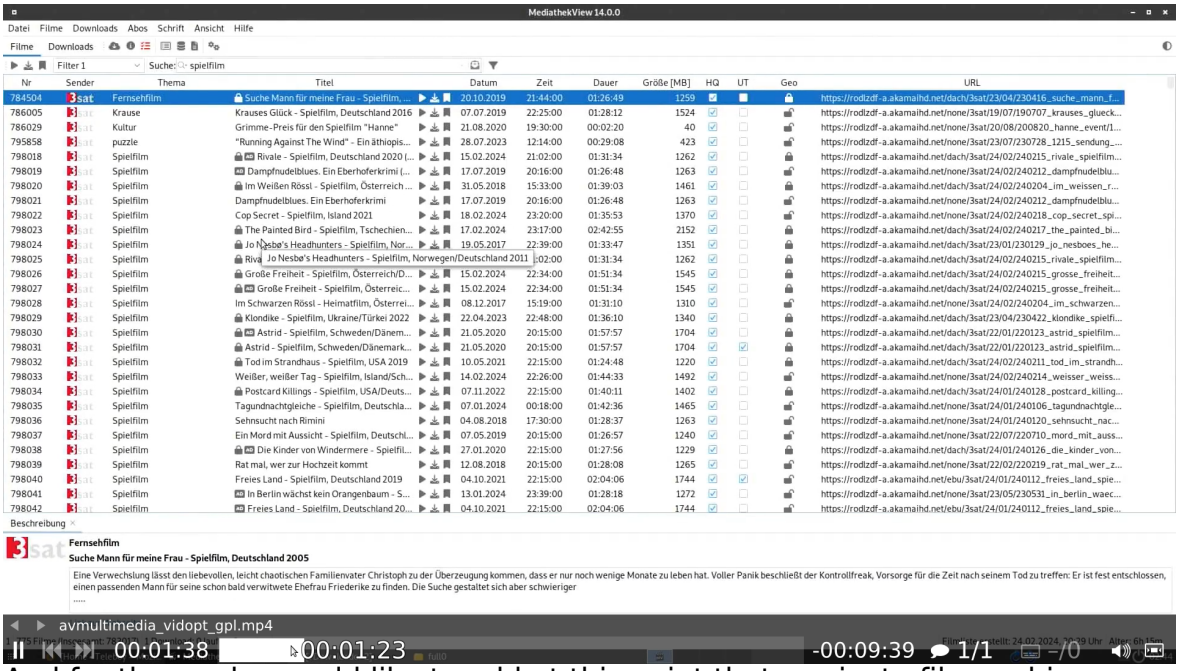
Automated video optimisation

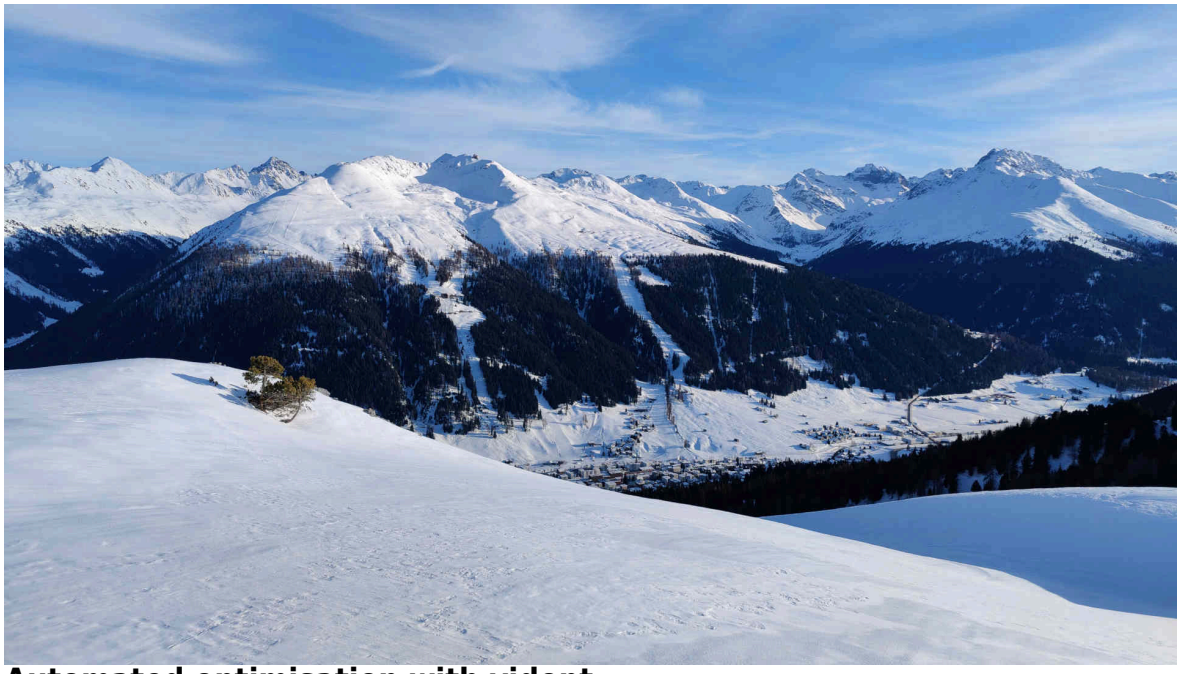
***Egg, 26 February 2024**: For some months now, the ArchivistaBox desktop has offered the option of optimising videos so that they require as little storage space as possible in HD quality. The corresponding tools are now also available in the open source Linux distribution AVMultimedia. The new version 2024/II also includes the 'subtitleadjust' utility to efficiently edit or modify subtitle files.*
https://archivista.ch/cms/wp-content/uploads/file/avmultimedia_vidopt_eng.mp4

Hurdles in the optimisation of videos

Let's start by asking why the solution presented here was created in the first place. Currently, video files can be obtained from many legal sources as part of private copying. Be it via MediathekView, via Internet television such as Yallo.tv or Teleboy.tv or even from YouTube (here the easiest way is via FreeTube). Unfortunately, these files are available in very different qualities.
This often results in considerable file sizes, which, for a private film archive for example, can put a lot of strain on the hard drive. A film downloaded from Teleboy in HD quality can easily take up 6 to 9 GB. Although MediathekView allows you to specify the quality when downloading, the attributes 'High', 'Medium' and 'Low' do not mean that the corresponding videos are only available in the same resolution.



And for those who would like to add at this point that a private film archive can always be stored on a hard drive, it should be noted that a feature film such as 'The Good Shepherd' (obtained via MediathekView) requires approx. 6.5 GB of space. Extrapolated to one TB, this is 130 films. After optimisation with the tools presented here, the same film still requires approx. 1.2 GB.
Of course, video files can also be converted using standard programmes such as FFMPeg or VLC. However, it is not always easy to "get" the right parameters. This is because different parameters are required depending on the quality of the original files.



Automated optimisation with vidopt

The solution presented here was developed for precisely this reason. “Guessing” the optimum values for ffmpeg often worked, but not always. If a film is available in 50 frames/second, it can be compressed quite easily with 25 frames/second. However, 25 frames/second should not be used for files with 29.97 or 30 frames/second. With 60 frames/second, it makes more sense to carry out the optimisation at 30 frames/second.

A film with 1920×1080 pixels can be scaled to HD quality with 1280×720 pixels relatively “safely”. The situation is not quite so simple if the film has 1440×720 pixels. Should it be optimised to 960×480 pixels?

Video files can also contain many audio tracks. Of course, it can make sense to preserve these for films that contain several languages. However, this is not the case if there are two or more audio tracks (e.g. eac3 and aac). In this case, one audio track is sufficient.

The videos are optimised in three steps. Before this can be started, the files to be optimised must be placed in a folder (e.g. /home/archivista/data/in for AVMultimedia). The following three programmes must then be started one after the other:

```
vidopt1 /home/archivista/data/in
vidopt2 /home/archivista/data/in
vidopt3 /home/archivista/data/in
```

It is important to note that vidopt2 can only be started once all files have been processed with vidopt1. The same applies to vidopt3. The console program ‘ffmpeg’ is used for optimisation. If you want to know how these programmes work, you can take a look at the log files.

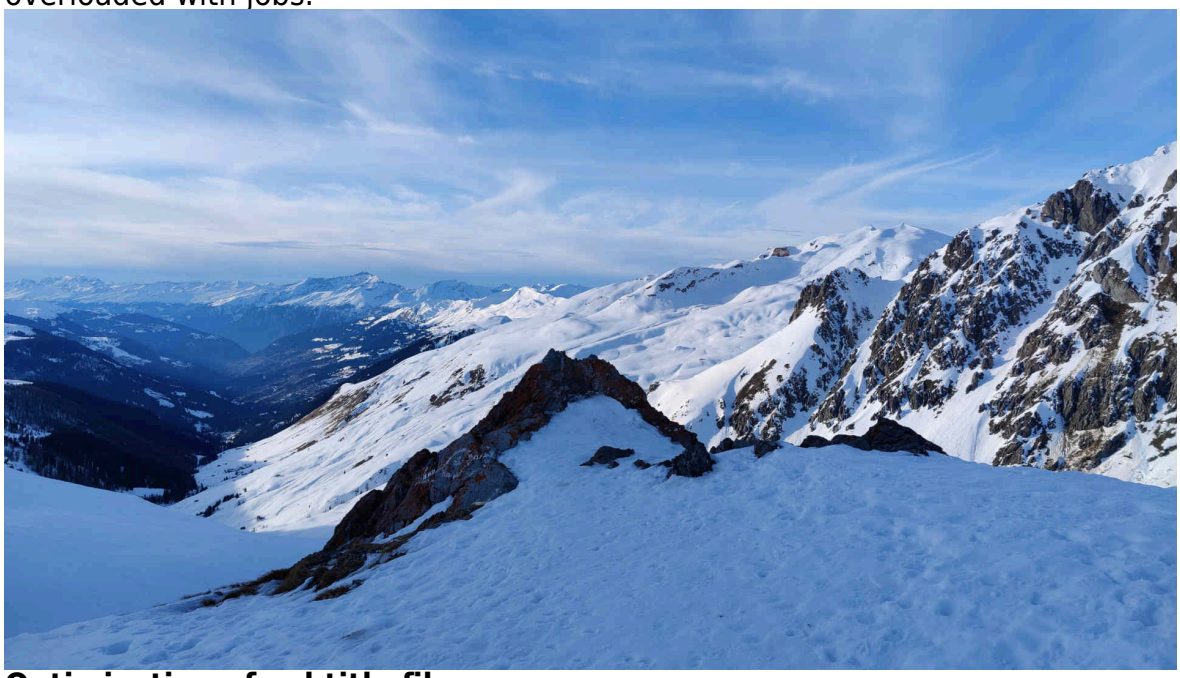
```
tail -f /home/data/archivista/av.log
tail -f /home/data/archivista/vidopt1.log
tail -f /home/data/archivista/vidopt2.log
```

After running the three programs, the optimised videos can be found in the directory. The original film files can be found in the following directory:

```
/home/archivista/data/vidbackup
```

Finally, three comments: Firstly, the optimisation requires considerable computing power. A minimum of 8 CPU cores or 16 threads should be available. Secondly, the more CPU cores, the better, as all CPU cores plus/minus are used for the optimisation. Thirdly, the optimisation only takes place if ffmpeg is not already running. This ensures that the corresponding computer is not

overloaded with jobs.



Optimisation of subtitle files

Newer streams sometimes also have one (or more) “subtitle tracks”. Older films, on the other hand, are often only available in English. German subtitles can often be found for more well-known works (e.g. via opensubtitles.org). However, these subtitles somehow don’t fit. Be it because, for example, the subtitles are available at 24.97 frames/second, but the film runs a little “faster” at 25 frames/second. Even such a small difference results in a difference of 162 frames or 6.48 seconds over a 90-minute feature film (3600*1.5*25.0-3600*1.5*24.97). This is of course enough to “destroy” any cinematic enjoyment.

Subtitle files can also contain the same text passages several times. Here is an example from the Swiss television news programme of 21 February 2021:

```
00:03.600 --> 00:03.620
Mit Live-Untertiteln von SWISS TXT

00:05.200 --> 00:05.220
Mit Live-Untertiteln von SWISS TXT

00:07.380 --> 00:07.400
Guten Abend,
willkommen zur "Tagesschau".

00:08.400 --> 00:08.420
Guten Abend,
willkommen zur "Tagesschau".

00:10.000 --> 00:10.020
Guten Abend,
willkommen zur "Tagesschau".

00:10.640 --> 00:10.660
Das sind unsere Schlagzeilen
von heute Mittwoch:
```

With video players (e.g. mpv), this currently leads to the same subtitles being displayed several times, although one display would be sufficient. When

optimising video files, any subtitles are automatically optimised. The example above becomes:

```
00:00:03.520 --> 00:00:05.409
Mit Live-Untertiteln von SWISS TXT

00:00:07.380 --> 00:00:10.020
Guten Abend,
willkommen zur "Tagesschau".

00:00:10.640 --> 00:00:13.307
Das sind unsere Schlagzeilen^M
von heute Mittwoch:
```

This means that the subtitles are not displayed multiple times (sometimes “flickering”) during playback. The separate ‘subtitleadjust’ tool is called up automatically during video optimization. If a separate subtitle file is available, ‘subtitleadjust’ can also be called up individually. To eliminate any discrepancies in speed or start, there are corresponding options in ‘subtitleadjust’:

```
subtitleadjust in.srt out.srt [change=*x.xxx|+hh:mm:ss|-hh:mm:ss]
```

After the name of the output file, the speed and the start position can be adjusted. Here are two examples:

```
subtitleadjust in.srt out.srt *0.999
```

This results in the time data in out.srt being entered correspondingly slower (correction from 25 to 24.97 frames/second). If the start position is to be shifted forwards by 20 seconds, this can be solved in this way:

```
subtitleadjust in.srt out.srt -00:00:20
```

Sometimes subtitle files also have character sets that are not consistent. The programme subtitlefix8859-1 is available for this purpose. The following command can be used to add a subtitle file to a video:

```
 subtitle vidwithout.mp4 subtitle.vtt videowith.mp4
```

Final remarks

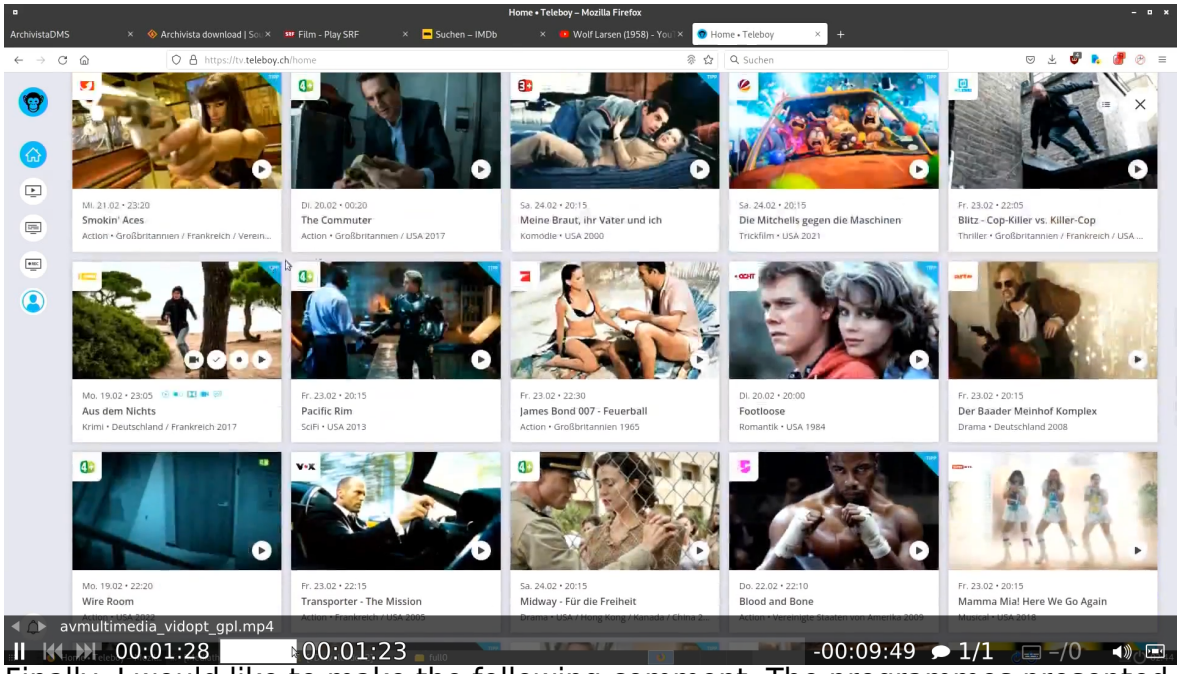
The programmes vidopt1, vidopt2, vidopt3 and subtitleadjust presented here are available in source code on every AVMultimedia distribution under /home/cvs/arcihvista/jobs. The wrappers vidopt1, vidopt2, vidopt3 and subtitleadjust are available under /usr/bin so that they can be called easily. There are also wrappers to adjust video and subtitle files accordingly. The most important tools are

```
videoonly video.mp4 videoooonly.mp4 [tracknr]
videocombine videoonly.mp4 audioonly.mp4 video.mp4
videocombine2 videoonly.mp4 audio1.mp4 audio2.mp4 video.mp4
audioonly video.mp4 audioonly.mp4 [tracknr]
subtitle video.mp4 subtitle.srt|subtitle.vtt
videosubtitle.mp4 [deu|eng|fra]
subtitle2 vidonly.mp4 audio1.mp4 audio2.mp4 sub.srt|sub.vtt
vidout.mp4 [lang]
subtitleonly video.mp4 subtitle.srt [tracknr]
subtitlefix8859-1 subtitle8859-1.srt subtitleutf8.srt
```

All these little helpers are small wrappers that make working with the ‘ffmpeg’ console programme a little easier. An example of this is videocombine2:

```
#!/bin/bash
if [ ! "$1" ];then
  echo "$0 videoonly.mp4 audio1.mp4 audio2.mp4 video.mp4"
  exit 1
else
  in="$1"
  a1="$2"
  a2="$3"
  out="$4"
  ffmpeg -i "$in" -i "$a1" -i "$a2" \
    -map 0:v:0 -map 1:a:0 -map 2:a:0 -c copy "$out"
fi
```

If you often work with ffmpeg, you may not need the utilities. However, for those who do not work with ffmpeg on a daily basis, these tools may be helpful. It should also be noted that there are currently no graphical tools for any of the helpers presented here. However, calling them up via the console is deliberately kept simple so that this should not be a problem in day-to-day work.



Finally, I would like to make the following comment. The programmes presented here comprise approx. 1500 lines of code. The programmes were applied to several thousand video files. The error rate is 1 to 2 per thousand, which were not satisfactorily optimised with the test files. It had to be established that a file was optimised too much or too “weakly”.
Several hundred hours of work were necessary before the tools ran as they currently do. This work was motivated by the fact that it is not practical for a private film archive to have to purchase several hard drives with many dozens of TB.
Compressing videos always leads to certain losses in quality. Measured against the price saved on hardware, this is probably the lesser evil. In the author’s opinion, a home cinema experience can certainly be achieved with a standard projector, even if “only” HD quality (1280×720 pixels) is available. With this in mind, have fun with AVMultimedia 2024/II.
Note: *Customers of the ArchivistaBox should know that these tools are of course also available in the current version of the ArchivistaBox. For the optional applications (e.g. LosslessCut), the flag ‘optos’ can be set for customers. If this is the case, the opt.os file is also installed during the update. And because the flag is available for the first time with version 2024/II, the update to version 2024/II must be carried (this time only) out twice.*