

SiggySoft Video Compare User Manual

SiggySoft Video Compare shows two versions of the same video in one picture, in step: a tape capture and its upscaled copy, a file before and after deinterlacing, two encodes of one master. A line moves across the picture, the difference view shows what changed, and the smaller video is enlarged with nearest neighbour, so an upscaler is measured against the pixels that were really there. This manual covers the window, each way of looking, moving through time, lining the two up, saving and measuring, and what to do when something does not work.

How to read this. Part A says what the program does, what it needs and where everything is in its window. Part B is a comparison from start to end: choosing the two videos (Chapter 4), bringing them to one size (Chapter 5), looking, and reading the difference views (Chapter 6), moving through time and lining them up (Chapter 7), the steps applied to each side, with interlacing and the field view (Chapter 8), the scopes (Chapter 9), the PSNR, SSIM and VMAF scores (Chapter 10), saving pictures (Chapter 11), and presets and sessions (Chapter 12). Part C is reference: settings, troubleshooting, the Feedback menu (Chapter 15), and every key in Appendix A. Help > Manual opens this page in the browser.

Contents

PART A · GETTING STARTED

- 1What SiggysSoft Video Compare does
 - Two versions of one video in one picture, in step; the files stay as they are
 - What it shows · The honest baseline · In step · What it does not do
- 2Before the first comparison
 - Installing, the setup check, the FFmpeg libraries, and updates
 - Installing · What the PC needs · The setup check · The FFmpeg libraries · Updates
- 3The window
 - The choices on the left, the picture in the middle, what is being compared on the right
 - The choices · The picture · The information panel · The two files, row by row · Window sizes and full screen

PART B · COMPARING

- 4Choosing the two videos
 - Left is the original, right the processed copy; the badges say what each file is
 - Left and right · Several right videos · What the lines under each box say · From the command line
- 5Bringing them to one size
 - The honest baseline: the smaller video is enlarged with nearest neighbour
 - The canvas · Nearest neighbour, and the others · Pixel shape and picture shape
- 6Looking
 - The layouts, the difference views and how to read them, zoom, the magnifier, the crop, and the text on the picture
 - Layouts and the split line · The difference views · Reading the difference views · Zoom and the magnifier · Cropping both sides · Labels, swapping and hiding · The text on the picture
- 7Moving through time and lining them up
 - Playing, stepping on any codec, the loops, the time shift, and finding it automatically
 - Playing and going to a time · One picture at a time · Loops · Lining them up: the time shift · Finding the shift automatically · Reading the offset finder's result
- 8The steps on each side
 - Deinterlacing, the field view, turning, colour, and FFmpeg filters
 - Automatic steps, one switch each · What interlaced video is · The field view · Checking a field order · Judging a deinterlacer · FFmpeg filters
- 9The scopes
 - The histogram, the vectorscope and the waveform of both sides, and how to read them
 - Opening the scopes · Levels: 16 to 235, and IRE · The histogram (F1) · The vectorscope (F2) · The waveform (F3)
- 10Measuring: PSNR, SSIM and VMAF
 - What each score measures, how the program measures it, and how to read the numbers
 - What a score is, and what it is not · How the program measures · PSNR · SSIM · VMAF · The three side by side, measured · Upscales: the scores follow the enlargement · Reading a stretch · Before you trust a score · An example: ranking two encodes
- 11Saving pictures, and the pixels under the mouse
 - PNG files of both sides and the screen, and the exact values of two pixels
 - Saving pictures · The pixels under the mouse
- 12Presets and sessions
 - Ten saved comparisons, and windows that each keep their own
 - Preset slots · Sessions · The PowerShell launcher's presets

PART C · REFERENCE

- 13Settings
 - Tools > Settings
- 14When something does not work
 - What the program says, and what to do
- 15Feedback
 - Reporting a problem, suggesting a feature, asking a question or saying thanks, from the Feedback menu
 - Sending a report · My feedback · When the site cannot be reached · What is sent, and who sees it

- A

Keys

Every key; H shows them on the picture, and Help > Keys finds one by name
- B

Where the program keeps its files

For the Windows user signed in
- C

Licences

SiggySoft's terms, and the software it uses under its own
- D

Coming from the PowerShell launcher

What stayed, and what changed

PART A

Getting started

What SiggysSoft Video Compare does, what it needs, and where everything is in its window.

What SiggySoft Video Compare does

Two versions of one video in one picture, in step; the files stay as they are

What it shows

- **Split with a moving line.** One picture, the original on the left of the line and the processed copy on the right. The line follows the mouse, so a slow sweep across a face shows exactly what an upscaler added. This is the view for filming.
- **Side by side, or one above the other.** Both pictures whole, for framing and colour.
- **The difference.** Black where the two are the same, bright where they differ: the clearest proof that a process changed something, and that it did not.
- **Up close.** Zoom to 1:1 pixels or up to 800 percent, a magnifier in a corner, or crop both sides to the same small area.

The honest baseline

A 720 x 480 capture and its 1620 x 1080 upscale are different sizes, so something has to enlarge the capture before the two can be laid over each other. When that something is a smooth filter, as most tools use without saying so, the upscaler is compared with a second upscaler and looks better than it is. SiggySoft Video Compare enlarges the smaller video with nearest neighbour unless you choose otherwise: each pixel becomes a block, and nothing is invented. Chapter 5 explains it, and the line under the choice says what was done to each side.

In step

The left video sets the time. The right one shows its own picture for that time, plus a shift you set when the two do not start together. Each picture is matched by its timestamp, not by counting frames, so a 29.97 capture lines up with its 59.94 deinterlaced copy without either one being converted. One step forward or back moves to the next or previous moment at which either side changes picture, on any codec (Chapter 7).

What it does not do

- It never changes, moves or deletes a video. It writes only the pictures you ask it to save, its settings and its log.
- It plays no sound.
- It makes no new video files. SiggySoft Video Editor does that.

Before the first comparison

Installing, the setup check, the FFmpeg libraries, and updates

Installing

Download `SiggySoftVideoCompare-win-Setup.exe` from www.middleSiggy.com and run it. It asks nothing: it installs the program for the Windows user signed in, in `%LocalAppData%\SiggySoftVideoCompare`, without administrator rights, puts SiggySoft Video Compare on the desktop and in the Start menu, and first installs Microsoft's .NET 10 Desktop Runtime when the PC has none. That runtime is Microsoft's own installer, so Windows Update keeps it patched. Settings > Apps > Installed apps removes the program again; your settings, presets and saved pictures stay where they are.

What the PC needs

- Windows 10 or 11, 64-bit.
- The Microsoft .NET 10 Desktop Runtime, or a newer one. Setup installs it when it is missing.
- A graphics adapter with Direct3D 10 or 11. Without a graphics driver, Windows draws the picture itself, slowly; the setup check says so.
- The FFmpeg libraries (below): 77 MB, downloaded once.

The setup check

At the first start a page lists what the program needs and whether .NET and Windows are up to date. Each line says READY, NEEDED, WORKS WITHOUT IT or OUT OF DATE, one sentence on why, and has a button where the program can fix it. The page is always at Tools > Setup check. At every later start a quick check runs in the background; when it finds a problem a notice appears under the menu, with Open the setup check and a cross that hides that notice until something new turns up.

The FFmpeg libraries

SiggySoft Video Compare opens and decodes every video with the FFmpeg libraries, free software that is not part of it. On the setup check, Install the FFmpeg libraries explains where they come from and asks; when you agree it downloads an LGPL build of FFmpeg 9.0.2 for Windows from a public FFmpeg build project (76,972,461 bytes), checks the file against the SHA-256 published with it, and keeps only the libraries and their license in `%LOCALAPPDATA%\SiggySoft\ffmpeg-lgpl`. The download is pinned to the build of 2026-09-30, which that project keeps for good.

Why an LGPL build. The program loads these libraries into itself. FFmpeg is LGPL unless it is built with its GPL parts, and a GPL build would put the whole program under the GPL. So the program reads the license string inside the libraries before it loads them and refuses any GPL build, including one you point it at in Settings. The build it downloads is under the LGPL version 3 or later.

Updates

A copy installed with Setup looks for a new version at every start (Settings > Updates turns this off) and when you choose Help > Check for updates. A new version downloads in the background, often as a small package of only the files that changed, and goes in when the program closes. The setup check names it, with Restart and update to install it at once. A copy run from another folder, such as the publish folder of a build, does not update itself and its update line says so.

The window

The choices on the left, the picture in the middle, what is being compared on the right



The choices

The left column holds everything that makes up a comparison: the hundred preset slots, ten banks of ten (Chapter 12), the two videos (Chapter 4), how they are brought to one size (Chapter 5), the view and the labels (Chapter 6), the automatic steps (Chapter 8) and playback. Every choice is saved as it changes; there is no Save button. A (?) beside each group opens an explanation that stays until you close it.

The picture

The picture is drawn by the graphics card. Over it: the split line, the labels when they are on, each side's time at the bottom, and a line at the top that names anything unusual (the difference view, a loop, the zoom, a crop, swapped sides). Under it, the timeline shows both files to scale, the right one moved by the shift, with the playhead across both; click or drag on it to go to a time. Below that: start, one picture back, play, one picture forward, end, the times of both sides, the shift, the speed, Loop, Save and full screen.

The information panel

The right column says what is being compared. **Worth knowing** appears when a choice changes what the picture shows, such as the difference view, an enlargement other than nearest neighbour, an interlaced side shown without deinterlacing, or a shift. A choice remembered from an earlier session can otherwise make the next one look broken. **The two files** sets their details side by side, with values that differ in amber (below). **Under the mouse** gives both sides' pixels at the same point of the canvas (Chapter 11). **Measurement** holds the last PSNR, SSIM and VMAF (Chapter 10), and **Lining up in time** the offset finder (Chapter 7). A bar with Cancel shows while either runs. View > The information panel hides the column.

Worth knowing names each of these when it is on: the difference view; an enlargement other than nearest neighbour; an interlaced side shown without deinterlacing; pixel-shape correction off; a time shift; the field view on an interlaced side; the split line locked; opening full screen; the sides swapped; a crop.

The two files, row by row

The right column is the right video on screen; with several, its head says which (**Right 2 of 3**). A value in amber differs between the two sides, which is often the point of the comparison and sometimes the reason a comparison looks wrong.

ROW	WHAT IT SAYS
File	The file's name; hover for the whole name.
Size	Pixels across and down, as stored.
Pixel shape	square, or the shape of one pixel, such as 8:9 for DV and DVD at 4:3.

Shown as	The size with square pixels, which is what the canvas uses when "Correct the pixel shape" is on: 720 x 480 at 8:9 is shown as 720 x 540.
Frame rate	Frames a second, as the file says.
Pictures a second	Pictures shown a second after the steps: double the frame rate when an interlaced side is deinterlaced one picture per field or shown in the field view, each field in turn.
Length	How long the file plays.
Scan, measured	Interlaced (top or bottom field first), progressive, or unclear, from the pictures (the badge).
Field flag	What the file itself claims: <code>progressive</code> , <code>tt</code> or <code>bb</code> (top or bottom field first), or none. When it disagrees with the measured scan, the measurement wins.
Codec	How the pictures are compressed: <code>dvvideo</code> , <code>h264</code> , <code>hevc</code> , <code>ffv1</code> .
Pixels	How the colour is stored and with how many bits: <code>yuv411p, 8 bits</code> is DV's quarter-width colour, <code>yuv420p</code> half width and half height.
Colour	The colour matrix and range used to turn the file into the picture on screen, with "assumed" when the file names none.
Bit rate	The file's average data rate.
Container	The kind of file: <code>avi</code> , <code>mov</code> (also MP4), <code>matroska</code> (MKV), <code>mpeg</code> .
Turn	The turn the file asks for (phone videos), or none.

Window sizes and full screen

View > Larger and Smaller, or `Ctrl` + `+` and `Ctrl` + `-`, or `Ctrl` with the mouse wheel, make everything in the window larger or smaller in steps of 10 percent, from 60 to 150 percent; `Ctrl` + `0` goes back to 100. At 80 percent the window's smallest size is 848 by 512. `F11` or `Alt` + `Enter` shows only the picture, on the whole screen: the menu, both panels, their splitters, the transport bar and the status line all go, so on a 1920 x 1080 screen the picture is 1920 x 1080. `Esc` or `F11` again brings the window back. `F11` works wherever the focus is in the window, the picture included.

PART B

Comparing

A comparison from start to end: the two videos, one size, looking, time, the steps on each side, the scopes, the scores, saving, presets.

Choosing the two videos

Left is the original, right the processed copy; the badges say what each file is

Left and right

LEFT is the “before”: the original, untouched file. RIGHT is the “after”: the upscaled, deinterlaced or otherwise processed copy. Type or paste a path into a box and press `Enter`, use the folder button beside it, or drag files from Explorer: onto a box, onto the picture, or anywhere on the window. Two files dropped together fill both sides, the first one on the left. One file dropped on the picture goes to the half it lands on, the right half for the right video (the bottom half when the two are stacked); dropped elsewhere on the window, it fills the empty side. Any file the FFmpeg libraries read works, including AVI and DV captures, MP4, MOV, MKV, MPEG and still pictures.

Which side is which matters for two things. The left video sets the time, and the right one is shown at the left time plus the shift. And the left picture is the reference when the program measures (Chapter 10): the scores say how far the right picture is from the left one.

Swap exchanges the two files. Same file on the right puts the left file on both sides, to compare it with itself: deinterlaced on one side only, with two different filters, or shifted in time.

Several right videos

One original can be compared with several processed copies, such as QTGMC against bwdif, or two upscalers. “Add a right video” under the right box adds another; a numbered button appears for each, and the lit one is on screen. `Tab` shows the next one and `Shift`+`Tab` the one before, at the same moment, so the switch itself is the comparison; `Ctrl`+`Shift`+`1` to `Ctrl`+`Shift`+`0` pick one by number. The right box, its badge, the labels and the shift belong to the one on screen; each copy keeps its own shift and label. The bottom of the picture names it, for example `RIGHT 2/3`, and so does the head of the right column in the information table.

Only the copy on screen decodes while playing. While paused they all follow the clock, so a switch shows the same moment at once. The others keep 4 pictures behind instead of the Buffer’s 50, so three 1080p copies do not fill the memory. The minus button beside “Add a right video” takes the copy on screen off the list; the file is not touched. A preset keeps the whole list.

Ranking copies. Several right videos and the scores work together: with two encodes of one capture as right videos 1 and 2, the scores of each against the same original say which stayed closer. Chapter 10 has an example.

What the lines under each box say

The first line gives the size, frame rate, codec and length, and the pixel shape when the pixels are not square (`pixels 8:9` on DV). The second lists the steps applied to that side and its colour conversion, for example `deinterlaced, bottom field first, one picture per field; colour BT.601 (assumed), limited range`. “Assumed” means the file does not say, and the program used what players use: BT.601 below 720 lines, BT.709 from 720 up.

The badge says **interlaced** (amber) or **progressive** (green), from the pictures themselves: FFmpeg’s interlace detector compares the two fields of each frame on three stretches of up to 100 frames, at 10, 50 and 85 percent of the file, by the same rule SiggySoft Video Capture uses. A file’s own flag is often missing or wrong; on the development PC a capture of tape 24 had no useful flag and measured 300 of 300 frames bottom field first. Hover over the badge for the counts. The result is kept, so the same file is measured once. **unclear** means the counts did not lean far enough either way; the program then treats the file as progressive.

From the command line

`"SiggySoft Video Compare.exe" before.mp4 after.mp4` opens the two files. The PowerShell launcher’s form, `-Left before.mp4 -Right after.mp4 -Session tapes`, works too.

Bringing them to one size

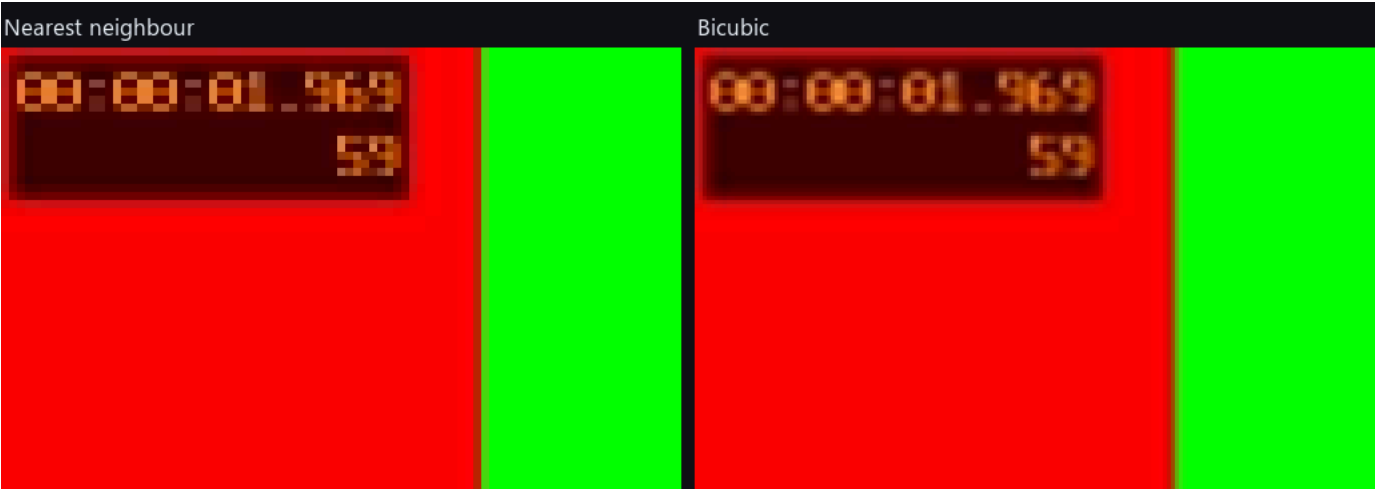
The honest baseline: the smaller video is enlarged with nearest neighbour

The canvas

Both pictures are drawn onto one canvas as large as the larger of the two, so the smaller one is enlarged and the larger one is never reduced. The line under “Enlarge the smaller video with” says how each side reaches the canvas, for example: `Canvas 1620 x 1080. Left enlarged 2.25 x 2.25, nearest neighbour. Right shown at its own size.` The scores in Chapter 10 are measured on this same canvas.

Nearest neighbour, and the others

CHOICE	WHAT IT DOES TO THE SMALLER PICTURE
Nearest neighbour (honest)	Each pixel becomes a block of the same colour. Nothing is invented: what you see is what the file holds. The default, and the baseline an upscaler should have to beat.
Bilinear	Blends each point from the 4 nearest pixels. Soft.
Bicubic	Blends from the 16 nearest (Catmull-Rom). Sharper than bilinear, and what most tools use without saying so.
Lanczos	Blends from the 36 nearest. The sharpest of the smooth ones, with faint rings at edges.



A 360 x 240 test picture enlarged twice onto a 720 x 480 canvas, here three times closer still. Nearest neighbour shows the pixels the file holds as blocks. Bicubic invents the in-between values, which softens the digits and the edge between the red and the green.

The enlargement and the zoom are one step, done by the graphics card. At 400 percent the original's real pixels show as blocks, not the pixels of an enlargement made earlier. With nearest neighbour at 1:1, the test routine compared 627,532 screen pixels with the 720 x 480 picture they came from and found every one equal; bicubic on the graphics card was within 1 level of 255 of the same sum done on the processor.

Not whole numbers. 480 rows to 1080 is 2.25 times, so nearest neighbour repeats some rows twice and others three times. It is still honest (no pixel is invented), but the blocks are not all the same height, and the line under the choice says so. When the smaller video is not enlarged with nearest neighbour, Worth knowing says that the comparison is against a second enlargement.

Pixel shape and picture shape

DV and DVD video store 720 pixels across a 4:3 picture, so its pixels are narrower than square. With **Correct the pixel shape** on, each side is measured at its real shape before the canvas is chosen. Narrow pixels make the picture taller, never narrower: a 720 x 480 picture with 8:9 pixels is shown at 720 x 540, so no column is dropped (1.0 showed it at 640 x 480, which dropped one column in nine). The information table's "Shown as" gives the size. When the two pictures have different shapes, such as 4:3 and 16:9, **Stretch both to one size** fills the canvas with both, and **Keep each one's shape** adds black bars instead. `(Shift)+(S)` switches between the two.

Looking

The layouts, the difference views and how to read them, zoom, the magnifier, the crop, and the text on the picture

Layouts and the split line

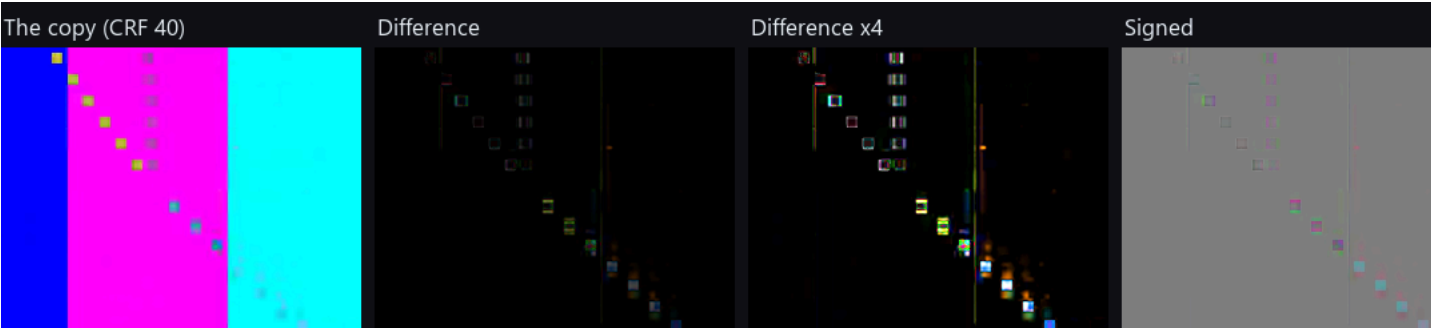
Layout chooses Split with a moving line, Side by side, or One above the other (**Shift**+**M** goes through them). In the split, the line follows the mouse over the picture; it stays where it is when the mouse leaves. **L** locks the line where it is: it stops following the mouse, so the mouse can point at things, and “split line locked (L)” shows at its foot. Dragging with the left button still moves it, and **L** again frees it. The lock is the same switch as “The split line follows the mouse” in the View group, and it is kept with the comparison.

The difference views

Difference (**O**) switches it on and off; **Y** goes through the kinds) shows what changed, pixel by pixel, after both pictures are on the canvas:

KIND	SHOWS
Difference	The size of the change in each colour, as brightness. Black where the two are the same.
Difference x4	The same, four times brighter, so a change of 5 levels shows as 20.
Signed	Grey where the same; lighter where the right is brighter, darker where it is darker. A tint says which colour moved.

“Brightness only” (**U**) leaves colour changes out. A file compared with itself gives black everywhere: the test routine checked 505,180 pixels and every one was 0.

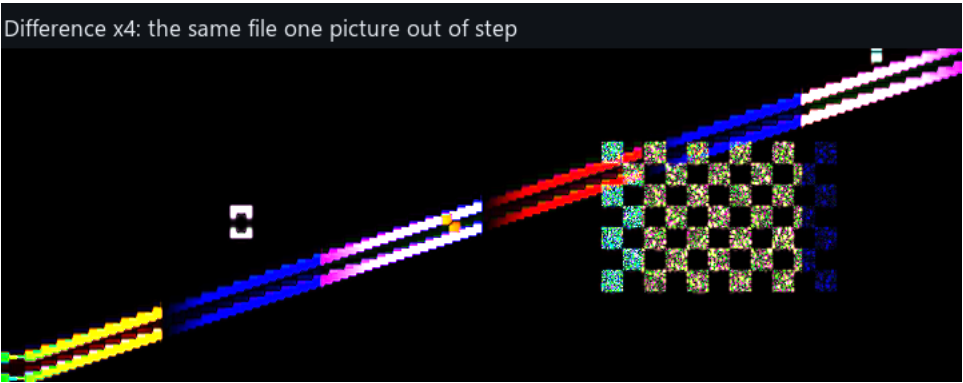


A test pattern against an H.264 copy made at CRF 40. At this size the copy looks like the original; the difference views show where the encoder lost detail, around the small moving squares and the thin lines, and that the flat areas came through unchanged.

Reading the difference views

What lights up, and where, tells you what kind of change you are looking at:

WHAT YOU SEE	WHAT IT USUALLY MEANS
Black everywhere	The two pictures are the same at this moment.
Outlines on edges and fine detail; flat areas black	Compression, sharpening or softening: the change sits where there is detail. Bright outlines in squares of 8 or 16 pixels point at the blocks of a compressed video.
Doubled outlines on everything that moves; still parts black	The two sides are out of step in time. Find the shift (O , Chapter 7), or move it with + and - .
Doubled outlines on every edge, moving or not	The two pictures are not in the same place: one is cropped, enlarged or moved by a few pixels.
An even grey or coloured veil over the whole picture	The brightness or the colour changed everywhere: levels, a colour matrix, a grade. With “Brightness only” on, a colour change disappears and a brightness change stays.
Thin horizontal lines on moving edges	One side is combed (interlaced, shown as stored) and the other deinterlaced. Chapter 8 explains why.
Everything bright	The two sides show different pictures: the wrong files, or far out of step.



The same file on both sides, one picture apart, in Difference x4. Everything that moved between the two pictures is outlined twice; the parts that stand still stay black.

Zoom and the magnifier

The mouse wheel zooms at the pointer; hold the right button and move to pan. **4** shows 1:1 pixels, one canvas pixel per screen pixel; **5** to **9** show 50 percent, fitted, 200, 400 and 800 percent of the fitted size; **R** fits the picture again and **E** moves the point under the mouse to the middle. Past 1:1, pixels stay square unless “Smooth pixels when zoomed in” is on (**T**). **Z** and **C** open a magnifier in the lower left or lower right corner that shows the area around the mouse four times closer; the same key closes it.

Cropping both sides

Shift+B, then drag a box on the picture: both sides are cropped to that part of the canvas and fill the picture. **Backspace** clears the crop. The crop changes only what is shown; the files are not touched, and the scores still cover the whole picture.

Labels, swapping and hiding

The labels at the top of each side come from the files. The switches above the picture choose what they show: **Labels** turns them all on or off; **File name** and **Folder** say which file it is; **Picture** gives its size, frame rate and codec as the file says (“3840 x 2160, 59.94 fps, hevc”); **Scan** gives progressive or interlaced as measured from its pictures; **My labels** adds the names typed under View, Labels on the picture, such as “Original VHS” and “Upscaled”. The first of these is the label, the rest a smaller line under it, in the text size chosen under View; Large reads well when the screen is filmed. The labels go into saved pictures too, and full screen hides the switches. **S** swaps the two sides on screen without saying so in the picture, for a blind test (the left file still sets the time, and the top line says “Sides swapped” unless the text is hidden). **1** and **2** hide the left or right side.

The text on the picture

WHERE	WHAT IT SAYS
Top, in the middle	Whatever is unusual now, separated by bars: the difference view and its kind, a loop, a speed other than 1x, the zoom, Cropped, Sides swapped, Field view, and messages such as a found shift. “Decoding...” means a side’s next picture is not ready yet, usually just after a jump; playback waits for it.
Bottom left and bottom right	The start time of the picture each side shows. The right one adds the shift (shift +33.4 ms) and, with several right videos, which one is on screen (RIGHT 2/3).
Top corners, large	The labels, when they are on.
At the foot of the split line	“split line locked (L)” while the line is locked.
In the middle, after V	Both files’ names, sizes, frame rates, codecs, badges and steps.
In the middle, after H	Every key.
Top right, after X	How many times a second the screen redraws the picture. This is the screen’s rate, not the video’s.

3 hides all of it except the labels, for a clean shot.

Moving through time and lining them up

Playing, stepping on any codec, the loops, the time shift, and finding it automatically

Playing and going to a time

[Space] plays and pauses. **[Left]** and **[Right]** go 1 second back or forward, **[Down]** and **[Up]** 15 seconds, **[Page Down]** and **[Page Up]** 600; **[Home]** and **[End]** go to the start and the end. **[I]** (or **[J]**) and **[J]** slow down and speed up, from a tenth to 8 times. Playback waits rather than skip when a side's next picture is not decoded yet, so every pair on screen belongs together. On the development PC both sides decoded faster than real time: 5.2 times for tape 24's 720 x 480 capture against its 1620 x 1080 copy, 2.6 times for a 720 x 480 file against a 1920 x 1080 one.

One picture at a time

[D] goes to the next moment at which either side changes picture, **[A]** to the previous one. With a 29.97 file on the left and a 59.94 one on the right, the right side changes at every step and the left at every second step; no picture of either side is skipped. Each side keeps the last pictures (the Buffer, 50 by default) so most steps back are instant. Further back, that side goes to the key frame before and decodes forward to the exact picture, so one step back lands on the right picture on any codec, including H.264 and HEVC files with a key frame only every few seconds. On the development PC, going to 100 seconds into tape 24's HEVC capture, which decodes from the key frame before, took 0.6 seconds.

A right picture that starts up to a quarter of a picture after the left picture on screen, at most 1.5 ms, is shown with it. MKV and WebM files keep times in whole milliseconds, so two files of the same pictures can disagree by a millisecond. In one such pair, 190 of 600 pictures of the copy started a millisecond after the original's; before 1.1.1 each of them was shown beside the picture before. The allowance counts from the left picture's start, not from the time you go to, so going to a moment just before a picture still shows the picture before on both sides.

Loops

[,] (comma) loops the pictures in the buffer forward and back, ping-pong, for a close-up you can talk over; **[.]** (full stop) loops them forward. The same key stops the loop and pauses. Auto-loop in the Playback group starts a loop by itself when the buffer is full or the left video ends. The timeline shades the loop.

Lining them up: the time shift

When one file starts a little earlier than the other, move the right video in time. **[+]** and **[-]** move it by one of its own pictures, **[Ctrl]+[Shift]** with them by 10, **[Alt]** with them by 100; the buttons under the picture do the same, and the box takes milliseconds. The shift is added to the left time to find the right picture: a positive shift shows the right video ahead, a negative one behind. The timeline draws the right file's bar moved by the shift, and Worth knowing names it, so a shift left over from an earlier comparison is not missed. Playback > Back in step clears it.

[Ctrl]+[C] copies the left time (0:01:40.017), and **[Ctrl]+[V]** goes to a copied time, in that form or in seconds.

Finding the shift automatically

[O], Find beside the shift, or "Find the shift" under Lining up in time on the information panel works the shift out from the pictures. It reads 8 seconds of the left video from the current time, through the same steps as on screen, and makes each picture 48 x 27 grey pixels. It does the same to the right video across the range chosen either side of the current shift (1 second to 1 minute; 5 by default). For every shift on a grid of half a picture it pairs each left picture with the right picture the screen would show beside it, and scores the pairs on two things: how alike their changes from one picture to the next are, which a still background cannot fake, and how alike the pictures are. The best shift is then moved onto the pictures' own start times, so both sides change picture at the same moments.

A clear match is applied at once, and Undo puts the earlier shift back. When the best is weak, or another shift scores almost as well, nothing is applied and "Apply anyway" is offered. An interlaced side shown as whole frames is matched field by field, so each frame is paired with its first field; matched as whole frames, either field fits equally well.

Reading the offset finder's result

The test routine's clip, against a copy of itself with the first 8 pictures cut off, gives:

Shift found: right -266.9 ms (-8 pictures of the right video). Match 1.00 there, next best 0.87 at -4.800 s, over 240 pictures: a clear match. Applied.

PART	WHAT IT MEANS
right -266.9 ms	The shift now set. Negative: the right file reaches each moment 266.9 ms sooner than the left, because it starts 8 pictures into the same footage.
(-8 pictures of the right video)	The same shift counted in the right video's own pictures. A whole number means the two line up picture for picture.
Match 1.00 there	How well the pairs agree at that shift, from -1 to 1. 1 means the small grey pictures and their changes line up exactly.
next best 0.87 at -4.800 s	The best score more than two pictures away from the one found. A high next best comes from a still stretch or from motion that repeats.
over 240 pictures	Left pictures compared: 8 seconds at 29.97. A side matched field by field counts each field.
a clear match	The match is 0.5 or more and leads the next best by 0.08 or more. Otherwise the result says "NOT a clear match", and offers Apply anyway.

Real footage scores lower than a test clip and is still clear. A 1991 VHS capture saved as MPEG-2, against its deinterlaced copy: right -1.7 ms (-0.1 pictures of the right video). Match 0.61 there, next best 0.36 at +0.043 s, over 471 pictures. The copy was in step already; the 1.7 ms is how far the two files' times disagree, and it pairs the same pictures as no shift. When the result says "The stretch hardly moves", fewer than a fifth of the pictures changed, and only the pictures themselves were matched: go to a

stretch with movement and search again.

The steps on each side

Deinterlacing, the field view, turning, colour, and FFmpeg filters

Automatic steps, one switch each

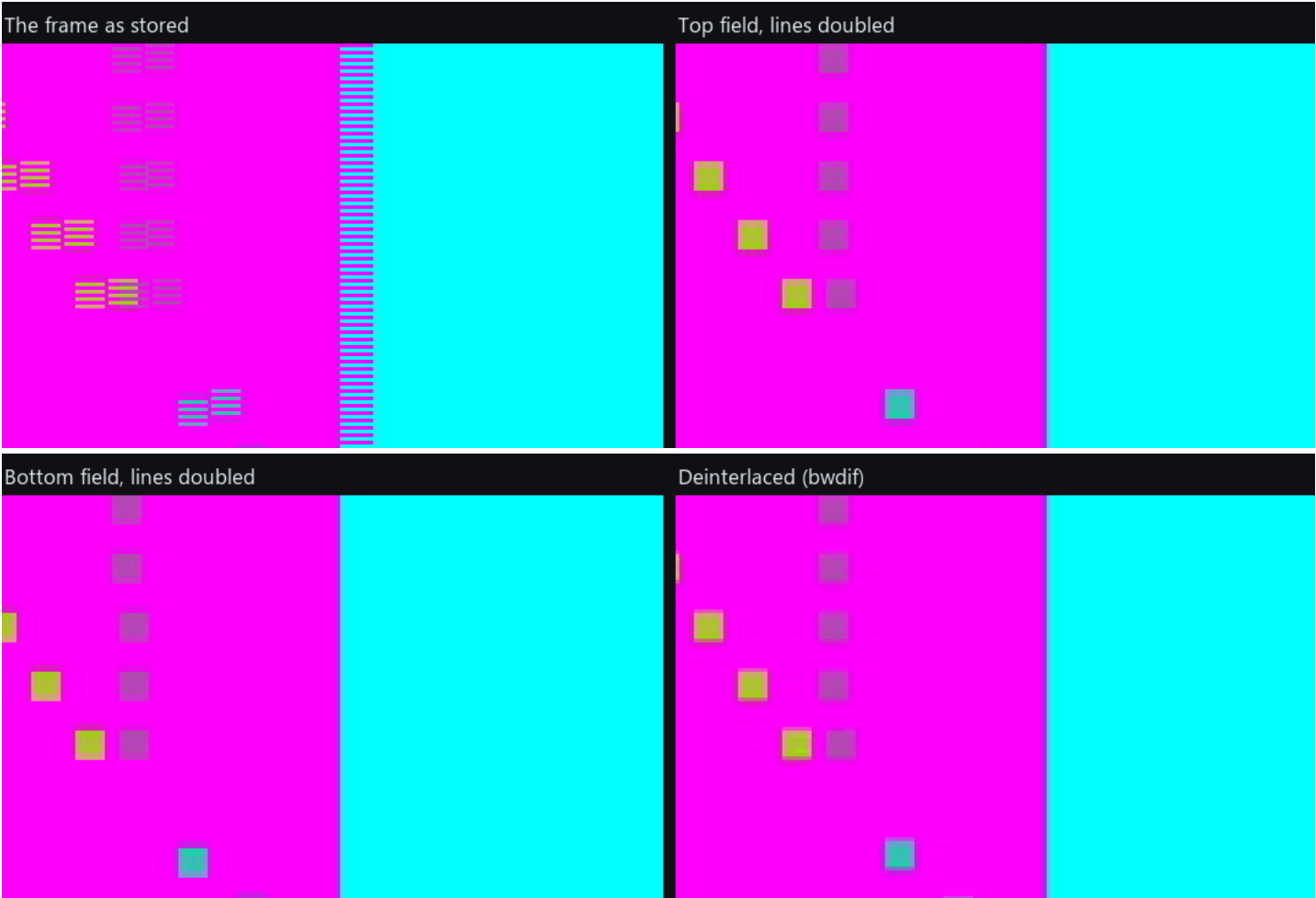
STEP	WHEN	WHAT IT DOES
Deinterlace	The side is interlaced by its badge (by its flag while the badge is measured)	bwdif, in the field order measured. With "One picture per field" (the default) a 29.97 interlaced file gives 59.94 pictures a second, as a television shows it.
Field view	Chosen, and the side is interlaced	Shows the fields as they are, without deinterlacing (below).
Turn as the file asks	The file says how to turn it (phone videos)	A quarter, half or three-quarter turn.
Colour	Always	Each file's own matrix and range, or BT.601 below 720 lines and BT.709 from 720 up when it names none.

Each side has its own switches, so turning deinterlacing off on one side leaves the other side and every other step as they are. That is how to show combing on camera: untick Deinterlace on the left and the capture is shown as the file holds it, while the right side stays as it is. Frame rates need no step, because the clock matches pictures by their time.

What interlaced video is

A camera that records interlaced video (DV, VHS, broadcast television) takes 59.94 pictures a second in NTSC countries (50 in PAL countries), but stores each at half height: one picture, a field, on the even lines of a frame, the next on the odd lines. Each frame holds two fields taken 1/59.94 of a second apart. Where nothing moves, the two fit together into one sharp picture. Where something moves, it was in a different place for each field, and its edges zig-zag from line to line: combing.

Which field was taken first is the field order: top field first or bottom field first. NTSC DV is bottom field first, and the DV tape captures tested for this manual measured bottom first; a VHS capture saved as MPEG-2 measured top first. A deinterlacer has to know the order, which is why the program measures it rather than trust the file's flag.



A test pattern moving sideways, three times closer. As stored, every moving edge and the small squares comb. Each field alone is clean, and the edge sits in a different place in each, because each field is a different moment. bwdif, the deinterlacer the program uses, rebuilds a whole picture from the first field.

The field view

The field view (**G**), or the list under the automatic steps) shows the fields as they are, without deinterlacing, each with its lines doubled so it keeps the frame's height:

CHOICE	SHOWS
Each field in turn	59.94 pictures a second; A and D step one field at a time, 16.68 ms apart. The fields come in the order measured.
The top field only	The even lines, one picture per frame.
The bottom field only	The odd lines, one picture per frame.

It changes interlaced sides only, and takes the place of deinterlacing on them while it is on. Doubled lines look blocky in height: that is the honest picture of what one field holds, half the lines of the frame. The top line of the picture says “Field view” while it is on.

Checking a field order

Choose Each field in turn, go to a stretch where something moves steadily across, and step with **D**. In the right order, the movement goes one way, a little each step. In the wrong order it goes two steps forward and one back, every frame. The field view uses the measured order, so this checks the measurement too: when the movement jerks, the badge's order is wrong for that file.

Judging a deinterlacer

Put the original on the left with Each field in turn, and the deinterlaced copy on the right. Both now show 59.94 pictures a second, and each step pairs a field with the copy's picture for the same moment. The left shows exactly what the deinterlacer was given; the right shows what it made of it. Step through movement and look for:

- **Combing left behind:** thin horizontal lines on moving edges in the copy.
- **Flicker:** fine horizontal detail, such as text or a roof edge, that appears and disappears from one step to the next.
- **Jagged diagonals:** stair steps on slanted edges where the deinterlacer filled the missing lines badly.
- **Ghosts:** a faint second edge, where two moments were blended.

With several right videos (Chapter 4), two deinterlacers can be checked against the same fields, switching with **Tab** at each step.

FFmpeg filters

Under “FFmpeg filters (advanced)”, filter text for both sides, the left only, or the right only goes in after the automatic steps, for example **unsharp** or **crop=640:400:40:40** ; separate several with commas. The text is checked as you type. A filter that cannot be used is left out, and the line under its box says why. The FFmpeg libraries are an LGPL build, which leaves out the filters that need the GPL (hqdn3d and eq among them); the line names such a filter as “not in this FFmpeg build”.

The scopes

The histogram, the vectorscope and the waveform of both sides, and how to read them

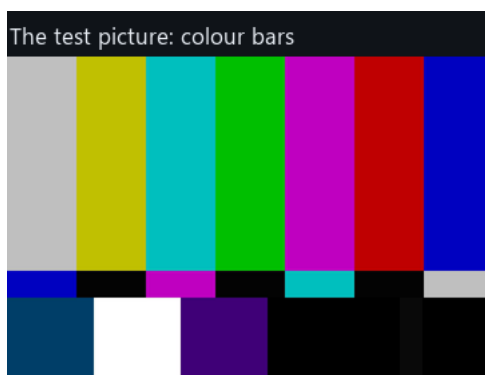
Opening the scopes

F1 opens a histogram, **F2** a vectorscope and **F3** a waveform (View > Histogram, Vectorscope, Waveform do the same). They show the left picture and the right picture on screen side by side, in a window of their own, and follow the picture as it plays or steps. The same key again closes the window; another of the three keys switches it. The window stays on top of the main one unless “Keep on top” is unticked, and passes the comparison’s keys on, so **A** and **D** step while it has the focus. Above each scope: the side, the time of its picture, and its colour conversion. With several right videos, **Tab** changes the right scope with the picture.

Scopes measure what the eye judges badly: whether blacks are crushed or whites clipped, whether a process shifted the colours, whether two copies have the same levels. They read the picture after the side’s steps (deinterlacing, turning, filters), as it is compared.

Levels: 16 to 235, and IRE

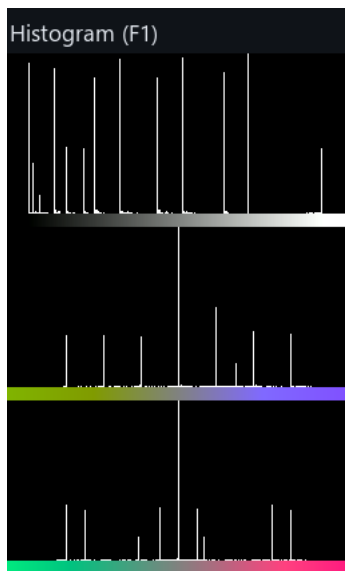
Video almost always stores brightness in “limited range”: on the 0 to 255 scale of 8-bit video, black is 16 and white is 235, with room below and above for overshoot. The waveform’s scale is IRE: 0 at black (16) and 100 at white (235). A trace below 0 is blacker than black, and above 100 whiter than white; many players and conversions cut both off. A “full range” file uses 0 to 255. The scopes convert each picture back with the range and colour matrix it was shown with, so the levels are the file’s own, within one step of 255 from the conversion.



The three scopes below are of this picture: colour bars, a standard test picture of seven colours at 75 percent strength.

The histogram (F1)

Three graphs stacked: brightness at the top, then the two colour channels (blue minus brightness, then red minus brightness). Across, each runs from 0 at the left to 255 at the right; the colour strip under each graph shows what the values mean. Up, the height says how many pixels have that value. The bars picture has a few flat colours, so each is a single spike.

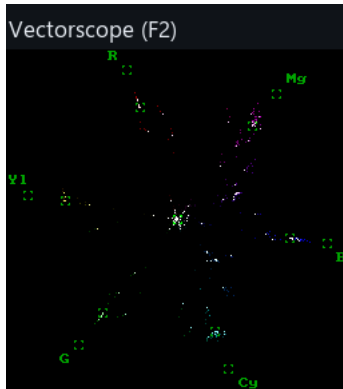


The histogram of the colour bars. Each spike is one bar’s value; in the two colour graphs, the tallest spike, in the middle, is the grey, black and white areas, which have no colour.

- **Brightness piled against an end:** a tall spike at the far left or far right means shadows or highlights cut off flat. When only the copy has the spike, the process clipped them.
- **A comb of empty columns:** gaps between used values mean an earlier step stretched the levels, which leaves some values unused.
- **The colour graphs:** a picture with no colour is one spike in the middle (128). A colour cast moves the bulk of the graph off the middle; a copy that lost colour has narrower colour graphs than the original.

The vectorscope (F2)

Colour only, brightness left out. The angle around the centre is the hue; the distance from the centre is how strong the colour is. Grey, black and white sit in the centre. The six small boxes, named R, Mg, B, Cy, G and Yl, mark where fully saturated red, magenta, blue, cyan, green and yellow fall.

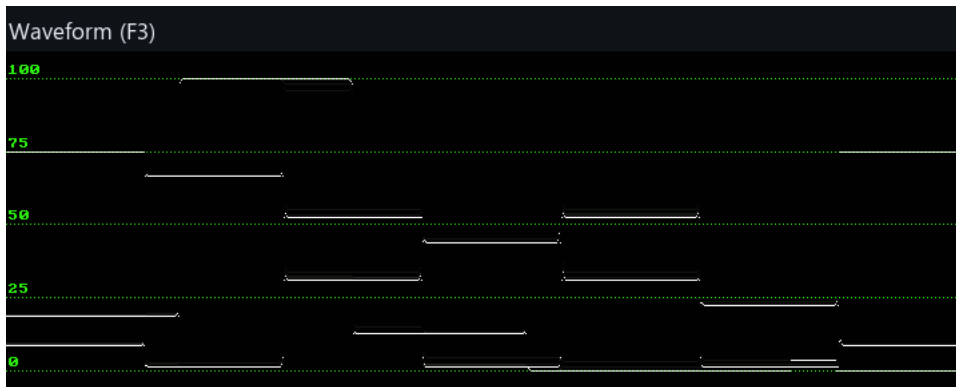


The vectorscope of the colour bars. The bars are 75 percent colours, so each is a dot three quarters of the way from the centre to its box. The dot in the centre is the grey, black and white.

- **A colour cast:** the parts that should be grey sit off the centre, towards the colour of the cast.
- **A hue shift:** the whole cloud turns around the centre. The copy's cloud at an angle to the original's means its colours moved.
- **Saturation:** a copy whose cloud is smaller has weaker colours; one that reaches past the boxes is more saturated than full colour bars.

The waveform (F3)

Brightness only. Each column of the waveform belongs to the same column of the picture: every pixel in that column is plotted at its brightness, 0 IRE at the bottom line and 100 at the top. So the waveform keeps the picture's left and right, and a bright sky across the top of the picture shows as a band high up across the whole width.



The waveform of the colour bars. Each bar is a flat line at its own brightness, stepping down from the grey bar at 75 IRE on the left; the white block in the bottom row reaches 100, and the black areas sit at 0.

- **Footage using its range:** the trace spreads from near 0 to near 100.
- **Clipping:** a trace pressed flat against 0 or 100 over a wide area has lost the detail there.
- **Comparing sides:** a process that lifts the blacks moves the bottom of the copy's trace up; one that adds contrast stretches the trace towards both lines.

Measuring: PSNR, SSIM and VMAF

What each score measures, how the program measures it, and how to read the numbers

What a score is, and what it is not

Each of the three scores answers one question: how far is the right picture from the left picture? The left is the reference and is taken as perfect, so a score can only say how close the copy stayed to the original. It cannot say that the copy looks better. An upscaler that adds believable detail moves the picture away from the original and scores lower for it, even when people prefer the result.

That makes the scores good for four jobs:

- **Checking that a process did no harm**, such as a re-encode or a copy made for an archive, where the copy should be as close to the original as possible.
- **Ranking processed copies** of one original against each other, with several right videos (Chapter 4): the same reference, the same canvas, the same stretch.
- **Finding the worst moments** of a stretch, with "Go to the lowest", so you look at the pictures that suffered most instead of guessing.
- **Catching a comparison that is set up wrong**: scores that collapse usually mean the two sides are out of step or not in the same place.

How the program measures

- 1 **The pairs.** `[M]` measures the two pictures on screen. `[Ctrl]+[M]` measures a stretch: every left picture from the current time for the length chosen under Measurement (1 second to 1 minute), each with the right picture the screen would show beside it at the current shift. A 29.97 left against a 59.94 right gives 30 pairs a second, using every second right picture.
- 2 **The same steps as on screen.** Each side is read through its own steps: deinterlacing or the field view, turning, filters. A stretch is read from the files apart from the screen, so the picture stays where it is while it runs.
- 3 **The canvas.** Both pictures are placed on the canvas as the screen places them (Chapter 5): the smaller one enlarged with the resampler chosen, black bars when each keeps its shape.
- 4 **YUV.** Both are converted to 8-bit YUV 4:2:0 with the left file's colour matrix, which is what FFmpeg's psnr, ssim and libvmaf filters measure.
- 5 **The scores.** FFmpeg's filters give each pair's PSNR, SSIM and, when "VMAF too" is ticked, VMAF. A stretch reports each score's mean, and its lowest with the time it happened.

On the development PC the program's numbers matched FFmpeg's own command-line tool to the hundredth on the same pictures (PSNR 25.19 dB, lowest 25.14 dB, VMAF 100.0 on a test pair). One measurement or search runs at a time, with a progress bar and Cancel on the information panel; `[Esc]` also stops it.

PSNR

Peak signal-to-noise ratio: the average of the squared differences between the two pictures' pixel values, turned into decibels against the largest value, 255. The formula is $10 \times \log_{10}(255 \times 255 / \text{mean squared difference})$. Halving the typical difference (the square root of that mean) adds 6.02 dB. Identical pictures give infinity, shown as "infinite"; in a stretch's mean, an identical pair counts as 100 dB. The program reports FFmpeg's average over the brightness and the two colour channels.

PSNR counts every difference the same, wherever it is and whether or not anyone would see it. A picture made 10 levels brighter everywhere scored 29.89 dB on the snow stretch below, lower than any of the compressed copies, though it looks like the original with the lights turned up.

SSIM

Structural similarity, published by Zhou Wang, Alan Bovik, Hamid Sheikh and Eero Simoncelli in 2004 ("Image quality assessment: from error visibility to structural similarity", IEEE Transactions on Image Processing). It compares small areas of the two pictures for their average brightness, their contrast and their structure, the pattern of light and dark, and averages the result. 1 is identical. The program shows FFmpeg's value for the three channels together, and "brightness" for the brightness channel alone.

Its scale is squeezed towards 1, so read SSIM in the second and third decimals: 0.92 is not "92 percent good". Two unrelated moments of the same test pattern, three seconds apart, still scored 0.9205, because they share large flat areas of the same colour. On the snow stretch below, going from H.264 at CRF 18 to CRF 35 moved SSIM from 0.9852 to 0.9383.

VMAF

Video Multi-method Assessment Fusion: a model trained on scores people gave to compressed and enlarged video, which predicts such a score from 0 to 100 for each pair. It combines measures of how much detail was lost, of how much of the original's visual information survives, and of motion, all computed on the brightness channel. The program uses the models built into the FFmpeg libraries: `vmaf_v0.6.1`, or `vmaf_4k_v0.6.1` for a canvas 2160 lines or taller. VMAF's documentation says the default model was trained on 1080p pictures watched from three times the screen's height, and the 4K model on 4K pictures watched from one and a half times.

Five things about VMAF, each seen in the measurements below:

- **A perfect copy does not score 100.** The motion measure compares each left picture with the one before it, and the less the video moves, the lower even an identical copy scores: 99.9 on the snow stretch, 98.5 on the slow underwater one, and 97.4 for a single pair, which has no picture before it.
- **Its scores stop at 100**, so two copies that both score 100.0 are not told apart.
- **It rewards sharpening.** A sharpened copy scored 100.0 on both stretches, more than the original against itself, while its PSNR on the snow stretch was below the CRF 18 copy's.

- **It punishes softness** far more than PSNR and SSIM do. Blurred slightly (Gaussian, sigma 1), the snow stretch kept a PSNR of 46.77 dB and an SSIM of 0.9942, both above the CRF 18 copy's, while VMAF fell to 81.3 against 95.0; under water, 77.1 against 93.1.
- **It does not see colour.** With every colour turned 20 degrees round the colour wheel, VMAF stayed at 99.9 on the snow stretch while PSNR fell from infinite to 43.78 dB. Check colour with the vectorscope (Chapter 9), with PSNR, or with SSIM's value for the three channels.

Scores from elsewhere. The program measures at the canvas size. VMAF's documentation describes scaling both videos to 1080 lines for the default model, so a score here for a standard-definition capture is not the same number a website publishes for the same footage.

The three side by side, measured

Two 10-second stretches of DV tape captures, deinterlaced to 720 x 480 at 59.94 pictures a second: children sledding in snow (2003, plenty of movement and fine detail), and diving under water (2005, slow, soft and blue). Each was copied without loss, then changed in one known way with the FFmpeg command-line tool, and each change was measured against the copy over all 600 pairs, on a 720 x 540 canvas (the DV pixels made square), with Siggysoft Video Compare 1.1.1. Each cell gives the mean over the stretch.

THE COPY	SNOW: PSNR	SSIM	VMAF	UNDER WATER: PSNR	SSIM	VMAF
Identical	infinite	1.0000	99.9	infinite	1.0000	98.5
H.264, CRF 18	45.83	0.9852	95.0	48.40	0.9907	93.1
H.264, CRF 23	43.56	0.9785	88.9	46.59	0.9885	89.1
H.264, CRF 28	41.01	0.9670	78.3	44.43	0.9849	82.8
H.264, CRF 35	37.03	0.9383	53.4	40.97	0.9767	66.1
Half size and back (bicubic)	47.78	0.9930	96.2	48.04	0.9947	87.9
Blurred (sigma 1)	46.77	0.9942	81.3	48.30	0.9965	77.1
Sharpened (unsharp 5 x 5, amount 1)	45.24	0.9926	100.0	47.04	0.9959	100.0
10 levels brighter	29.89	0.9971	100.0	30.61	0.9961	81.4
Colours turned 20 degrees	43.78	0.9990	99.9	31.79	0.9956	99.6
Identical, one picture out of step	29.21	0.8956	20.2	38.81	0.9778	57.1

The H.264 copies' sizes, for the same 10 seconds: 2,352, 1,112, 592 and 288 kbit/s for CRF 18, 23, 28 and 35 on the snow stretch, and 1,479, 657, 326 and 139 kbit/s under water. What the table shows:

- **The same setting scores differently on different footage.** CRF 35 gave VMAF 53.4 on the busy snow and 66.1 on the slow water. Compare scores within one stretch, not between videos.
- **The three disagree about softness.** On the snow stretch, PSNR and SSIM put the blurred copy above CRF 18 and VMAF puts it well below. Halving the size and enlarging it back scored VMAF 96.2 on the snow, above CRF 18, and 87.9 under water, below it.
- **PSNR is the score that notices levels and colour;** SSIM barely moves for them, and VMAF ignores colour. The brighter copy scored VMAF 100.0 on the snow and 81.4 under water: one change, two verdicts, depending on the picture.
- **Being out of step outweighs every quality change.** One picture off took the snow stretch to VMAF 20.2, far below the worst encode. Line the two up (Chapter 7) before you trust any score.

Upscales: the scores follow the enlargement

When the copy is larger than the original, the original is enlarged onto the canvas before it is measured, with the resampler chosen in Chapter 5. The score then says how close the copy is to that enlargement. Both stretches, enlarged to 1440 x 960 with bicubic, measured against the original enlarged two ways:

THE ORIGINAL ENLARGED WITH	SNOW: PSNR	SSIM	VMAF	UNDER WATER: PSNR	SSIM	VMAF
Nearest neighbour	47.34	0.9954	99.4	49.86	0.9977	95.7
Bicubic	57.80	0.9986	100.0	58.62	0.9988	98.9

The copy is itself a bicubic enlargement, so against a bicubic reference it is almost the same picture, and PSNR rises by 9 to 10 dB. Nothing about the copy changed; only the reference did. To rank two upscalers, measure both against the same original with the same resampler, and keep nearest neighbour unless you have a reason: it is the reference that invents nothing.

Reading a stretch

A stretch's result gives each score's mean and lowest, for example the snow stretch at CRF 28: PSNR mean 41.01 dB, lowest 37.96 dB, SSIM mean 0.9670, lowest 0.9390, VMAF mean 78.3, lowest 65.5, each lowest with its time. The mean says how the copy does overall; the lowest says how bad it gets. A copy with a good mean and one very low picture has a problem at one moment: a scene cut the encoder handled badly, a flash, a dropped picture. "Go to the lowest" goes to the pair with the lowest VMAF (the lowest SSIM when VMAF was not measured), so you can look at it with the difference view.

A VMAF of 0 at one moment, or every score dropping at once, means the two pictures there are very different. In the measurements for this chapter it came from pairs one picture apart. Step through that moment with **A** and **D** and watch the times at the bottom of the picture.

Before you trust a score

- 1 **In step.** Press **O**, or look at the difference view for doubled outlines on moving parts.
- 2 **In place.** No doubled outlines on still edges; when one side is cropped or framed differently, the scores measure the framing.
- 3 **The same steps.** Deinterlacing, the field view and filters apply before measuring. An interlaced original shown combed against a deinterlaced copy scores the combing; with deinterlacing on for both, or the field view on the left, the scores are about the deinterlacer.
- 4 **The same colour.** When the result says the two sides were converted with different colour matrices, the scores include that colour difference. The second line under each file names its matrix.
- 5 **The same reference enlargement.** Keep the resampler the same for every copy you rank.

An example: ranking two encodes

To choose between two encodes of one capture: open the capture on the left and the first encode on the right, add the second with “Add a right video”, and press **O** once on each to line it up. Go to a stretch with movement, choose 10 seconds under Measurement, and press **Ctrl + M**; press **Tab** and **Ctrl + M** again for the second. On the snow stretch above, CRF 23 against CRF 28 gives PSNR 43.56 against 41.01 dB, SSIM 0.9785 against 0.9670 and VMAF 88.9 against 78.3, for 1,112 against 592 kbit/s. All three agree on the order. Then “Go to the lowest” on the smaller one shows where it lost most, and the difference view (Chapter 6) shows what was lost there.

Saving pictures, and the pixels under the mouse

PNG files of both sides and the screen, and the exact values of two pixels

Saving pictures

F, or Save under the picture, writes three PNG files: the left picture and the right picture at their own sizes, after their steps, and the screen as shown, with the labels and lines. They go to `Pictures\SiggySoft Video Compare` (Settings changes it), named after both files and the time, for example `tape 24 vs upscale at 0.01.40.017 - left.png`. Each file also holds both paths, both times, the shift and the steps, as text inside the PNG. **Ctrl**+**Shift**+**C** copies the screen picture to the clipboard.

The pixels under the mouse

Under the mouse, on the information panel, reads the two pictures at the same point of the canvas. Four lines:

LINE	WHAT IT GIVES
Canvas X, Y	The point on the shared canvas.
Left pixel X, Y R G B #RRGGBB	Which pixel of the left picture's own grid lies there, and its red, green and blue from 0 to 255, also as a colour code. On an enlarged side, several canvas points give the same pixel.
Right pixel ...	The same for the right picture.
Right minus left R G B	The difference in each colour: what the difference view draws at that point, as numbers.

P copies the four lines. "outside the picture" means the point is in a black bar or beyond the picture's edge.

Presets and sessions

Ten saved comparisons, and windows that each keep their own

Preset slots

There are a hundred slots: ten banks, A to J, of ten slots each. The letters across the top choose a bank, and the numbers 1 to 10 under them are that bank's slots. The bank shown is lit, a letter in bold holds at least one preset, and the heading names the bank ("Presets, bank B"). The program starts on the bank last shown. Bank A is the ten slots of version 1.2.1 and before.

A preset keeps both files, every right video and every choice. Press "Save to a slot" (it turns amber), then a number; to save into another bank, press its letter before the number, and the button stays amber. Click a number to bring that comparison back. Right-click a number to name it ("Tape 24 deinterlace"), to keep the current comparison there, or to empty it; emptying a slot touches no file. Filled numbers are outlined in blue, and hovering over one shows what it holds. The slots are shared by every window.

Sessions

Each window keeps its comparison, including where it was, and comes back to it at the next start. File > New window asks for a name and opens a second window with a session of its own, so several comparisons can be kept set up at once. File > Reset this session's choices puts every choice back to its first value and keeps the two files.

The PowerShell launcher's presets

At the first start that finds them, the program offers to bring in the presets and sessions of the earlier PowerShell launcher. Empty slots are filled and slots already in use are kept; the launcher's own files are only read. Each old setting becomes the choice that shows the same thing: its honest-baseline box becomes nearest neighbour, and a preset without it becomes bicubic, which is what the viewer it started used. File > Import the PowerShell launcher's presets does it again.

PART C

Reference

Settings, when something does not work, sending feedback to SiggySoft, every key, where the program keeps its files, and the licences.

Settings

Tools > Settings

SETTING	WHAT IT DOES
Where saved pictures go	The folder for the PNG files <code>F</code> writes. Empty: <code>Pictures\SiggySoft Video Compare .</code>
The FFmpeg libraries	Another folder with FFmpeg 9.0 libraries, which must be an LGPL build. Empty: the ones the setup check installs. A change takes effect at the next start, because loaded libraries cannot be swapped.
Look for a new version at every start	Only a copy installed with Setup updates itself. The new version downloads in the background and goes in when the program closes.
Check online once a day for .NET fixes	Reads Microsoft's list of .NET releases (7 KB) for the setup check's .NET line. Nothing about the PC is sent.
Where new versions come from	A folder or an HTTPS address. Empty: the one built into the program.

When something does not work

What the program says, and what to do

WHAT YOU SEE	WHY, AND WHAT TO DO
"The FFmpeg libraries are not installed yet" under a file	Open Tools > Setup check and press Install the FFmpeg libraries (77 MB, once).
"these libraries are a GPL ... build"	The folder named in Settings holds a GPL build of FFmpeg, which the program will not load. Empty the setting to use the libraries the setup check installs.
"Cannot open it: ..." under a file	FFmpeg's own reason: the file is missing, damaged, or in a format this build cannot read. The rest of the program keeps working.
"Not used: ..." under a filter box	The filter text has a mistake, or names a filter this LGPL build leaves out. The other filters and steps still apply.
The picture looks black with coloured edges	The difference view is on: press [D] . Worth knowing names it.
One side seems late or early	A time shift is set: Worth knowing names it, and Playback > Back in step clears it.
The combing is gone from an interlaced capture	Deinterlacing is on for that side (the default). Untick Deinterlace for that side under Automatic steps.
The two sides look one picture apart, or the difference view outlines moving things twice	They are out of step. Press [O] to find the shift, or move it with [+] and [-] (Chapter 7).
"NOT a clear match" from the offset finder	The stretch hardly moves, or something in it repeats. Go to a stretch with movement and press [O] again; widen the range when the files may be further apart.
The scores collapse, or VMAF is near 0	The pairs are not of the same moment or the same place. Line them up first (Chapter 10).
VMAF says 100 but PSNR is low	The copy is sharper, brighter or a different colour: VMAF rewards the first two and ignores colour (Chapter 10).
"converted with different colour matrices" under a measurement	The two files name different colour matrices, or one names none and was assumed. The scores include that difference; the second line under each file names its matrix.
The field view changes nothing	Neither side is measured as interlaced; the field view changes interlaced sides only.
The scopes window says no comparison is open	Choose two videos first; the scopes show the pictures on screen.
"Decoding..." at the top of the picture	A side's next picture is still being decoded, usually just after a jump to another time. Playback waits for it rather than skip.
The setup check says the graphics have no driver	Windows draws the picture itself, slowly. Install the graphics driver from Windows Update or the maker's page.
Something else went wrong	Tools > Open the log folder: the day's log says what the program did, with every error. Feedback > Report a problem sends it to SiggySoft, with the log's last 200 lines when you tick that box (Chapter 15).

Feedback

Reporting a problem, suggesting a feature, asking a question or saying thanks, from the Feedback menu

Sending a report

The Feedback menu, last on the menu bar, opens the same window four ways: Report a problem, Suggest a feature, Ask a question and Say thanks. The kind can still be changed in the window.

- **Title and description.** The title says in a few words what the report is about. For a problem, the description says what happened, what you expected and the steps that lead to it. For a comparison that looks wrong, name the two files and the time on the timeline.
- **Email address.** Optional. Give one for an answer by email, and tick *SiggySoft may email me about this report*. Replies also appear in My feedback without it.
- **System details.** Ticked at first: the Windows and .NET versions, the processor, the memory, the graphics card and the screen size.
- **This program's log.** Not ticked at first: the last 200 lines of SiggySoft Video Compare's log. Your user folder in it is written as %USERPROFILE%.
- **Pictures and files.** Up to 3, 5 MB together: pictures of SiggySoft Video Compare's windows (*Add pictures of this program's windows*), or PNG, JPEG and text files (*Add a file*). The picture of the main window shows the comparison as it is on screen, with the text over it; an open scopes window comes as a picture of its own.

Show exactly what will be sent displays the whole report before anything leaves the PC. *Send* sends it and shows its ticket number, such as FB-2026-000123, with a link to its page on www.siggysoft.org. *Save to send later* keeps it on the PC instead.

My feedback

Feedback, My feedback lists the reports sent from this PC by every SiggySoft program, newest first, with their status (New, Seen, Planned, In progress, Fixed in a version, Won't fix, Duplicate or Closed) and SiggySoft's replies. Refresh asks the site again.

When the site cannot be reached

A report that cannot be sent waits on the PC in %LOCALAPPDATA%\SiggySoft\Footer\outbox. It goes the next time any SiggySoft program starts, or when you press *Send them now* in My feedback, and is deleted from the PC once the site has given it a ticket number.

What is sent, and who sees it

SiggySoft Video Compare sends no feedback in the background: no usage counts and no crash reports. A report leaves the PC when you press Send, or when a saved report goes from the outbox. It travels over HTTPS, signed with a key this PC gets from the site with its first report. The key is kept in %APPDATA%\SiggySoft\Footer\client.json, readable only by your Windows account. A report's page shows its title, status and replies, never your email address, the system details, the log or the pictures. Feedback, What is sent and why lists every part of a report.

Keys

Every key; *H* shows them on the picture, and *Help > Keys* finds one by name

KEYS	WHAT THEY DO
Space	Play or pause
A D	One picture back or forward (either side's next picture; any codec)
Left Right	1 second back or forward; with Shift , a tenth
Down Up	15 seconds back or forward; with Shift , 5
Page Down Page Up	600 seconds back or forward
Home End	The start, the end
[or J ,]	Slower, faster
,	Loop the buffer forward and back; again to stop
.	Loop the buffer forward; again to stop
+ -	Shift the right video one of its pictures later or earlier; with Ctrl + Shift , 10; with Alt , 100
O	Find the shift automatically; a clear match is applied
Ctrl + C Ctrl + V	Copy the time; go to a copied time
Tab Shift + Tab	The next or the previous right video, at the same moment
Ctrl + Shift + 1 to 0	Right video 1 to 10
mouse over the picture	Moves the split line
L	Lock the split line where it is (dragging still moves it); again to free it
mouse wheel	Zoom at the mouse; with Shift , in smaller steps
right button, drag	Move the picture
4	1:1 pixels
5 6 7 8 9	Zoom 50 percent, fit, 200, 400, 800 percent
R E	Fit again; centre on the mouse
Z C	Magnifier in the lower left, lower right; again to close
T	Smooth or square pixels when zoomed in
S	Swap the sides on screen
1 2 3	Hide the left side, the right side, the text on the picture
Shift + M	Next layout
Shift + S	Stretch both to one size, or keep each one's shape
Shift + B Backspace	Crop both sides to a box you drag; clear the crop
G	Field view: each field in turn, the top field, the bottom field, off (interlaced sides)
0 Y U	Difference view on or off; next kind; brightness only
M Ctrl + M	PSNR, SSIM and VMAF of the two pictures; of a stretch from here
F1 F2 F3	Histogram, vectorscope, waveform of both sides; again to close
P	Copy the pixels under the mouse
V H X	The files' details, the keys, the screen's frames per second, on the picture
F Ctrl + Shift + C	Save the left picture, the right picture and the screen; copy the screen picture
F11 or Alt + Enter , Esc	Full screen, the picture alone; leave it. Esc first stops a measurement or a search that is running

Ctrl + + Ctrl + - Ctrl + 0

The window's size; Ctrl and the mouse wheel too

Where the program keeps its files

For the Windows user signed in

WHAT	WHERE
The program, when installed with Setup	%LocalAppData%\SiggySoftVideoCompare
Settings	%APPDATA%\SiggySoft\Video Compare\settings.json
Each session's comparison	%APPDATA%\SiggySoft\Video Compare\session-NAME.json
The preset slots	Bank A: %APPDATA%\SiggySoft\Video Compare\presets.json ; banks B to J: preset banks.json beside it
Logs, one a day	%LOCALAPPDATA%\SiggySoft\Video Compare\logs\videocompare-YYYYMMDD.log
Interlace results	%LOCALAPPDATA%\SiggySoft\Video Compare\scan types.json
The FFmpeg libraries	%LOCALAPPDATA%\SiggySoft\ffmpeg-lgpl
Microsoft's .NET release list	%LOCALAPPDATA%\SiggySoft\dotnet releases-index.json
Saved pictures	Pictures\SiggySoft Video Compare , or the folder in Settings
This PC's feedback key, shared by every SiggySoft program	%APPDATA%\SiggySoft\Feedback\client.json , protected by Windows for your account
Feedback waiting to be sent, and the list of reports sent	%LOCALAPPDATA%\SiggySoft\Feedback\outbox and sent.json beside it

Licences

SiggySoft's terms, and the software it uses under its own

SiggySoft Video Compare is a SiggySoft LLC program, free to use for anyone who received it from SiggySoft LLC or a source SiggySoft authorizes. Help > About shows the whole license and the third-party notices.

COMPONENT	LICENSE	HOW IT IS USED
FFmpeg libraries 9.0.2	LGPL version 3 or later	Downloaded by the setup check, never shipped; loaded into the program. Their source: https://github.com/FFmpeg/FFmpeg/tree/n9.0.2
FFmpeg.AutoGen 9.0.1.1	MIT	The C# declarations for the FFmpeg libraries
Vortice.Windows 3.8.3, Vortice.Mathematics 2.1.0	MIT	Direct3D 11, Direct2D and DirectWrite from C#
SharpGen.Runtime 2.4.2	MIT	Used by the Vortice libraries
Velopack 1.2.161	MIT	The installer and the updates
.NET 10, WPF, System.Management	MIT	The runtime the program runs on
System.Security.Cryptography.ProtectedData	MIT	Part of .NET: keeps this PC's feedback key protected by Windows (Chapter 15)

Coming from the PowerShell launcher

What stayed, and what changed

The earlier launcher, MiddleSiggy's Video Compare GUI, set up a command line for a separate comparison viewer and started it. SiggySoft Video Compare is the comparison program itself, so there is no other program to download or point at, and what used to print to a console window shows on the information panel.

- **The same keys** for the shots worth rehearsing: **0**, **Z** and **C**, **4** to **9**, **,**, **S**, **F**, **Shift + B**, **M**.
- **Changed:** **Ctrl** with **+** and **-** now sizes the window, as in every SiggySoft program, so the 10-picture shift moved to **Ctrl + Shift**. **Esc** leaves full screen instead of closing. A left click on the picture no longer jumps in time; the timeline does that.
- **The honest baseline** is a choice of its own, nearest neighbour by default, instead of a filter added to the left side.
- **The automatic steps** have a switch each, instead of one switch for all of them.
- **Presets and sessions** come in once, when the program first finds them (Chapter 12).

Copyright (c) 2026 SiggySoft LLC. SiggySoft Video Compare is free to use for anyone who received it from SiggySoft LLC or a source SiggySoft authorizes; see the license in Help > About.