On the Bright Side of Data Migrations

Reto Kromer, AV Preservation by reto.ch, Switzerland

Let's be very clear from the very beginning: I do not consider data migration a good thing at all for the archive community. On the contrary, it costs a lot of time, money, and effort to be achieved accurately. But it cannot be avoided. I will discuss here how data migrations can be used efficiently for modifying, where necessary, the archive's containers, codecs, data and metadata. During the two dozen of data migrations we have carried out for ourselves and our clients, we could actually fix errors in the structure and metadata of the archive, and also we could replace obsolete or endangered formats with current ones. This allows us to change or adjust the strategy when needed. We could update the data and therefore realise maintenance of the digital archive.

1. In the Jungle of File Formats

When I announced that I was going to present on the bright sides of data migration,[1] my colleagues replied to me that it will be a very short speech…five or ten seconds at best! And, indeed, data migration is mostly a bad thing, also because it can be imposed by vendors on archives (when archives have service contracts for software or hardware products) and it can cost a lot of time and money that archives often lack. Inaction is almost never a good choice in the digital domain. Often, it is better to choose an intermediate step, one that can be improved or modified later. Nevertheless, my presentations—and consequently this article—do focus on how a data migration can be used in a positive way, for example to modify where needed the archive's containers and codecs. And I do apologise that my text has a strong personal tone.

In 2002 I began teaching the conservation and restoration of moving images at the Bern University of Applied Sciences. Since that first lecture, every single year I am explaining to the students that in the digital domain it is essential to preserve not only the files, but also at least the source code of the codecs used to create those files. This remains good practice 16 years later, in my opinion. Of course, it is easier when that code is released as open source. Otherwise, it may be much more difficult or even impossible, because one would need to steal intellectual property and/or to reverse engineer to find the applied algorithm. It is completely feasible to crack the code, as it has been done for example for all the ProRes 422 and 4444 variants, but again it costs time and money to the archive. In my opinion, this is indeed a very important point: we as an archival community do need to know in every single detail the file formats we are supposed to preserve. Not each digital archivist needs to be a technical expert, but at least one person in each team should be highly skilled.

1.1 HuffYUV and FFV1

When I left my position as head of preservation at the Cinémathèque suisse in 2003, I established my own conservation and restoration company, AV Preservation by reto.ch, opening both a full photochemical lab and a full digital lab. As at that time no scanner was available so as to be able to gently handle fragile archival material, my team and I often had to build our own equipment, for example by modifying a Truca-like film printer, and sometimes by manufacturing from scratch the machines we needed. Back in 2003 the only possibility I had

1 This paper is the written version of presentations I have given on this topic from November 2017 to April 2018 on several occasions, including No Time to Wait 2 in Vienna, The Reel Thing XLII in New Orleans, Restoration Asia V in Bangkok, the SEAPAVAA Conference in Bangkok, and the FIAF Congress in Prague.

to encode, at a reasonable speed, lossless video during the digitising process was by using the HuffYUV video codec.[2]

Some months later I also started testing FFV1, but I have to admit, that at that time I used more the HuffYUV encoding rather than the FFV1 encoding, because it was a more mature video codec and still today it is the faster one of the two.[3]

We did use the AVI container, because at that time it was the easiest for me to implement in our workflow. And I have to admit that at that time I did not know of the existence of Matroska, which had been born one year earlier, in 2002.

1.2 TIFF, DPX, and OpenEXR

For the single-image based film content, we mainly used, since 2005, TIFF in folders and DPX in XML containers.

In 2013 we started testing OpenEXR. The following year we presented for the first time how the Academy's ACES and OpenEXR can be used in an audiovisual conservation and restoration context. Since spring 2015 our regular digital restoration workflow, which applies to two thirds of our work, is based on these technologies. We do actually use MXF, and today we do it in conformity with AS-07. We have presented our experimentations during a *Late Summer School*, in which we mixed together both high-end photochemical experimentations and high-end digital experimentations.

2 I used intensively the version 2.1.1 which is a very solid one. There was also a 2.2 release, but sadly so buggy that it was simply impossible to use in production. Also a multi-threaded flavour of HuffYUV 2.1.1 did exist, but I never played with that one. Later the FFmpeg team released its own encoder and decoder, supporting a wider variety of pixel formats and bit-depths other than 8 bit. This HuffYUV implementation is actually FFV1's close relative, but that's another story. Today, in parallel with FFV1, we are still using the HuffYUV video codec (in the FFmpeg's flavour) as a native encoding format for digitisation, because it's 25% faster than FFV1 and achieves approximately the same lossless compression rate.

3 My personal love and hate story with the FFV1 video codec is given as a footnote:
- The first Description of the FFV1 Video Codec I have worked with was published by Michael Niedermayer on 9 June 2003. This version, later called 0, for a long time in an experimental stage and is considered stable only since 14 April 2006. Sporadically we did actually use the version 0 in production from the beginning of 2004 on for $Y'C_BC_R$ 4:2:2 video at 8-bit per channel.
- I cannot remember when I came first into contact with the version 1, which is stable since 24 April 2009, but we did use it in production a long time before that date, for 8-bit video capturing, in parallel with HuffYUV.
- FFV1 version 2 existed only in an experimental form and its report was finalised by Michael Niedermayer on 8 April 2012 as FFV1 Video Codec Specification, probably the day after the first draft of version 3 was published with the same title. We generated a dozen of files in both version 1 and version 2 for test purpose, but we never switched from 1 to 2 in production.
- The first FFV1 Video Codec Specification of version 3 I know of was published by Michael Niedermayer on 7 April 2012. This version was then improved and tested in-depth by a large group of people around Michael Niedermayer (including Peter Bubestinger-Steindl, Hermann Lewetz, Georg Lippitsch, Carl Eugen Hoyos and Dave Rice) and is officially deemed stable since 17 August 2013. It is currently being standardised by IETF's CELLAR group. In 2016 my company hired Michael Niedermayer for implementing the handling of 16-bit RGB content. Today, in parallel with HuffYUV, we are still using FFV1 as native encoding for digitisation, because it offers good features to the archive and it's on the way to be officially standardised.
- Since Autumn 2015 I have been suggesting enhancements, including tuning the encoding and decoding performances, implementation of LUTs, handling of Bayer-based «raw» formats, HDR and 24-bit per channel formats. I am happy that CELLAR, while finalising the standardisation of versions 0, 1 and 3, has recently started working on an overdue future version 4 of FFV1 as well.

1.3 Beyond RGB

Fifteen years later, in 2018, we are still carrying out in-depth experimentation and our test implementations on both hardware and software are going in various directions.

The majority of the current mid-range and low-range film scanners use a Bayer pattern sensor. This is not a crime at all. On the contrary, it is often the only possibility that archives can afford. The open source codec CineForm RAW and the new proprietary codec ProRes RAW allow users to store these data forms natively. Today FFV1 cannot store raw Bayer-encoded video and can only store the image content after the additional step of de-Bayering, also called de-mosaicing. FFV1 should implement in version 4 the possibility to handle natively Bayer-based content as well.

Currently FFV1 is limited to the $Y'C_BC_R$ and RGB pixel-format families. There is an increasing need to store multispectral content from moving image as well, as done for many years in other fields of conservation and restoration.[4] In one experimental implementation, for example, we apply a quick-and-dirty hack by storing 15 spectral scans as 5 RGB images. This has many similarities to the Filmic project by Jim Lindner, and some similarities as well to high-dynamic range (HDR). Our approach allows us to store in a transparent way this data today. I am confident that once an open-source file format for multispectral moving images is established, we will be able to trans-multiplex and/or transcode our data in that format.

The research on 24-bit per colour channel is going on both the film industry in Hollywood and the IT and gaming industries in Silicon Valley. Of course, at some point such files will also arrive in the archives and therefore we are experimenting on those formats as well and are testing their appropriate handling.

In the internal context of my company, the adoption of a format in the outside world is not an argument and internally we do use many experimental formats on a daily basis. We simply apply at the end a script or program to generate standard files for the delivery to the clients.

2. Data Migration

In 2014 we migrated our internal archive from LTO-4 cartridges, recorded in proprietary formats and as old-styled TAR archives, to LTO-6 with the more modern LTFS formatting. That was 5.7 PB of data on almost 7,500 LTO-4 cartridges. *En passant*, we also transcoded all the uncompressed $Y'C_BC_R$ 4:2:2 video files into lossless compressed FFV1 and wrapped them into Matroska containers. That has saved between one and two thirds of the required storage, depending on the image content. The whole process resulted with an archive of less than 2000 LTO-6 cartridges.

Then we completed two dozen data migrations for different clients' archives. During almost all those data migrations we actually have fixed errors in the structure of the archive, or in the naming and the metadata of the files, and sometimes we have transcoded obsolete formats into current ones as well. Therefore, we actually have maintained and even updated the archives. For example, in one case, the client wished us to replace the existing MD5 checksums with newer SHA-1 checksums.

4 Hyperspectral imaging can be seen as a kind of generalisation of multispectral imaging. Today's computers cannot handle in an efficient way such content, especially not in the domain of moving image, but to keep an eye on these evolutions can do no harm.

Let us take a look at the workflow. The files are read from the source cartridge. The content is sent to a script or a computer program, which can modify what needs to be modified.[5] Then the writing procedure writes the files onto the destination cartridge. The external script or computer program can do a variety of actions on the data, including modifying the container and/or the codec. It might be the audio codec and/or the video codec (and/or other codecs as well). This feature opens a lot of possibilities. The majority of the changes can be done on the fly, without having to save the files temporarily on a hard-disk drive or a solid-state drive.[6]

Possibilities we actually have used so far are documented in the following sections.

2.1 Change the container

ProRes is a very popular format not only for post-production, for which I assume it was originally designed, but also for capture. Therefore we, as archivists, have to find a way to preserve these native ProRes files, as well as the result of a ProRes-based post-production, because this is actually the highest quality available. QuickTime (.mov) is the natural container of the ProRes video codec. Unfortunately, Apple has already limited the support of QuickTime format for Windows, and most probably they will limit its support on macOS soon too, with the launch of the fully 64-bit operating system. One piece of good news is that the bitstream syntax and decoding process of ProRes have been disclosed and published by SMPTE in 2015, therefore we do have officially some technical information about the format. Another piece of good news is that ProRes can also be wrapped into the Matroska container.

One change we sometimes do apply during data migrations is to transform files from ProRes encoded video that is wrapped in a QuickTime (.mov) container to ProRes encoded video in Matroska (.mkv) files. This is technically called trans-multiplexing (or transmuxing) and consists of the de-multiplexing (or demuxing) of the old files followed by a re-multiplexing (or remuxing) into the new files.[7] This operation is also called re-wrapping.

The file can be trans-multiplexed (i.e., the file is first de-multiplexed and then re-multiplexed) very quickly, because transcoding (i.e., the extremely time-consuming decoding and re-encoding) of the file's content is not required. This can be done easily, if needed, during a data migration, without any additional cost.

5 So far, we have used mainly Bash and rarely C, Mathematica or Go to program the scripts for transcoding and/or trans-multiplexing the data, modifying the metadata or deleting unwanted files, yet almost any programming language should work well.

6 Technically this is realised via classic data pipelines.

7 The raw format for the four flavours of ProRes 422 is the same as previously for uncompressed 10-bit video (technically it is called yuv422p10le), but the encoded image is lossy compressed by the ProRes video codec. This image content is neither uncompressed nor lossless compressed! There are also two flavours of ProRes 4444 which may have 10-bit or 12-bit per channel.

2.2 Change both the container and the video codec

During our own data migration, we have transcoded and trans-multiplexed all the AVI files, containing HuffYUV or FFV1 version 0 or 1 encoded video, into Matroska files, encoded in FFV1 version 3 video for preservation purposes.

For different clients we often have trans-multiplexed and transcoded video content from stream-based $Y'C_BC_R$ 4:2:2 uncompressed 10-bit video in AVI, QuickTime, or MP4 files into lossless compressed FFV1 version 3 video wrapped into Matroska files.

Sometimes we have trans-multiplexed and transcoded from single-image-based DPX or TIFF 16-bit, 12-bit or 10-bit images in MXF, ZIP, or TAR files (or even plain folders) into lossless compressed FFV1 version 3 video wrapped into Matroska or into lossless compressed JPEG 2000 wrapped into an MXF container, usually according to AS-07.

2.3 Other changes

There is almost no limitation of the possible changes during data migration. The range spans from simply to perform actions, such as modifying file naming conventions, to more time-consuming actions, such as modifying the metadata. This includes replacing the MD5 checksums by SHA-1 checksums, or fixing incorrect or missing information about frame rate or colour space, for example.

Data migration can also be used to delete unwanted files such as Apple's .DS_Store and Window's Desktop.ini files which may be written inadvertently onto the source cartridge.

In conclusion, taking account of our experiences so far, I am personally very confident that in the near future, when we will be obliged to migrate our company's archive from LTO-6 to LTO-8, we will be able to replace the experimental containers and video codecs by more robust formats. This may be the case possibly in 2019 or 2020, after the launch of LTO-9, when the prices for LTO-8 will be diminishing.[8]

In my opinion, archives should adopt the formats that they can better master and handle today, knowing full well that their choices are not made forever. On the contrary, they will be able to change, if needed, during each of the coming data migrations at no or little additional cost.

8 Most probably we will then also switch from making three copies to only two copies, yet this topic's discussion is for another article.