DGS-KORPUS

H.264 and iLex

Author: Sven Wagner, 2014-12-17

Introduction

The current versions of iLex (5.0d50n and 5.1a24) are using the same Mac OS routines as the QuickTime Player 7. This is to be kept in mind.
Our streams are using the Apple ProRes 422 HQ (apch) codec with up to 220 Mbit/s as more or less the raw data of our recordings. But this material has got such an enormous data rate that it can't be handled properly by the current software/hardware combination. Therefore we must compress our ProRes streams. We have chosen the very efficient H.264 codec. iLex has some built-in movie processings tools like cropping, scaling, deinterlacing and resampling as an interface to ffmpeg. The ffmpeg parameters can be set within iLex[1]. To use this built-in movie processing features you need an additional installation of ffmpeg on the computer which is used to perform the movie jobs. We ran in some problems regarding the H.264 codec and routines for QuickTime Player 7, which are used in iLex. Here are some hints to avoid the problems.

Color space and chroma subsampling

The Apple ProRes 422 HQ (apch) coded uses a 4:2:2 chroma subsampling. If you convert such a movie with ffmpeg into an h.264 video, ffmpeg uses also a 4:2:2 chroma subsampling by default. But such a video is showing only black frames in QuickTime Player (version 10.2 and version 7.6.6) and consequently also in iLex even though VLC (version 2.1.5) shows the content of such a movie file correctly. You have to use the ffmpeg parameter '-pix_fmt' to set a different chroma subsampling. It looks like QuickTime Player only plays H.264 video files with a 4:2:0 chroma subsampling. This can be achieved by using '-pix_fmt yuv420p'.

Deinterlacing

Some of our cameras recorded at 25i and some others at 50p. So we had to deinterlace the 25i movies to 50fps. We did this with JES Deinterlacer, but meanwhile you can do that also with ffmpeg. iLex offers you the possibility to deinterlace a movie track via ffmpeg. This is done with the help of the ffmpeg filter '-vf yadif=1'. The result with ffmpeg+yadif is better than the deinterlaced film from JES Deinterlacer. With ffmpeg you have also the option of using deinterlacing filters like w3fdif or mcdeint but these are for our concern to slow in processing.

GOP length

An H.264 stream is organised by groups of pictures (GOP) and every GOP starts with an I-frame also known as key frame. The length of a GOP is the distance between two I-frames. I-

[1] The ffmpeg parameters can be set in the table movie_job_parameters (drop-down menu iLex/Data/Parameters/Movie Job Parameters).

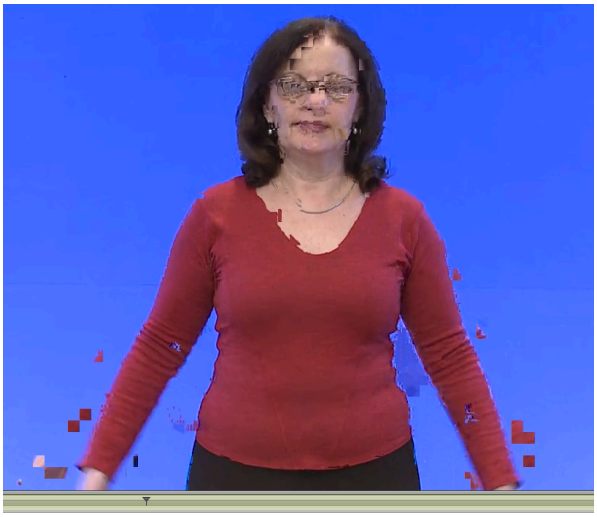
frames are autonomous pictures so they don't use any information from another frame unlike P-frames or B-frames. For seeking purposes the GOP should be short but this will increase the file size. A short GOP allows the user to step through the frames with the arrow keys within iLex. We use a GOP of the length 10. The ffmpeg parameter for this is '-g 10'. The default ffmpeg value is currently 250. A closely related parameter is '-keyint_min' which determines the minimum GOP length e.g. in case of a scene cut. This parameter is set to a tenth of the GOP length by default. So at a GOP of 10 the value of keyint_min is 1.

Video quality

Due to the enormous data rate of the Apple ProRes 422 HQ (apch) the default parameters for quality in ffmpeg do not fit. The primary parameter for quality in ffmpeg for a H.264 encoding is the CRF (Constant Rate Factor) parameter. The default value is currently 23. But this does not always result in nice pictures and causes some quality issues on the GOP borders. On every I-Frame there is a "recalibration" of the pixels and due to the GOP length of 10 there is the impression that the movie is stuttering. To avoid that, you have to increase the quality even if the file size rises. In our case the "recalibration" of the pixels at the GOP boarders are mostly invisible (at the resolutions 1920x1080 and 1280x720) if the CRF value is set to 15. In ffmpeg this is done with the parameter '-crf 15'.

ffmpeg presets

The presets are collections of ffmpeg parameters which affect the encoding speed and the file size. We think the preset 'veryfast' is a good choice between speed and file size. The file size is more or less the same as the preset 'veryslow' but nearly twice as fast. To set a certain preset in ffmpeg use the parameter '-preset veryfast'. You can run into a problem in iLex if you use the preset 'veryslow' and you have a GOP length longer than 42. Then there are some artefacts visible in QuickTime Player 7 and iLex. The following picture is from an ffmpeg movie with the preset 'veryslow' and the default GOP length of 250.



The reason for this is a combination of two parameters in ffmpeg. The preset 'veryslow' sets the parameter for reference frames to 16. If this is set manually with '-refs 8' to the maximum of 8 reference frames the artefacts are gone. In the preset 'veryslow' is set the amount of B-Frames to 8. If this parameter is set with '-bf 9' to a higher number, the artefacts are back. So

it is a combination of those two parameters which are causing the artefacts in QuickTime Player 7 and iLex. QuickTime Player 10 doesn't show these artefacts. If the GOP length is shorter than 43, there will be also no artefacts visible.

Another issue with the presets is, that only the fastest preset uses the H.264 profile 'Baseline'. All other presets are using the H.264 profile 'High'. A major difference between the two profiles is that the profile 'High' allows B-Frames. But this can cause problems with stepping through the movie by the arrow keys. The B-Frames can slow down forward stepping when the GOP length is to high. If you go backwards, it can slow down if there are no B-Frames (ffmpeg parameter '-bf 0' for no B-Frames) allowed. So the best way for a fast seeking is a short GOP length. We recommend a GOP length of 10 (ffmpeg parameter '-g 10').

Hints

The tool MediaInfo[2] shows the ffmpeg parameter set of an H.264 video file. So you can easily compare two encodings with ffmpeg.

Helpful links

https://trac.ffmpeg.org/wiki/Encode/H.264
https://www.ffmpeg.org/ffmpeg-filters.html
http://encodingwissen.de/x264/referenz (German)

[2] https://mediaarea.net/de/MediaInfo