	607	Drone videos (DR)	549	Ellipse (DR)	601
dnxhr_444	606	Drop Shadow (DR)	563	Ellipsoid distortion effect	200
dnxhr_hq	606	dshow	395	Embedding date and time	339
dnxhr_hqx	606	DTS	830	Emboss (DR)	564
dnxhr_lb	606			Enable or disable a node (DR)	542
				enable=	409, 442

Encoding speed (FFmpeg).....	432	Ethernet.....	433
Enlarge the corners of a live video.....	247	ETTL.....	830
enlarging eyes (DR).....	564	ETTR.....	676, 830
Enlarging the eyes.....	312	EV.....	830
Entaniya HAL200.....	655	eval=frame.....	105, 284, 508
Entaniya HAL250.....	655	Evaluation of mathematical functions.....	423
Entire Timeline (DR).....	545	Ex. Tele Conv.....	390, 690, 818
ENV.....	759	Exact Audio Copy.....	624
Envelope.....	759	Examine a file.....	513
Environment variable.....	830	Exchange clips (DR).....	531
Environment Variables.....	14	exchanging the red and green components.....	33
Ephemerides.....	773	Exclude two regions from a mask (DR).....	567
eq.....	26ff., 30, 53, 364, 381, 420, 448	Execute a script file.....	648
equidistant.....	183, 197, 202	Execution time.....	440
equinox.....	239	EXIF.....	337, 613
Equipment.....	507	exiftool.....	714ff., 727
equirect.....	181	Exiftool.....	429, 613
equirect_to_xyz.....	197	exiftool -u.....	613
equirectangular...190, 192, 197, 202, 215f., 218, 225, 449, 711		exit (Windows batch file).....	619
Equirectangular.....	449	exp.....	420
Equirectangular images of night sky....184, 780		Explosion effect (DR).....	575
Equirectangular panorama.....	717	Export a Fusion composition (DR).....	537
Equirectangular test image.....	663	Export Project Archive (DR).....	528
Equirectangular test pattern.....	663	Export Skybox (SE).....	798
Equirectangular to Angular (DR).....	593	Export Textures (SE).....	797
Equirectangular to DomeMaster (DR).....	593	Expose To The Left.....	830
Equirectangular to Fisheye (DR).....	593	Expose To The Right.....	830
Equirectangular to fisheye conversion.....	195	Exposing For Middle Gray.....	677
Equirectangular to InverseAngular (DR).....	593	Exposing to the right.....	676
Equirectangular to MeshUV (DR).....	593	exposure (FFmpeg filter).....	29, 449
equisolid.....	182f., 199, 205	Exposure compensation.....	677
equisolid_to_xyz.....	199	Expression (DR).....	601
erode (morpho filter).....	165	Expression evaluation.....	420, 449
Error handling in expressions.....	424	Extended desktop.....	517
escape commas.....	114	External HDMI (GH5S).....	690
Escaping.....	172, 608, 627	Extra Tele Conversion.....	830
ETC.....	830	extract a picture.....	626
		Extract a time segment.....	84
		Extract and merge planes.....	140
		Extract Images.....	25
		Extract many images.....	25
		Extract moving objects from a video.....	159
		Extract the alpha channel.....	141
		Extract the first frame.....	26
		Extract the last 30 seconds.....	87
		Extract the last frame.....	26
		Extract the N_th frame.....	26
		extractplanes.....	53, 138, 140, 364
		extractplanes=a.....	141
		Extracts each n_th frame.....	25
		Eyepiece.....	120
		Eyes.....	312
		f-stops.....	643
		F1-F12 keys.....	807
		Facebook.....	429f., 474, 712
		Facebook Live - Mode "Camera" (OBS).....	637
		Facebook Live - Mode "Connect" (OBS).....	638
		Facebook Live (OBS).....	637
		fade.....	88f., 378, 385, 450
		Fade a text in and out.....	96
		fade-in.....	404
		Fade-in.....	88
		fade-out.....	404
		Fade-out.....	88
		Fairlight --> Input_Monitor_Style (DR).....	589
		Fairlight room (DR).....	588
		fast.....	334
		fastdecode.....	334
		faster.....	334
		Faster, Pussycat! Kill! Kill!.....	506
		Fastnoise (DR).....	570
		Faststone Image Viewer.....	616
		FAT32.....	742
		fb.....	181
		fc.....	744
		FC.....	733

feedback.....	367	filter_complex_script.....	451
Feria (SE).....	795	Find an object in a video and hide it.....	366
ffmetadata.....	337	Find objects from Tycho catalog.....	779
FFmpeg in detail.....	17	Find the coordinates of a moving object.....	368
FFmpeg source code.....	441	Find the keyframe timestamps.....	514
ffmpeg-all.html.....	10, 450	find_rect.....	282f., 366, 368, 451
ffmpeg.exe.....	10	findstr.....	613
ffplay. 149, 236, 317, 435, 482, 492, 497, 725, 736		Fine adjustment of shutter speed (GH5S)....	690
FFplay.....	436f., 479, 481	Fireflies.....	827
FFplay options.....	516	firequalizer.....	452, 496
ffplay-all.htm.....	450	fisheye.....	181, 190, 197, 202
ffplay.exe.....	10	Fisheye.....	222, 654, 711
FFprobe.....	368, 513	Fisheye lens (distortion).....	168
ffprobe.exe.....	10	Fisheye lenses.....	655
ffprobe.xsd.....	369	Fisheye projection lenses.....	659
FFT filter.....	482	Fisheye projections.....	183
FFT Filtering.....	77	Fisheye to Equirectangular (DR).....	593
fftdenoiz.....	315	fisheye_to_xyz.....	197
fftdnoiz.....	364	Fit length to music.....	18
fftfilt.....	79f., 82f., 364, 451	Fit to Fill (DR).....	531
FFV1.....	333	FITS.....	370
FHD 10 bit Modes GH5S.....	702	Fitswork.....	802, 813
FHD 8 bit Modes GH5S.....	701	FL.....	733
Field of view.....	190	flags=.....	160
Field of view (Full Frame).....	691	flags=neighbor (scale filter).....	77
Field of view (GH5S).....	645, 668, 691	Flange distances.....	662
File --> Media_Management (DR).....	605	flat.....	181
File --> Project_Settings (DR).....	526	Flat video in fulldome.....	235
Filenames for images.....	339	flat_to_xyz.....	197
fill (remap option).....	186, 216	Flicker Addition (DR).....	564
fillborders.....	451	Flicker-free calculator.....	690
film.....	334	Flickering.....	690
film (video size).....	376	floodfill.....	452
Film Damage (DR).....	561, 564	floor.....	359, 420
Film Gate (DR).....	581	Fluorescent light flicker.....	132
Film Grain (DR).....	564	Fluoro Light deflicker (DR).....	131
Filter curves.....	622	Fog Effect (DR).....	574
Filter type.....	643	fontcolor.....	96, 301
		fontcolor_expr.....	97
		fontfile.....	96
		fontsize.....	96, 301
		Foot-candle.....	744
		for.....	87
		force_style.....	403
		Foreground video.....	146
		format.....	128, 381
		Format negotiation.....	104, 325
		format=argb.....	120, 191, 262, 308
		format=bgra.....	192, 725
		format=gbrp.....	34, 130, 229f.
		format=gray.....	161
		format=gray8.....	428
		format=pix_fmts=gray8.....	310
		format=pix_fmts=rgb24.....	310
		format=rgb.....	284
		format=rgb24.....	227, 394
		format=rgba.....	104, 664, 731
		format=yuv422p10le.....	325, 366
		Formula for VLog curve.....	677
		Forward enlarging effect.....	196
		forward slashes.....	608
		FOTGA DP500/A70TLS monitor.....	63
		Fountain.....	646
		Four Color Gradient (DR).....	562
		fps.....	22, 108, 193, 249, 275, 385
		FPS.....	830
		fps (console output).....	440
		FR.....	92, 733
		Frame Display Mode (DR).....	527
		frame_tags=lavfi.rect.x.....	283, 368
		frame=pkt_pts_time.....	283, 368
		framerate.....	19, 385
		Framerate.....	92, 386, 511
		Framerate (DR).....	525
		framestep.....	25
		framesync.....	186

Franklin, Paul.....	253	Generate a LUT (DR).....	559
Free CD to MP3 converter.....	624	Generate LUT (DR).....	543
freezeframes.....	117	Geometric distortion.....	172
frei0r.....	406	geq 28, 30, 78, 161, 186, 190, 221, 227, 229, 262, 340, 344, 353, 359, 361, 364, 422, 428, 452	
Frequencies and control voltages of the keys	757	geq=a=.....	120, 141, 262
Frequency domain.....	486	geq=lum=.....	131
Frequency response.....	496	GFX.....	830
Frequency response of a filter.....	498	GH5S.....	674, 764f.
Freudenberg, Michael.....	752	GH5S (Serial number).....	613
frigid.....	794	GH5S Autofocus.....	674
frigid (SE).....	794	GH5S Custom settings C1, C2, C3.....	685
Fujinon 1.8mm.....	656	GH5S Exposing for VLog-L.....	676
Fujinon 2.7mm.....	656	GH5S Field of view.....	691
Full HD.....	654	GH5S HLG (Hybrid Log Gamma).....	681
Full path.....	608	GH5S Metering Modes.....	683
Fulldome test patterns.....	663	GIF.....	370
Fullscreen mode (DR).....	522	Gimbal.....	764
Fundamental wave.....	489	Gimp.....	175, 616, 813
Fur windshield.....	507, 509	GIMP.....	58, 372
Fusion Clips (DR).....	539	git.....	647
Fusion page (DR).....	537	git add.....	650
fwqvga.....	376	git clone http://git.ffmpeg.org/ffmpeg.git.....	650
FX.....	830	git commit.....	649
FX effect (DR).....	543	git commit -a.....	650
Gaia2 catalog.....	778	git format-patch HEAD^.....	649
Gain (DR).....	540	git reset.....	650
Galaxy.....	356	git reset --hard HEAD^.....	650
Gamepad tester.....	789	git reset --hard origin/master.....	650
Gamma.....	27	git reset HEAD^.....	649
Gamma (DR).....	540	Glow (DR).....	563
Gamma LUT.....	65	Glowworms.....	827
Gamut Limiter (DR).....	562	gnomonic.....	181
Gamut Mapping (DR).....	563	GOP.....	335, 830
Gaussian Blur (DR).....	562	GoPro VR Player 3.....	727
gblur.....	367, 452	goto.....	87
gbrp.....	140, 183, 229, 319	goto :eof.....	619
gdigrab.....	192, 396, 452	goto (Windows batch file).....	618
		GPU.....	438
		Grab Still (DR).....	541, 543
		Gradation curves.....	58
		gradfun.....	364
		Gradfun.....	164
		Gradient.....	340
		gradient (morpho filter).....	165
		Gradient to black.....	346, 350
		Gradient to white.....	348, 350
		gradients.....	341, 344, 352, 453, 665
		Gradual ramp.....	94
		grain.....	334
		grainextract (blend filter).....	118
		Graphical effects.....	830
		Graphical user interface.....	13
		graphmonitor.....	325
		Gravitational Lensing.....	253
		Gravitational wave.....	493
		Gravitational waves from binary black hole mergers.....	493
		Gravity constant.....	493
		Gravity labels (Stellarium).....	781
		gray.....	319
		Gray card (DR).....	541
		gray14be.....	319
		gray14le.....	319
		gray16be.....	319
		gray16le.....	319
		gray8 (geq filter).....	422
		grayf32.....	370
		grayf32be.....	319
		grayf32le.....	319
		Grayscale (DR).....	561
		Grayscale test image.....	360
		Great red spot.....	775
		Greatest common divisor.....	510
		green.....	341, 343
		Green screen.....	146

Green screen (OBS).....	641	hd480.....	376	Horizontal stripes.....	690
Grey Scale (DR).....	562	hd720.....	375, 377	Horror Machine.....	752
Grid (DR).....	563	HDMI Audio capture.....	398	hot (SE).....	794
Grid of dots.....	359	HDMI to USB converter.149, 157, 244, 246, 398,		hours:minutes:seconds.milliseconds.....	99
Grid video.....	359	690		hqdn3d.....	315, 364
Grossgrade.....	629	HDR.....	830	hqvga.....	376
Group (OBS).....	641	he.....	182	HSB.....	830
Group of Pictures.....	335, 830	Height over ground correction.....	200f.	HSL.....	331
gsettings.....	648	Hemistar.....	659	hstack....22, 78, 83, 111, 215, 249, 299, 301, 330,	
gsxga.....	377	hequirect.....	182	364, 506, 627, 663, 731	
gt.....	120f., 420	HEVC.....	830	HSV.....	73, 331, 830
gte.....	344, 420	hevc_nvenc.....	438	hsvhold.....	73, 364, 453
GUI Controls (DR).....	523	Hex editor.....	97	hsvkey.....	144, 146f., 364, 453
Guide 9.1.....	771	Hex Editor MX.....	97	HSVkey.....	144
h_offset (v360 filter).....	209	hflip.....	111	hsxga.....	377
h.264.....	438	hidden files or folders (Unix).....	648	hue.....	27, 40, 364, 453
H.264 (DR).....	605	Hide something from a 360° image.....	714	Hue.....	27
H.265 (DR).....	605	hide taskbar.....	236	HUE.....	331
h.265/HEVC.....	438	High Dynamic Range.....	830	Hue (DR).....	540
h264.....	675	High Efficiency Video Coding.....	830	Hue-Saturation-Brightness.....	830
H264.....	333	Highlight Weighted.....	683	Hue-Saturation-Value.....	73, 830
H264 lossless.....	333	highpass.....	473, 475, 477, 482	huesaturation.....	42, 364
h264_nvenc.....	438	Highpass (FFT filter).....	83	Huesaturation filter.....	42
h265.....	17, 675	Highpass filter.....	83	Hugh Hou.....	703
H265.....	333	Hign pass filter.....	473	Hurdy Gurdy Wheel.....	752
Habitable zone.....	822	Histogram.....	128, 676	hvga.....	376
Hackathon 2022.....	264	HL (DR).....	540	hwdownload.....	439
halclut.....	61, 66, 71	HLG.....	681, 830	hwupload.....	439
halclutsrc.....	60f., 66, 71, 383, 427	HLG (DR).....	558	hypot....83, 120, 169, 190, 216ff., 223, 229, 249,	
Half Float.....	370	HLG (Hybrid Log Gamma).....	684	258, 262, 353, 420	
Half speed.....	92	HLG to Rec709 LUT.....	681	hysteresis.....	454
hammer.....	181	Horizon.....	820	I-frame.....	830
Hardware acceleration.....	438	Horizon file.....	778	Identity *.cube LUT.....	64
Hardware Acceleration.....	439	Horizon line.....	776	identity map (remap filter).....	193
Hardware Decoding (DR).....	605	Horizontal 360° rotation.....	718	IDS.....	636
Harz Mountains.....	210, 811	Horizontal flipping.....	111	if.....	350, 420
Hazelden, Andrew.....	592	Horizontal slices.....	139	if (in batch file).....	620
hd1080.....	375, 377	horizontal stripes.....	132	if %MODE%==1.....	620

ifnot.....	420, 425	Interstellar (Film).....	259
Illuminance.....	744f., 819	Interval.....	763
Im Lauf der Zeit.....	35	Interviewing two people.....	642
Image circle diameter.....	655	intraframe.....	109
Image formats.....	370	Intraframe.....	830
Image processing devices.....	636	Intraframe compression.....	830
Image size limits.....	371	Introduction to FFmpeg.....	10, 17
Image warping with displace filter.....	310	invalid result.....	424
Image_Width (Beamer).....	660	Inverse function.....	170
Immersive 360° videos.....	703	Invert channels.....	30
Import a Fusion composition (DR).....	537	Invert Color (DR).....	563
Import a timeline (DR).....	528	Inverting a video.....	30
Impulse Response.....	830	Invisible Lattice Mast.....	826
in_time.....	91	Invisible tripod.....	600
In/Out Range (DR).....	545	IPB.....	667, 830
Ina (moon crater).....	816	ipconfig.....	433
inf.....	424	IPv4 Address.....	433
Infinite loop.....	425	IR.....	830
Input Color Space (DR).....	559	IRE.....	676, 678, 831
Input Gamma (DR).....	559	IrfanView.....	372, 615
Input offset correction.....	200	IrfanView (Clone Tool).....	714
Input_Monitor_Style (DR).....	589	Irradiance.....	819
Insert a crosshair in a live video.....	246	Irregular mare patches.....	816
Insert clip (DR).....	530	Irregular quadrangle.....	308
Insert gap (DR).....	531	ISCO-Optic 1:2 21mm.....	659
Insert text.....	96	ISCO-Optic 1:4 14.3mm.....	659
inspect pixel values (Gimp).....	616	isinf.....	425
Insta360 GO 2.....	739	isnan.....	425
Insta360 ONE X2.....	703	Isolate Humans (DR).....	556
Insta360 Studio.....	739	it.....	91
Instances (DR).....	538	iZugar MKX200-ASPH 3.8mm.....	656
Integer arithmetic in Windows batch file.....	23	iZugar MKX22 3.25mm.....	656
interframe.....	109	James, Oliver.....	253
Interframe compression.....	830	Jitter.....	161
Intermediate video files.....	333	JJRC X17 drone.....	743
Internal Serial Number (ExifTool).....	613	JJRC X17 Drone.....	742
interp=near (v360).....	192	Joby Impulse.....	713
Interpolation band.....	230	join.....	470f., 508, 736f.
		joystick.....	809
		Joystick (SE).....	788
		Joystick tester.....	789
		JPEG Damage (DR).....	564
		JPG.....	370
		JPG Test images.....	358
		Judder.....	645
		Jupiter (SE).....	795
		KartaVR (DR).....	592
		Kerr black hole.....	253
		Keyboard.....	756
		Keyboard customization (DR).....	522
		Keyboard shortcuts (DR).....	521f.
		Keyboard shortcuts (SE).....	783
		Keyframe.....	335, 830
		Keyframe (DR).....	542
		Keyframes (DR).....	536
		Keypress while FFmpeg is running.....	288
		Kings of the Road.....	35
		Kit Scenarist.....	646
		Kodak PixPro SP360 4K.....	268
		Kodak PIXPRO SP360 4K camera.....	136, 187, 705
		Kodak SP360_4K.....	379, 652
		Korven, Mark.....	752
		kvrReframe360Ultra.....	592
		L Lower 3rd (DR).....	561
		L4500 video lights.....	744
		Iacustrine (SE).....	794
		lagfun.....	134f., 140, 364, 454
		Lagfun filter.....	129
		LAN.....	433
		LAN cable.....	640
		language=ger.....	403
		Laowa 24mm f/14.....	668, 691
		Laowa 4mm.....	171, 655
		Last message repeated n times.....	423, 426
		Latitude.....	812
		Lattice mast.....	826

Layer (DR).....	539	Limit the length.....	113
Id.....	161f., 169, 258, 310, 421, 489	limited BT.601.....	326
LD IOCA SUB18A.....	496	Limited range RGB.....	326
Leica DG 12-60mm f/2.8-4.0.....	613, 645, 691, 764f.	limiter (FFmpeg filter).....	29
Lens Blur (DR).....	562	Limiting magnitude.....	818f.
Lens correction models.....	175, 602	LINE GH5S.....	751
Lens distortion.....	176	Line of sight.....	811
Lens distortion (DR).....	602	Linear chirp.....	486, 489
Lens Distortion (DR).....	564	Linear waves (DR).....	564
Lens Flare (DR).....	563	Linearization function.....	374
Lens profile.....	175	Link Clips (DR).....	535
Lens Reflections (DR).....	563	Linked Selection (DR).....	535
Lens Serial Number (ExifTool).....	613	Linux.....	10
lenscorrection.....	168, 364	List of all pixel formats.....	320
LensDistort (DR).....	175, 602	List R,G,B values as ASCII text.....	427
Lensfun.....	172	Little planet.....	216, 261
Lensfun database.....	172, 175f.	Little Planet (DR).....	594
lerp.....	123, 227, 258, 288, 330, 340, 421f., 449	Little-Planet.....	719
Let system position window.....	158	Live broadcasting.....	640
Letterbox (DR).....	555	Live rotation of the moon.....	244
LFE.....	733	Live Save (DR).....	524
LFO.....	759	Live Streaming with Ricoh Theta V.....	723
LHS.....	831	Live ultrasound conversion.....	482
libavutil.....	319	log.....	82, 421
Libration.....	779	Logarithmic chirp.....	486, 491
libswscale.....	325	long persistance time.....	129
LibTiff.....	814	Longer Exposure Time GH5S.....	687
libx264.....	333	Longitude.....	812
libxvid.....	333	loop.....	150, 187, 517
Licence.....	485	lossless.....	370
Lift (DR).....	540	Lossless H264 codec.....	333
Light curve.....	388, 390	Lossless video codec.....	333
Light Rays (DR).....	563	Low Frequency Oscillator.....	759
light value.....	767	Low-noise.....	507
lighten (blend filter).....	118	lowpass.....	475, 482, 498, 736
Lightning.....	743	Lowpass (FFT filter).....	83
Lightning bugs.....	827	Lowpass filter.....	79f., 83
lime.....	341, 343	lt.....	344, 353, 421, 490
		lte.....	421
		Lum Mix (DR).....	540
		lum_expr.....	227
		lumakey.....	145, 364
		Lumakey.....	145
		Luminance Level GH5S.....	684, 686
		LUT.....	66, 626, 831
		LUT (DR).....	559
		LUT (GH5S).....	681
		LUT (OBS).....	636
		LUT_3D_SIZE.....	63, 356
		lut3d.....	66, 357, 607, 627
		lut3d=\%LUT%\'	608
		lutyuv.....	28
		lux.....	744
		Lux.....	819
		M Lower 3rd (DR).....	561
		M42.....	662
		Macroblock motion vectors.....	163
		mag.....	819
		Magic Mask (DR).....	556
		Magnetic compass calibration.....	742
		Magnitude.....	819
		Main sequence star.....	796
		make.....	647
		Make a Unix script executable.....	648
		MakeMKV.....	624
		Manley, Scott.....	259
		Manual GH5S.....	674
		map_metadata.....	333, 608
		Mapping streams to the output.....	406
		Mare Crisium.....	242f.
		Mare Marginis.....	243
		Mare Smythii.....	243
		marine (SE).....	794
		Marker (DR).....	547
		maskedmerge.....	159, 219ff., 229f., 239, 715f.
		maskfun.....	454

Mass prefix (SE).....	794	Microphone.....	478, 482	Mount.....	655
Master Pedestal Level GH5S.....	686	Microphone cable.....	507	Mountainbike.....	305
Matrixbrute.....	757	Middle gray.....	677	Mouse coordinates (IrfanView).....	615
Matrixbrute control voltage.....	758	Midi Control Center.....	759	Mouse pointer.....	635
MatrixBrute Control Voltages I/O.....	760	Midi controllers.....	758	MOV.....	675
MatrixBrute power saving mode.....	760	MIDI number.....	758	mov_text.....	403
Matte (DR).....	577	min.....	126, 421	movie.....	149, 283, 368
Matyas, Eric.....	485	minterpolate.....	387	Moving a clip (DR).....	532
max.....	421	Mirror sphere.....	218	Moving black hole.....	258
max_colors.....	115	Mirror sphere (DR).....	598	Moving sound source.....	732
MB/min.....	689	Mirror-sphere video.....	261	Moving wormhole.....	282
MB/s.....	689	Mirrors (DR).....	564	MP4.....	675
MCC.....	759	mirrorsphere.....	198f., 204f.	MP4 (LPCM).....	675
MD (DR).....	540	mix.....	454	MP4 codec.....	333
measure RGB colors.....	160	Mixing frequency.....	475	MP4 HEVC.....	675
Measuring audio volume.....	499	MJPEG.....	667	Mpeg TS system clock timebase.....	511
Mechanical / Electronic Shutter GH5S.....	687	mklink.....	294	mpeg4.....	333, 454
Media offline (DR).....	528	mod.....	22, 99, 161, 249, 421, 490	mpv.....	622
Media offline (error message in DR).....	610	modify alpha.....	158	MPV.....	622
Media page (DR).....	527	Monitor calibration.....	624	mpv.com.....	622
Medialn (DR).....	539, 550, 554f.	monochrome.....	50, 365, 454	mpv.exe.....	622
medium.....	334	Monochrome filter.....	50	MSVCP120.dll.....	628
Meike 3.5mm.....	171, 655	Monochrome picture.....	474	MSVCR120.dll.....	628
Meike 6-11mm.....	655	Moon.....	240, 244	MTF.....	692, 831
Meike 6-11mm f/3.5.....	668, 691	Moon libration.....	244, 779	Multicam Editing (DR).....	603
Meike 6.5mm.....	171, 655	Moon map.....	816	Multicast addresses.....	435
Meike 8mm.....	655	Moon observing.....	816	Multiple audio streams.....	472
mercator.....	181	Moon phase.....	816	Multiple outputs.....	406
Merge (DR).....	554, 572, 601	Moon surface.....	817	Multiple partitions.....	763
Merge3D (DR).....	598	Moon videos.....	817	Multiple playheads (DR).....	542
mergeplanes.....	140	morpho.....	165, 167	multiply.....	122ff.
Metadata.....	337	Morphological transforms.....	165	multiply (blend filter).....	118
Metadata (DR).....	581, 609	Mosaic Blur (DR).....	553, 562	multithreading.....	426
Meteors.....	87, 129	Mosquito.....	732	Music.....	473
Meyer, Russ.....	506	Motion Estimation (DR).....	585	Mute (DR).....	588, 590
MFT.....	655, 662	Motion interpolation.....	387	Mute the speakers (DR).....	523
Mic_Socket.....	751	Motion interpolation (DR).....	585	mv (Unix command).....	648
Microbrute.....	757	Motionless mark in space.....	826	n (option of showwaves filter).....	501

nan.....	424, 452	non-mapped pixels.....	202	Olympus M.Zuiko 8-25mm f/4.0.....	645, 691
Nature sounds.....	473, 811	non-monotonically increasing dts.....	193	Olympus M.Zuiko 8mm.....	655
Navigating in the 3D viewer (DR).....	582	Non-printable character.....	97	Olympus M.Zuiko 8mm f/1.8.....	691
Navitar.....	659f.	Nonlinear fisheye.....	197, 203	Olympus Mons.....	799
ND.....	643	normal (blend filter).....	118	One Shot Color.....	831
NDI.....	266	Normalize Audio (DR).....	536	one-time events.....	317
Needle impulse.....	490	North-south pole image.....	665	opacity (blend filter).....	118, 444
Needle impulses.....	498	Nose.....	310	open (morpho filter).....	165
negate.....	30, 135, 301, 365, 663, 731	Notch (FFT filter).....	83	Open Broadcaster Software.....	634
Negative.....	30	Notch filter.....	83	Open FX Overlay (DR).....	577
Neptune (SE).....	795	Notepad.....	97f.	Open GOP.....	335
Net.....	327	Notepad++.....	98	Open-source.....	10
NetMQ.....	286, 296	Notepad++ open a new empty file.....	98	OpenCL.....	439
NetMQ library.....	293	NP-F970 battery.....	744	OpenFX (DR).....	561
Neutral Density Filters.....	643	NR.....	831	OpenFX Overlay (DR).....	564
Neutron star.....	821	NT1.....	509	OpenShot.....	612
New Bin (DR).....	527	NTG2.....	509	OpenTopoMap.....	812
New filters.....	454	ntsc.....	375f.	Opteka 6.5mm	655
New Fusion Clip (DR).....	539	ntsc-film (Video size).....	376	optical axis of beamer.....	214
New Fusion Composition (DR).....	598	NUL.....	499	Optical Density.....	643
New Project (DR).....	527	null.....	105, 365	Optical Flow (DR).....	534, 585
nhd.....	376	null (video filter).....	338	Optical sound effect.....	506
Night scenes.....	35	nullsrc.....	190, 229, 249, 262, 310, 334, 341, 455		
Night sky videos with GH5S.....	383	Number of frames.....	514	Option key (Mac).....	582
night-to-day timelapse.....	767	Number of Pixels GH5S.....	686	or (blend filter).....	118
Nikon D800.....	654, 658	NuView.....	660	Order of rotations.....	185
Nikon F.....	662	Nvidia.....	396	orthographic.....	183, 199, 206
Nippon Kogaku 8mm.....	655	NVIDIA (DR).....	545	orthographic_to_xyz.....	199
Nippon Kogaku 8mm f/2.8.....	668, 685, 691	Nvidia GPU.....	438	OS-X.....	10
Nippon Kogaku 8mm Fisheye.....	383	OBS.....	634	OSC.....	831
nlmeans.....	315, 365	OBS Studio.....	723	OSCILLATOR.....	750
Node Label (DR).....	542	OBS Virtual Camera.....	395, 641	Oscillator tuning (MatrixBrute).....	760
noise.....	360	oceanic (SE).....	794	oscilloscope.....	341, 347, 394
Noise generator (video).....	360	octahedron.....	182	Oscilloscope.....	501
Noise map.....	815	Off-center fisheye projection.....	209	ot.....	91
Noise reduction.....	315	Offset (DR).....	540	out_time.....	91
Noise reduction (DR).....	586	og.....	182	Output offset correction.....	202
Noise video.....	360	Ohme, Frank.....	493	Overlapping fisheye videos.....	229
				overlay.....	104f., 116f., 134f., 146, 149, 191, 220,

224, 236, 239, 262f., 284, 289, 308, 366, 378, 385, 389, 393f., 455, 664, 731	
Overlay (Guide 9.1).....	774
Overlay an animated GIF.....	116
Overlay picture (DR).....	555
Overlay text (DR).....	557
overlay=format=yuv422p10.....	120
Overlays (GH5S).....	690
Overtone.....	489
Overwrite (DR).....	530
owdenoise.....	315, 365
P-frame.....	830
packed.....	319
pad.....	90f., 189, 308, 381, 776
pal.....	375f.
palettegen.....	115f.
paletteuse.....	115f.
PAM.....	370
pan.....	472, 730f., 733, 735f.
Panasonic GH5S.....	390, 818
Panasonic GH5S at 240fps.....	386
Panasonic LUMIX DC-GH5S.....	654
Panasonic LUMIX GH5S.....	652, 657, 674, 751
Panning Speed.....	645
pannini.....	181
PanoMap (DR).....	596
Panorama.....	186
Panoramic camera.....	711
PanoView XDV360.....	378, 652
PanoView XDV360 camera.....	704
Paragon Optics 26mm f/2.8 180°.....	660
Paragon Optics 63.5mm F/2.5 60°.....	660
Passing the FFmpeg output to FFplay.....	481
Patch Input/Output (DR).....	610
Patch Replacer (DR).....	563
Path in filtergraph.....	172
PATH system variable.....	14
Paths and filenames under Windows.....	627
Paul Bourke.....	171, 197, 203
pause.....	13
pc.....	606
PC(Storage) GH5S.....	764
PC(Tether) GH5S.....	764
pEmitter (DR).....	568, 575
Pencil Sketch (DR).....	564
Perigee.....	817
perspective.....	69f., 365
perspective (v360 option).....	181, 240
perspective (video filter).....	308, 455
perspective_to_xyz.....	198
pFastNoise (DR).....	574
pfm.....	319
PFM.....	370
PGM.....	370, 455
PGMYUV.....	370
Phantom power.....	507
PHM.....	370
Photo Style.....	684
PI.....	186, 262, 422
ping.....	433
pipe.....	709
pitch.....	180, 185, 189, 223, 230, 262, 289, 776
pivot.....	448
Pivot (DR).....	540
pix_fmt rgb48be.....	59
pix_fmts.....	190, 216, 318
Pixel format.....	318
Pixel formats.....	320
Pixel Size in Fulldome Planetariums.....	653
pixelize.....	164, 366
pixelize (DR).....	553
Pixels.....	654
pixels per linepair.....	81
Pixels per linepair.....	77f., 83
pixfmt.h.....	319
pixscope.....	160
pkt_size.....	434
planar.....	183, 319
Planar pixel format.....	34
PlanarTracker (DR).....	550, 554, 556
PlanarTransform (DR).....	551, 554f.
Planemo.....	797
planet.....	198, 204
Planet classification (SE).....	794
Planet classification scheme.....	794
Planet projection.....	208
Planetarium.....	264, 823
Planetarium in Bochum.....	653
Planetarium in Wolfsburg.....	653
Plants (DR).....	584
Play around selection (DR).....	521
Playback framerate (DR).....	525
Plugins (Ricoh Theta).....	713
PNG.....	370, 626
Point of view.....	831
point-to-point streaming.....	433
Point-to-point streaming.....	433
poly3 distortion model.....	176
Polygon (DR).....	571, 580
Polyline toolbar (DR).....	569
Polyphonic Synthesizer.....	756
Portable Arbitrary Map.....	370
Portable Float Map.....	370
Portable Graymap.....	370
Portable Pixmap.....	370
Portrait photography.....	743
Post-roll time (DR).....	533
Potentiometer behaviour.....	759
PotPlayer.....	726
POV.....	831
pow.....	78, 82, 169, 310, 421, 720
Power Bins (DR).....	527
Power User Task Menu.....	14
Powerbank.....	507

PowerGrade (DR).....	541	PTS-STARTPTS.....	94
PPM.....	370	pts:hmsl:%OFFSET%.....	100
PQ.....	831	pTurbulence (DR).....	569
Pre-roll time (DR).....	533	Pulnix TM-9701.....	654
pre-trigger.....	317	Pulse Width.....	759
Predicted frames.....	830	PW.....	759
Preferences (DR).....	524	qcif.....	376
pRender (DR).....	569	qhd.....	376
Presentation time stamp.....	339	qntsc.....	376
Presentation Time Stamp.....	831	qpal.....	376
preserve lightness.....	47, 49	qqvga.....	376
Presets / Patches.....	759	Quadrature Signal.....	492
Preview image.....	407	Qualifier (DR).....	566
Preview image.....	408	Quick_Export (DR).....	545
Preview image in Wordpress.....	408	Quotation character.....	291
PrimaryColour.....	403	Quotation mark.....	291
print.....	179, 421, 423ff., 427, 449	qvga.....	376
print (with loglevel).....	427	qxga.....	377
Prism Blur (DR).....	564	R Lower 3rd (DR).....	561
Process.Start.....	295	R'G'B'.....	319
ProcessExplorer.....	158	Radial Blur (DR).....	562
ProcessStartInfo().....	295f.	Radial distortion of lenses.....	168
Programming fuses (DR).....	603	Radial Gradients.....	352
Project Backups (DR).....	524	Rain sounds.....	811
Project Manager (DR).....	528	Rainbow-trail effect.....	138
Project settings (DR).....	525	random.....	161f., 421
Projection lens.....	210	RAW images.....	616
Projection lenses.....	660	RAW speed test (DR).....	610
Projection_Ratio.....	660	RawTherapee.....	175
Proper credit.....	485	Re-numbering images.....	338
Proper motion of stars.....	778	Reactor Installer (DR).....	592
Properties of clips (DR).....	609	Read-out chip size.....	654
prores_ks.....	333	Read-out Chip Size.....	686
Proxy Mode (DR).....	612	Real-time bluescreening	157
pseudocolor.....	361	Real-time zmq adjustment.....	296
Pseudocolors.....	361	realtime (FFmpeg filter).....	432
ptlens distortion model.....	176f.	Realtime (FFmpeg).....	432
PTS.....	339, 831	Realtime remapping.....	192
		Realtime Wormhole.....	264
		Rec Quality.....	386
		Rec.2020 HLG ARIB STD-B67 (DR).....	559
		Rec.2100 HLG (DR).....	559
		Rec.2100 HLG (Scene) (DR).....	559
		Rec.709 (DR).....	559
		Rec.709 (GH5S).....	681
		Rec.709 HLG ARIB STD-B67 (DR).....	559
		Recommended bitrate (DR).....	545
		Record Formats, GH5S.....	675
		Record nature sounds.....	811
		Record sound.....	478
		Record, process and play sound.....	481
		Recording duration on SD cards.....	689
		Recording nature sounds.....	811
		Rectangular wave.....	487, 489
		rectilinear.....	181, 183, 197, 203
		Rectilinear to Equirectangular (DR).....	593
		Red filter.....	35
		Redshift.....	356, 358
		Reducing the framerate (DR).....	585
		Reel (DR).....	605
		Reflect (DR).....	598
		Reframing 360° videos with KartaVR (DR).....	592
		reinit.....	302
		Reinstance (DR).....	538
		Relativistic redshift.....	356
		Relink Media (DR).....	528
		rem.....	13
		remap....	101, 169, 186, 190, 193, 195f., 206, 223,
			241, 245, 249, 258, 261ff., 284, 365, 455, 459,
			720
		Remap filter.....	253
		Remap fisheye to equirectangular.....	186
		remap_opengl.....	439, 455
		remap=fill=.....	202
		remap=fill=green.....	216
		Remote App (Ricoh Theta).....	713

Remove low frequencies.....	473	RGB.....	331
Remove silence.....	474	RGB Color Cube.....	327
removegrain.....	315, 365	RGB/XYZ matrix.....	374
Removing a special effect (DR).....	561	rgb0.....	192
Removing subtitles.....	405	rgb24.....	319
ren.....	338	rgb48.....	128, 381
rename (Unix).....	648	rgb48be.....	319
Render Cache (DR).....	525	rgb48le.....	319
Render Cache Color Output (DR).....	524	rgba.....	192, 319
Renderer3D (DR).....	598	rgbashift.....	74, 162, 165, 354, 365
Replace (DR).....	530	RHS.....	831
Replace lensfun by remap.....	176	Ricoh Theta.....	230, 269
Replace one frame by another.....	117	Ricoh Theta Basic app.....	712, 716
replace v360 by remap.....	196	Ricoh Theta Movie Converter App.....	712
RequestSocket().....	296	Ricoh Theta V.....	712
Resampling (IrfanView).....	615	RICOH THETA V/Z1 4K.....	723
reserve_transparent.....	115	RICOH THETA V/Z1 FullDH.....	723
Reset the user interface (DR).....	525	Ripple (DR).....	564
Reset UI Layout (DR).....	525	Ripple Delete (DR).....	530
Reset User Preferences (DR).....	524	Ripple Edit (DR).....	532
reset_rot.....	301	Ripple Overwrite (DR).....	531
Resize (DR).....	597	Rode NT1 microphone.....	507, 509
Resize a clip (DR).....	532	Rode NTG2 microphone.....	509
Resolution of the moon surface.....	817	roll.....	180, 185, 223, 230, 262, 289, 776
Resolve Live (DR).....	584	Roll Edit (DR).....	531
ResolveFX Revival --> Deflicker.....	131	Rolling shutter.....	298
Restore Project Archive (DR).....	528	rolling shutter flicker.....	132
Retime and Scaling (DR).....	534	Romanyuk, Vladimir.....	794
Retime And Scaling (DR).....	585	root.....	170, 203, 421, 449
Retime Controls (DR).....	534	rorder.....	180, 185, 776
Retime Frame (DR).....	585	Rosin.....	754
Retime Process (DR).....	534, 585	rotate.....	237, 266, 365, 385, 455
reverse.....	89	Rotating earth.....	289
Reverse a video (DR).....	587	Rotating the earth, moon or planet.....	238
reverse speed (DR).....	587	Rotation axis.....	289
Reverse the speed (DR).....	534	Rotation matrix.....	828
Revert to last saved Project.....	611	Rotation order.....	180, 185
RG-59 coaxial cable.....	640	round.....	421
		Royalty-free music.....	485
		RR-BK01 remote control.....	706
		RS232 port.....	771
		RTMP Streaming.....	437
		rtp protocol.....	437
		rubberband.....	465
		Rucker, Rudy.....	259
		Running clock.....	99
		Running water.....	811
		S/2009 S1.....	797
		Sachtler.....	642
		same expression for the R, G and B.....	422
		Sample rate.....	465, 510
		Samyang 8mm Fisheye II.....	655
		Sankt Andreasberg observatory.....	653
		Sankt Andreasberg Observatory.....	210
		SAR.....	440, 831
		Sat (DR).....	540
		Saturation.....	27
		saturation (hue filter).....	40, 42
		Saturn's rings.....	797
		Save Color Gradings (DR).....	543
		Save New Preset (DR).....	545
		Saving a still (DR).....	541
		Sawtooth wave.....	487, 732
		SBV.....	403
		scale....	70, 77, 83, 105, 116f., 146, 149, 245, 289, 365, 378, 381, 456, 776
		scale filter (without any options).....	325
		scale_qsv.....	456
		scale=flags=accurate_rnd.....	326
		scale=out_range=pc.....	326
		scale=sws_dither=none.....	326
		scale2ref.....	104f., 220, 456
		Scanlines (DR).....	564
		Scenes (OBS).....	635
		Schulze, Gerald.....	570
		Schwarzschild radius.....	201, 254, 278

screen (blend filter).....	118	SER video file format.....	448
Screen pixel format of operating system.....	192	Serial numbers (ExifTool).....	613
Screenplain.....	646	Serial port.....	771
Screenwriting.....	646	set.....	13
scroll.....	664, 718	set /a.....	23, 109, 241, 249, 482, 619
Scroll text (DR).....	561	set a metadata tag (exiftool).....	614
Scrollbar.....	297	setpts.....	92ff., 117, 138f., 316, 381, 386
Scrollbars.....	157	setpts=PTS-STARTPTS.....	92
scrolling up.....	98	setsar.....	400
Scrolling up.....	404	Setting file (DR).....	537
SD card.....	763	Settings, GH5S.....	684
sdl2.....	236, 456	setts=dts=DTS-STARTDTS.....	193
SDR.....	831	sg.....	181
Second screen.....	640	sgn.....	421
Secondary Adjustments (DR).....	543	Shackleton Crater.....	818
sed (Unix).....	648	Shad (DR).....	540
seeking.....	84	Shape3D (DR).....	579, 598
Segment.....	93	Shapecatcher.....	831
segment (video filter).....	108	SharpCap.....	634, 806f.
Segment-wise linear interpolation.....	422	Sharpen (DR).....	563
Segoe UI Symbol.....	522	Sharpen Edges (DR).....	563
select.....	26, 135	Sharpen images.....	75
Select a clip under the playhead (DR).....	522	Shell script (Unix).....	620
Select a gap (DR).....	522	Short test tone.....	499
Select All Points (DR).....	580	shortest=1.....	381
Selection follows Playhead (DR).....	525	Shotcut.....	612
Selection Mode (DR).....	522, 532	Show filter curves.....	622
selectivecolor.....	44, 365, 456	Show Reference Wipe (DR).....	544
Selectivecolor filter.....	44	showfreqs.....	498
Semitone.....	758	showinfo.....	229
sendcmd114, 162, 275, 284, 286, 288f., 291, 301, 450, 456		showspectrum.....	457, 476
sendcmd examples.....	291	showwaves.....	457, 496, 500f., 506f., 737
sendcmd=c=.....	114	showwavespic.....	457, 466, 500, 729, 738
Sending commands with ZMQ.....	292	shufflepixels.....	457
sense.....	308	shuffleplanes.....	34, 457
Sentech.....	636	Shutter angle.....	644
Sequential number.....	339	Sigma 14mm f/1.8.....	645, 668, 691
		Sigma 24mm f/1.4.....	668, 691
		Sigma 4.5mm f/2.5.....	171
		Sigma 50mm f/1.4.....	668, 691
		Sigma 8mm f/3.5.....	171
		Sigma EX DG 4.5mm.....	655
		Sigma EX DG 4.5mm f/2.8.....	668, 691
		Sigma EX DG 8mm.....	655
		Sigma EX DG 8mm f/3.5.....	668, 691
		Signal range.....	374
		signalstats.....	389, 393
		sin.....	169, 421, 492
		sine....	359, 465f., 472, 475, 477, 486, 499f., 508,
		730f.	
		sine intensity profiles.....	81
		Sine tone.....	486
		sine-shaped crossfade.....	119
		Single quotation mark.....	291
		single quotes.....	481
		Single quotes.....	291
		sinusoidal.....	181
		Sinusoidally rising and falling tone.....	486, 488
		Size of color-look-up tables.....	62
		Size of the universe.....	822
		Slide Edit (DR).....	533
		Slideshow with scrolling images.....	22
		Slip Edit (DR).....	532
		slow.....	334
		Slow motion.....	92, 94
		Slow motion (DR).....	585
		Slow motion ramp.....	95
		slower.....	334
		Small, medium and large texture (DR).....	563
		Smaller nose.....	310
		Smart Bin (DR).....	527
		Smart Reframe (DR).....	548
		smartblur.....	159, 365
		Smear (DR).....	586
		Smooth cut (DR).....	529
		Smooth transition region.....	120

smoothing (vidstabtransform filter).....	299	Special characters.....	831	sqrt(-1).....	424
SMPTE.....	831	Special effects.....	830	squish.....	79, 421
SMPTE Color Bar (DR).....	562	Special effects (DR).....	561	sRGB.....	373
SMPTE Color Bars.....	358	Spectral class.....	796	SRT.....	403
SMPTE timecode (DR).....	605	Spectrum.....	347, 350	st.....	161f., 169, 258, 310, 422, 489
smpてbars.....	358	Spectrum images.....	341	Stabilization.....	298, 764
SMTSEC 2.27mm.....	656	speechnorm.....	457	Stabilization (DR).....	548
Snapping tool (DR).....	522	Speed Change (DR).....	585	Stabilization of 360° Videos.....	301
Snowfall (DR).....	568	Speed Editor (DR).....	529	Stabilization of 360° Videos, improved.....	306
SNR.....	831	speed factor.....	386	Stack videos.....	111
sntsc.....	376	Speed of light.....	493	Standard Still Duration (DR).....	524
Soften & Sharpen (DR).....	563	Speed of sound.....	512	Standard Vector Graphics.....	612
Solar mass.....	494	Speed ramp.....	95	Star occultation.....	388
Solar System Model.....	822	SpeedBooster.....	657, 692	Star tracking.....	246
Solid Color (DR).....	562	SpeedBooster 0.64x.....	390, 691	Star trails.....	134
Solo (DR).....	588, 590	SphereMap (DR).....	598	start (Batch command).....	436, 620
solstice.....	239	Spherical 360 image.....	429	Start a FFplay process (C#).....	295
Solve (DR).....	581	Spherical coordinate system.....	185	Start counting PTS from zero.....	92
Sony A7S II.....	654, 657	Spherical rotation.....	185	Start Timecode (DR).....	524
Sony E.....	655	Spherical sidedata.....	727	start_duration.....	86, 110
Sony E-Mount.....	662	Spherical stabilizer (DR).....	307, 596	Starting timecode (DR).....	605
Sound effect.....	466	Spherical transformation.....	196	Static keyframe (DR).....	542
Sound effects.....	486	Spherical Transformations with two Inputs.....	276	STC-TB33USB.....	636
Sound effects for horror films.....	752	SphericalStabilizer (DR).....	548, 596	StdErr.....	427
Sound levels (DR).....	536	Spill removal (DR).....	555	StdOut.....	427
Sound library (DR).....	609	Spinning Black Holes.....	253	Stellarium.....	780
Sound records.....	507	Spiral effect.....	251	Step response.....	737
Sound_Rec_Level_Adj.....	751	Spline Editor (DR).....	560	stereo3d.....	111
Sound_Rec_Level_Limiter.....	751	split...78, 116, 129, 132, 139, 215, 229, 299, 330, 380f., 394, 506, 627, 731		stereographic.....	183, 199, 205
Source code.....	647	Split a video in multiple segments.....	109	stereographic_to_xyz.....	199
Space Engine.....	782	Split a video into audio and video.....	484	Stereomix.....	396
Space simulation.....	782	Split planes.....	140	Sticky Note (DR).....	538
SpaceEngine.....	192, 195	Spoken dialogue (DR).....	536	Still (DR).....	541
spal.....	376	Spot.....	683	stillimage.....	334
Spatial Media Metadata Injector.....	187, 727, 730	SpyderX Elite.....	624	Stitch double-fisheye images with alignment errors.....	231
spatial metadata.....	727	sqcif.....	376	Stitching.....	711
Spatial metadata.....	730	sqrt.....	421	Stitching artefacts.....	227
Speakers.....	482				

Stop Motion (DR).....	564	Swap planes.....	34
Stops.....	678, 682	swapuv.....	57, 365
Stream Key (OBS).....	638	Switch between two cameras.....	110
stream metadata.....	338	sxga.....	377
streaming.....	433	symbolic link.....	294
Streaming.....	432	symlink.....	294
Streaming (OBS).....	635	Synchro Scan GH5S.....	690
streaming it to a UDP address.....	317	Synchronize audio with video.....	484
streamselect.....	114, 275, 284, 291, 458	Synchronize audio with video (DR).....	547
Strong contrast enhancement.....	28	Synthesizer Abbreviations.....	759
Studio Mode (OBS).....	641	Synthesizers.....	756
Stylize (DR).....	564	System Frequency.....	386
Subnet Mask.....	433	System variables.....	14
SubRip.....	402	T 2.....	662
Subsequent 37.....	757	Tactile planetarium.....	823
Subtitle (DR).....	561	Takahashi FS-128.....	668, 692
subtitles.....	403, 405	tan.....	422
Subtitles.....	402, 404	TASCAM DR-701D Recorder.....	507
Subtitles (DR).....	604	TASCAM DR-70D.....	746
subtract (blend filter).....	118	TASCAM DR-70D recorder.....	507
Subtracting a darkframe.....	128	Task Manager.....	636
Subtracting a darkframe (DR).....	603	taskbar.....	236
SubView (DR).....	581	Taskbar.....	236
Subwoofer.....	496	taylor.....	449
sudo.....	647	TB.....	92, 95
sudo apt install.....	648	tbc.....	440
sudo make install.....	648	TBC.....	831
Suggestions for improvement.....	442	tbn.....	440
Super 8 simulation (DR).....	585	tbr.....	440
Super Earth.....	822	tcp://*:5555.....	292
superequalizer.....	458	tcp://localhost:5555.....	296
superfast.....	334	Teleconverter.....	668, 692
Supernova remnant.....	821	Telescopic effect GH5S.....	690
superoceanic (SE).....	794	Temp (DR).....	540
Supported Codecs (DR).....	605	temperate (SE).....	794
Supported pixel formats.....	371	Temperature class.....	794
Surface of the earth.....	822	Templates (DR).....	537
svga.....	375f.	Temporal averaging (DR).....	586
		Temporal slice-stacking effect.....	139
		Terra (SE).....	795
		Test image with shifted RGB channels.....	354
		Test patterns.....	663
		Test video with audio.....	359
		testsrc2.....	26, 94, 161f., 358f., 458
		tetrahedron.....	182
		text.....	96
		Text (DR).....	561
		text_h (drawtext filter).....	98
		text_w.....	98
		Text+ (DR).....	561
		TGA.....	370
		thickness (drawbox filter).....	224, 304
		Thorne, Kip.....	253, 259
		threshold.....	283, 368
		TI.....	287
		TIFF.....	370
		Tilt-Shift Blur (DR).....	564
		Time Base Corrector.....	831
		Time delay.....	316
		Time domain.....	486
		Time ramp.....	95
		Time remapping.....	94f.
		Time Remapping (DR).....	585
		Time segment.....	474
		Time signal sender.....	824
		Timebase.....	92
		timecode.....	100
		Timecode (DR).....	605
		Timecode stream.....	407
		Timelapse.....	92, 778
		Timelapse (DR).....	585
		Timelapse duration table.....	763
		Timelapse_Control_V2.....	767
		Timelapse+ View.....	766
		Timeline framerate (DR).....	525
		Timeline is destroyed (DR).....	611

Timeline resolution (DR).....	526	Triangular wave.....	487, 490
Timeline resolution (DR).....	525, 545, 596	trim.....	23, 84f., 94, 108, 458, 517
timeline support.....	442	Trim Edit Mode (DR).....	522, 532, 535
Timeline support.....	409	trim=end_frame=1.....	408
Timeline_Proxy_Mode (DR).....	526	Tripod.....	642
timeout (Windows batch file).....	618	trunc.....	78, 422, 679
TimeSpeed (DR).....	539, 585	tsp.....	182
Tint (DR).....	540	Tunnel effect.....	249
Tiny Planet Projection (DR).....	592	Tunzelmann, Eugenie von	253
TMB Refractor.....	668, 692	Tutorial for git.....	649
tmedian.....	134f., 315, 365	Tutorials on Youtube (DR).....	519
tmix.....	128, 315, 365, 380f., 458	tv.....	606
Toggle between two images.....	114	twilight zone.....	239
Toggle between two video streams.....	114	Two 360° worlds.....	276
toilet bowling.....	742	Two-dimensional gradients.....	340
tonemap.....	372	TYC.....	779
tophat (morpho filter).....	165	Tycho catalog.....	779
torrid (SE).....	794	Ubuntu.....	647
tpad.....	86, 110, 139f., 317	UDP.....	317, 434
Track an object (DR).....	550	udp://.....	317, 433, 435
Track an object and corner pin an image (DR).....	556	uEye UI-1240ML.....	636
Track an object and cover it by a colored rectangle (DR).....	554	UHD 4K spherical test pattern.....	517
Track an object and cover it by an image (DR).....	555	uhd2160.....	375, 377
Track and blur (or pixelize) an object (DR)....	553	uhd4320.....	377
Trails (DR).....	573	ultrafast.....	334
Transcoding (DR).....	605	Ultrasound.....	475
Transform (DR).....	554f., 597	Unable to acquire the dpkg frontend lock.....	647
transform360.....	181	Uncompressed raw video.....	333
Transformation characteristics.....	373f.	Undo (DR).....	522
Transmission.....	643	Undo spherical rotations.....	185
Transparency color in animated GIF.....	116	Unicode character recognition.....	831
Transparent background.....	104	Unix.....	647
Transparent layer.....	104	Unix script file.....	648
transpose.....	77f., 83, 169, 301, 506f., 663, 731	unmapped pixels.....	197, 202, 278
trc.....	373	Unmapped pixels (remap filter).....	186
		unsharp.....	75, 365
		Unsharp edge.....	120, 715
		Uploading spherical 360 images to Facebook.....	429
		Uploading videos to Facebook.....	430
		USB Audio Device.....	478
		USB controllers.....	636
		USB keyboard.....	807
		USB Mode GH5S.....	764
		USB soundcard.....	478
		USB soundcard (DR).....	610
		USB Video.....	149
		USB-C (GH5S).....	690
		USB/MIDI Adapter.....	758
		User Datagram Protocol.....	317
		User-defined horizon.....	776
		UTF-8.....	97
		uxga.....	375, 377
		v_offset (v360 filter).....	209f.
		V-Log.....	562
		V-Log/V-Gamut Reference manual.....	677
		v360.....	103, 180, 185, 189ff., 196, 209, 215, 219, 223, 229f., 236, 240, 262f., 284, 289, 301, 365, 459, 663, 720, 735, 776
		v360=c6x1:e.....	798
		vaguedenoiser.....	315, 365
		varblur.....	76
		Variable blur.....	76
		Variable Frame Rate GH5S.....	688
		Variable neutral density filter.....	642
		Variables.....	16
		Variables in batch file.....	619
		variables in expressions.....	162
		VCA.....	759
		VCF.....	759
		VCO.....	759
		vectorscope.....	394
		Vectorscope.....	394
		Vertical alignment of text.....	98
		Vertical border.....	227
		Vertical flipping.....	111

Vertical jitter effect.....	162	VLog-L (DR).....	558	Window (Special effect in DR).....	562
veryfast.....	334	VLOG-L curve.....	680	Window Capture (OBS).....	634
veryslow.....	334	VLog-L to Rec709 LUT.....	607	Windows.....	10
vflip.....	111, 776	VO.....	831	Windows Accessories.....	831
VFR.....	831	Voice-Over.....	479	Windows firewall.....	433
vga.....	375f.	Voice-Over (DR).....	588	Windows taskbar.....	236
vibrance.....	54, 365, 459	Volatiles class (SE).....	794	Windows-IP-Konfiguration.....	433
Video astronomy.....	818	Voltage Controlled Amplifier.....	759	Wipes (DR).....	544
Video Capture Device (OBS).....	634, 641	Voltage Controlled Filter.....	759	Wireless Node (DR).....	538
Video Codecs.....	333	Voltage Controlled Oscillator.....	759	WLAN.....	433
Video codecs with alpha channel.....	336	volume. 385, 396, 462, 466, 472f., 475, 507f., 736		Wormhole.....	270, 276, 282, 826
Video format.....	17	Volume.....	463	Wormhole effect (DR).....	602
Video line jitter.....	161	Volume knob.....	473	Wormhole simulation.....	259, 665
Video Modes GH5S.....	693	volume=0:enable=.....	508	woxga.....	377
Video resolution.....	654	volume=enable.....	467	wqsxga.....	377
Video Size GH5S.....	686	volumedetect.....	499	wquxga.....	377
Video sizes.....	375	Vortex (DR).....	564	wqvgva.....	376
Video stabilization (DR).....	548	VR360 (DR).....	591	WQXGA.....	377
Video-in-Video.....	308	vstack.....	111, 132, 139, 330, 357, 361, 365, 393, 507, 664	Write text on a transparent layer.....	104
vidstabdetect.....	298f.	W/m^2.....	819	wsxga.....	377
vidstabtransform.....	298f., 459	warm (SE).....	794	wuxga.....	375, 377
View through eyepiece.....	120	Warper (DR).....	564	wvga.....	376
Viewer for 360° images.....	716	Watercolor (DR).....	564	wxga.....	377
Viewing below the horizon.....	820	Waveform Monitor.....	677	X-Rite ColorChecker.....	67, 625
Viewing direction.....	222	Waviness (DR).....	564	X-Rite ColorChecker Classic.....	558
Viewing direction (spherical images).....	714	Webcam.....	634, 707	X,Y,Z space.....	197, 277, 279
vignette.....	127, 365, 428, 460	Wegner, Gunter.....	519	X17 drone.....	743
Vignette (DR).....	564	while.....	422, 425	X17 Drone.....	742
Vignetting.....	127, 172	White level (DR).....	558	xfade.....	89, 460
Virtual Camera (OBS).....	641	White point (DR).....	540	xga.....	375, 377
VirtualDub.....	298	whsxga.....	377	xmap.....	186
VLC.....	709f.	whuxga.....	377	xmap and ymap files.....	196, 277
VLC player.....	10, 636, 648	Wiki.....	10	xor (blend filter).....	118
VLC Player.....	437, 621, 716, 723	Wildcards.....	617	xstack.....	61, 111f., 247, 383
VLOG.....	63	Wim Wenders.....	35	xyz_to_ball.....	204
VLog (DR).....	558	win_size.....	497	xyz_to_cylindrical.....	203
VLog table.....	678	Wind noise.....	473, 811	xyz_to_dfisheye.....	203
VLOG-L.....	676, 684			xyz_to_equirect.....	202

xyz_to_equisolid	205	zmqsend	286	-c:v dnxhd	298, 333, 586, 606, 608
xyz_to_fisheye	202	zmqsend.exe	292	-c:v ffv1	333
xyz_to_flat	203	Zoom Blur (DR)	562	-c:v ffv1 -level 3	333
xyz_to_orthographic	206	Zoom level	812	-c:v h264	366, 437, 679
xyz_to_stereographic	205	zoompan	19, 91, 179, 288, 385, 461	-c:v h264_nvenc	396
Y'CbCr	319	zscale	373f., 461	-c:v libx264	333
yaw	180, 185, 262, 289	ZWO ASI178MM	654, 662	-c:v libx265	333
YCbCr	326	zzz (DR)	529	-c:v libxvid	333
YCbCr 4:4:4	326	PIXPRO SP360 4K as webcam	707	-c:v mpeg4	333, 400, 480
YCbCr Ramp (DR)	562	^ character	13	-c:v prores_ks	141, 333, 336
ymap	186	^G	618	-c:v rawvideo	333, 336
Youtube	181, 430	--enable-zlib	647	-channels	478
YouTube Live	432	--keep-open=yes (MPV)	622	-channels	481
YouTube Live (OBS)	639	--lavfi-complex= (MPV)	622	-codec:v	15
Youtube recommended settings	431	-ac	431, 463, 468	-color_range	298
Yumiki 2.5mm	656	-acodec	17	-color_range pc	333, 606, 608, 679
YUV/RGB transformation matrix	374	-acodec copy	17	-count_frames	514
yuv420p	319, 326, 430	-af	385, 516	-crf	17, 334, 431, 679
yuv420p10le	319	-alwaysontop	516	-disposition	407f.
yuv422p	319	-an	92, 516	-dumpgraph	292
yuv422p10le	319	-attach	408	-exitonkeydown	516
yuv444p	318f.	-audio_buffer_size	479, 481, 483	-exitonmousedown	516
yuv444p10le	319	-autoexit	236, 497, 516, 735f.	-f	516
yuva444p10le	333	-autoexit (FFplay)	517	-f concat	21
yuvj420p	667	-b:a	431	-f dshow	149, 245f., 395f., 481f., 724f.
yuvj422p	318, 666	-b:v	334, 725	-f gdigrab	192f., 195, 247, 396f.
yuvjxxxp	326	-bf	431	-f image2	294
Zebras	677	-bits_per_mb	333	-f image2pipe	450
Zebras GH5S	684	-bsf:v	193	-f lavfi	94f.
zerolatency	334	-bufsize	334	-f mp4	334
zeromq	286	-c copy	84, 87, 109	-f mpegs	266, 275, 317, 433, 725
Zhiyun Crane 3S	764	-c:a	431	-f null	499
zimg	373	-c:a copy	17	-f null NUL	427f., 499
zlib	647	-c:a pcm_s24le	298, 333, 606, 608	-f nut	481f., 497, 725, 735f.
zlib1g-dev	647	-c:s	403	-f rawvideo	333
ZLKC (OCDAY) 7.5mm	655	-c:s mov_text	604	-f rtp	437
zmq	286, 292, 296, 450, 460	-c:v copy	17, 401	-f sdl	160
ZMQ	157			-f sdl2	236, 245, 479, 725

-f sdl2	149f., 192f., 195, 247, 294, 317, 435	
-f segment.....	109	
-filter_complex.....	128, 381	
-filter_complex_script.....	316	
-filter_hw_device.....	439	
-flags.....	431	
-frame_pts.....	339	
-framerate.....	15, 18, 22, 113, 249, 266, 434	
-framerate (dshow input device).....	149, 317, 395, 399	
-framerate (gdigrab).....	396f.	
-frames.....	344	
-fs.....	236, 516, 735	
-g.....	335, 434	
-h.....	450	
-h demuxer=dshow.....	395	
-h encoder=.....	606	
-h filter=.....	147	
-h muxer=.....	606	
-hide_banner.....	334	
-hwaccel.....	438	
-i color.....	224	
-i desktop.....	192, 195, 397	
-i title.....	397	
-i video="USB Video".....	399	
-ignore_loop.....	116	
-inf.....	424	
-init_hw_device.....	439	
-itsoffset.....	484	
-lavfi.....	94f.	
-left.....	296, 450, 517	
-left (FFplay option).....	436, 516	
-list_devices.....	395, 481f., 724	
-list_options.....	395, 478, 481f., 724	
-loglevel repeat.....	423, 426	
-loglevel repeat+error.....	427f.	
-loop.....	113f., 474, 516	
-loop (FFplay).....	516f.	
-loop 1.....	134, 186f.	
-map.....	86, 406ff., 479, 730	
-map 0:0 -map 0:1.....	407	
-map 1:a.....	472	
-map_channel.....	470, 509, 728	
-map_metadata.....	298, 338, 727	
-map_metadata:s:v:0.....	338	
-maxrate.....	334	
-metadata:s:s:0.....	403, 604	
-minrate.....	334	
-movflags.....	298	
-movflags faststart.....	431	
-movflags write_colr.....	333, 606, 608	
-noauto_conversion_filters.....	325	
-noborder.....	265, 436, 516	
-nodisp.....	481, 516	
-of.....	514	
-of csv.....	368	
-of csv=p=0.....	283, 368, 513	
-offset_x.....	192, 195, 397	
-offset_y.....	192, 397	
-overwrite_original.....	429	
-pix_fmt.....	99, 318, 325, 333, 388f., 430	
-pix_fmt rgba.....	336	
-pix_fmt rgba64be.....	336	
-pix_fmt rgba64le.....	336	
-pix_fmt yuv422p10le.....	298, 333, 608	
-pix_fmt yuv444p10le.....	679	
-pix_fmt yuva444p10le.....	336	
-pix_fmt yuva444p12le.....	336	
-pix_fmts.....	320	
-pixel_format.....	150, 395	
-pixel_format yuyv422.....	399	
-preset.....	17, 334	
-preset help.....	334	
-profile:v.....	298, 333, 606, 608	
-profile:v 4.....	333	
-ProjectionType (exiftool).....	714, 716, 727	
-protocol_whitelist.....	437	
-q:v.....	15	
-qp.....	334	
-r.....	18, 113, 431	
-r:a.....	431	
-re.....	294, 432, 437, 479	
-readrate 1.....	294	
-report.....	427	
-rtbufsize.....	395, 400	
-s.....	474	
-sample_rate.....	481	
-sample_size.....	481	
-sar.....	400	
-sdp_file.....	437	
-segment_time.....	109	
-select_streams.....	514	
-shortest.....	15, 111, 467f., 480	
-show_entries.....	283, 368, 513f.	
-show_frames.....	514	
-show_region.....	397	
-skip_frame.....	514	
-slices.....	426	
-sn.....	405, 516	
-ss 59, 61, 84, 99f., 108f., 136, 146, 462, 474, 516		
-ss before and after input file.....	85	
-sseof.....	26, 87, 300	
-start_number.....	15, 22, 249, 284, 339, 358	
-stream_loop -1.....	294	
-strftime.....	339	
-strict -1.....	266	
-strict experimental.....	727	
-strict unofficial.....	727	
-t.....	108, 462, 627	
-t (FFplay).....	516f.	
-threads.....	426ff.	
-to.....	84, 109	
-top.....	296, 450, 517	
-top (FFplay option).....	436, 516	

-tune	334	./my_script	648	*.sdp	437
-update	300	[expr]	114, 286, 288, 301	*.ser	448
-v debug	229	[vid1] (MPV)	622	*.srt	402
-v error	514	[vo] (MPV)	622	*.srt (DR)	604
-v verbose	229, 325	*.3dl	66	*.svg	612
-vcodec	17, 149, 317	*.ass	404	*.txt	427
-vcodec (as an input option)	401	*.ass subtitles	98	*.uov (Guide 9.1)	774
-vcodec copy	17	*.bat	13	*.xml	172, 174, 176
-vf	27, 516	*.cmd	286	/dev/null	499
-video_size	149, 192, 195, 247, 317, 395, 397, 399, 450	*.csp	66	/var/lib/dpkg/lock-frontend	647
-video_size (FFplay option)	516	*.csv	368, 513	#!/bin/bash	648
-vn	516	*.cube	59, 63ff., 356	% character	13
-vsync	21	*.dat	66	%~	617
-window_borderless	192f., 195, 436, 456	*.dcp (Adobe digital camera profile)	175	%0nd	339
-window_x	149f., 192, 195, 245, 247, 436	*.dra (DR)	528	%1	513
-window_y	149f., 192, 245, 436	*.drf (DR)	528	%d	339
-x (FFplay option)	436, 516	*.drp (DR)	528	%H	339
-XMP-GSpherical:Spherical="true"	727	*.lcp (Adobe lens correction profile)	175	%m	339
-y	25, 626	*.lnk	294	%M	339
-y (FFplay option)	436, 516	*.m3d	66	%nd	339
::	13	*.mkv (DR)	605	%S	339
:eof	619	*.mov (GH5S)	675	%Y	339
./configure	647	*.mp4 (GH5S)	675	> (Windows batch file)	619
./configure --enable-encoder=png	647	*.mts (GH5S)	675	>> (Windows batch file)	619
./configure --list-encoders	647	*.nut	333, 336	 character	291
		*.pgm	207, 424, 452		