detailed approach also when the transfer of rights forms part of an employment contract between the producer of the recording and the live crew.

- Since the area of activity most probably qualifies as part of the 'cultural sector', separate remuneration for each method of exploitation should be stipulated in the contract. If no separate remuneration system has been set up, right holders might at any time invoke the legal default mechanism. This default mechanism grants a proportionate part of the gross revenue linked to a specific method of exploitation to the right holders. The producer may also be obliged to provide an annual overview of the gross revenue per way of exploitation. This clause is crucial in order to avoid unforeseen financial and administrative burdens in a later phase.

- Determine geographical scope and, if necessary, the duration of the transfer for each way of exploitation.

- Include future methods of exploitation in the contract. Although the legitimacy of this kind of clause is not completely guaranteed, we believe future or unknown ways of exploitation can be covered. In order to guarantee the applicability of such a clause, make sure two additional conditions are fulfilled: (1) if the transfer of future/unknown means of exploitation is done within the framework of an employment contract, part of the profit generated through that specific method of future/unknown exploitation should be granted to the right holder and; (2) insert a clause which secures the validity of the rest of the contract in case one clause turns out to be invalid.

table of contents

Digital recording of performing arts: formats and conversion

Stijn Notebaert, Jan De Cock, Sam Coppens, Erik Mannens, Rik Van de Walle (IBBT-MMLab-UGent)
Marc Jacobs, Joeri Barbarien, Peter Schelkens (IBBT-ETRO-VUB)

In today's digital era, the cultural sector is confronted with a growing demand for making digital recordings – audio, video and still images – of stage performances available over a multitude of channels, including digital television and the internet. Essentially, this can be accomplished in two different ways. A single entity can act as a content aggregator, collecting digital recordings from several cultural partners and making this content available to content distributors or each individual partner can distribute its own recordings via the internet. Both methods (content aggregation and individual internet distribution) imply a different set of requirements for audio-visual compression and container formats.

Content aggregation requires high-resolution, high-quality material, suitable for editing/post-processing and conversion to different audio-visual formats tailored to specific distribution channels. Compression and container formats must be chosen so that the content can be processed using (semi)professional production tools. This means that *interoperability* is essential, which implies the use of internationally standardized solutions. Since the material is typically transferred offline, storage and bandwidth limitations are of secondary importance. However, constantly increasing quality and resolution demands and the need for an efficient production chain prohibit the use of older, sub-optimal compression techniques.

For internet distribution, the formats must be flexible enough to support a wide range of terminals (PC, personal media player, smartphone, etc.) and bandwidth-limited connections (xDSL, UMTS, etc.). This specifically implies the use of compression algorithms with a very good rate-distortion performance, (i.e. algorithms which require a minimal number of bits to obtain a given, suitable quality) over a large range of resolutions, frame rates, and quality levels. Additionally, it is of prime importance that the distribution formats are supported by popular playback software.

Establishing a standard set of suitable content aggregation and distribution formats necessitates efficient tools for the conversion of contributed and legacy content to the chosen formats. This conversion is often a time- and resource-consuming process. The efficiency of conversion tools can be improved by performing transcoding instead of re-encoding. Transcoding implies the use of information present in the compressed representation of the original material to facilitate low-complexity conversion to a different format.

In the project, the problems of compression and container format selection and that of efficient format conversion through transcoding have been investigated in great detail. In this chapter, the main conclusions of this work will be presented. A selection of compression formats for video, audio and still images will be discussed. The next section will elaborate on the choice of a proper container format. Finally, the work on transcoding will be summarized.

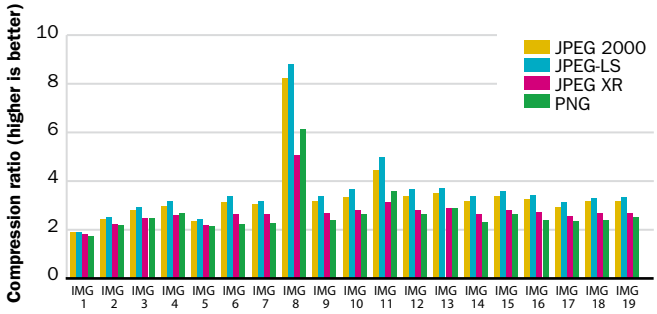
COMPRESSION FORMATS

Still image compression

As mentioned in the introduction, there is a difference in the requirements for the two scenarios, content aggregation and distribution via the internet. The first scenario puts the emphasis on quality, implying the use of lossless image compression, whereas the second scenario must take into account the limitations imposed by the internet connection bandwidth, implying the use of lossy image compression. For the first scenario, the following standardized image coding techniques, operated in lossless mode, were tested: JPEG-LS, JPEG 2000, JPEG XR (HD Photo) and PNG. The evaluation of the compression performance of these codecs is straightforward. The most efficient codec is the one that offers the highest compression ratio, i.e. the size of the uncompressed image divided by the size of the compressed image, for a set of representative test images. In our evaluation, the employed test material consisted of a set of high resolution images related to performing arts productions and exhibiting varying characteristics. The results of our experiments clearly show that PNG has the worst compression efficiency. JPEG XR performs significantly better than PNG but its efficiency is still considerably worse than that of JPEG-LS and JPEG 2000. For black and white images, the performance of JPEG-LS and JPEG 2000 is very similar but for colour images, JPEG-LS outperforms JPEG 2000.

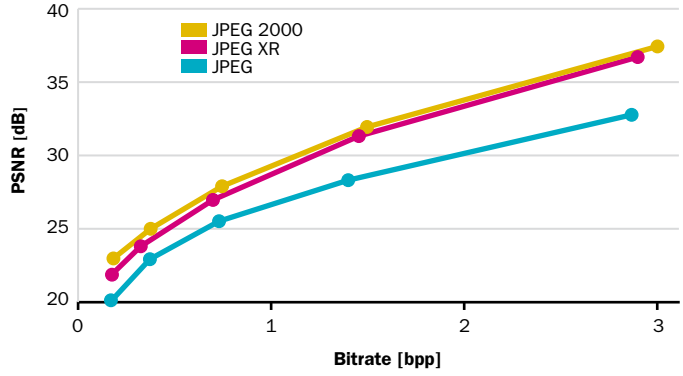
From the viewpoint of professional support, none of the compared compression formats should pose a problem for professional image processing and desktop publishing software. This leads us to conclude that JPEG-LS is the best choice for image compression in the content aggregation scenario.

Figure 1: Compression ratios for lossless still image compression (scenario 1)



For the second scenario, distribution via the internet, the following standardized lossy image coding techniques were tested: JPEG, JPEG 2000 and JPEG XR (HDPhoto). The images used were the same as for the first scenario but their resolution was uniformly reduced to obtain pictures with a maximum height of 768 pixels (typical height of a modern computer monitor). The test images were coded using different bit rates (ranging from 0.1875 to 3 bits per pixel). In each case, the quality of the decoded video material was measured using the peak signal-to-noise ratio (PSNR), which expresses the quality difference between the original and the compressed material. The test results for the second scenario clearly show that classical JPEG coding is no match for the other two image coding techniques. The results for JPEG 2000 and JPEG XR show that JPEG 2000 slightly outperforms JPEG XR. However, the performance difference is limited. Based on its performance, JPEG 2000 is the winner, closely followed by JPEG XR. However, the end-user support for JPEG 2000 is rather limited, whereas Microsoft is actively pushing support for JPEG XR, as shown by the fact that JPEG XR is natively supported in all versions of Windows Vista. As a conclusion, based on its performance and end-user support, JPEG XR is our recommended choice for the internet distribution scenario. Figure 2 presents a typical PSNR graph – showing PSNR values in function of corresponding bit rates – from our tests with lossy image compression.

Figure 2: Typical PSNR graph for lossy image compression (scenario 2)



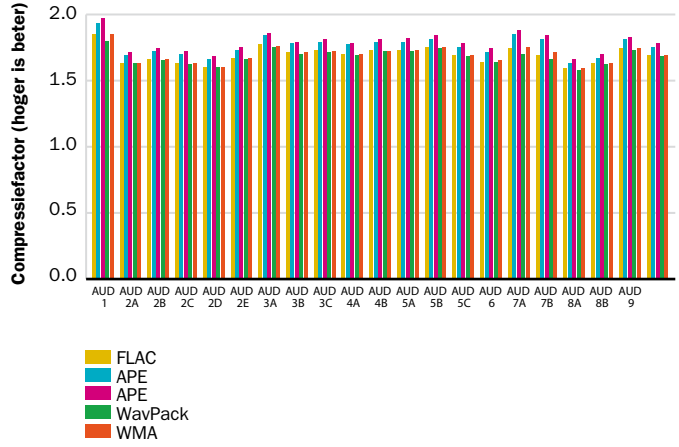
Audio compression

For the content aggregation scenario the following audio compression techniques were evaluated: MPEG-4 Audio Lossless Coding (ALS), Free Lossless Audio Codec (FLAC), Monkey's Audio (APE), Windows Media Audio (WMA) Lossless 9.2 and WAVPACK. Lossless audio compression techniques were selected, since they offer sufficient *size reduction* to be practically useful

in professional environments, while introducing no quality degradation. Apart from MPEG-4 ALS, these compression techniques are not officially standardized and WMA Lossless is even a closed-source audio compression technique. To evaluate the performance of the selected codecs, their compression ratio was compared on different representative audio fragments with varying characteristics. All fragments had two channels (stereo) and a sampling rate of 96 KHz with 24 bits per sample.

The experimental results, graphically presented in Figure 3, show that MPEG-4 ALS yields the best compression performance, closely followed by Monkey's Audio. The third place is shared by FLAC and WMA Lossless. WAVPACK shows the worst performance. Based on compression performance alone, MPEG-4 ALS is clearly the winner. However, the current generation of media applications seems to have only limited support for lossless audio compression in general and for MPEG-4 ALS and Monkey's Audio specifically. FLAC seems to be the most supported lossless audio compression technology, followed by WMA Lossless. Therefore the conclusion is that FLAC is the lossless audio compression technology of choice for the content aggregation scenario.

Figure 3: Compression ratios for lossless audio compression (scenario 1)



The second scenario, internet distribution, warrants the use of lossy audio compression techniques for *bandwidth efficiency* reasons. The following audio compression technologies are commonly used for distributing audio content via the internet: MPEG-2/MPEG-4 Advanced Audio Coding (HE-AAC v2), Microsoft Windows Media Audio (WMA 10 Pro), Ogg Vorbis (aoTuV), and MPEG-1/2 Layer-3 – commonly known as 'MP3'. Unfortunately, the performance evaluation of these techniques poses a significant problem. In the community of audio compression specialists there is no consensus concerning an adequate objective quality measurement tool for audio material. This implies that lossy audio compression techniques have to be evaluated with standardized subjective tests. However, these tests are very complex with very high demands on qualified expert listeners and very strict requirements for the testing

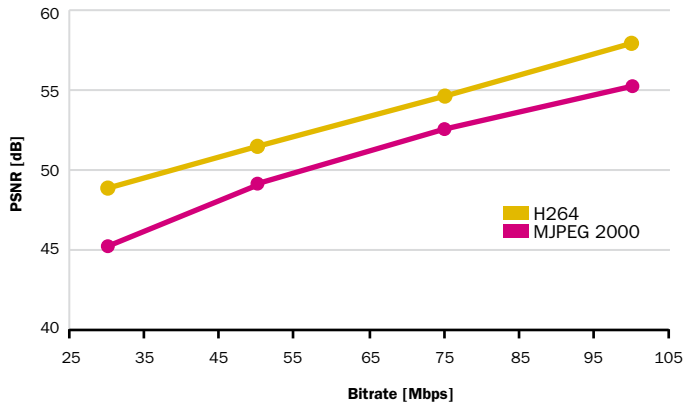
hardware and the testing environment. Because the necessary expert listeners were not available for the testing hardware and testing environment, the tests were replaced by a literature study encompassing test results from subjective tests performed by international organizations such as MPEG and EBU and by independent audio compression experts. Test results presented in the literature indicate that other compression techniques, which have been developed more recently, significantly outperformed the older MP3 codec. Among these newer techniques, there seem to be only small performance differences. On average, HE-AAC v2 seems to show the best performance. The best supported audio compression technology is still MP3, with HE-AAC v2 coming in second. Based on its performance and (end-user) support, HE-AAC v2 is the recommended choice for the internet distribution scenario.

Video compression

Given the requirements detailed in the introduction of this chapter, two video coding techniques, H.264/AVC Intra and MJPEG2000 were selected and compared for use in the content aggregation scenario. Both MJPEG2000 and H.264/AVC are international standards. In both techniques, each frame of the video sequence is coded independently, ensuring optimal edit-friendliness. While, for still images and audio material, lossless compression was advocated for use in the content aggregation scenario, this is not a practical solution for video since it implies unrealistic bandwidth and storage requirements. However, to ensure high quality results, lossy compression with visually imperceptible quality degradation is used, which significantly reduces the storage and bandwidth requirements in comparison to true lossless coding.

To evaluate the compression efficiency of the selected techniques, a performing arts event was captured in 720p HD resolution (1280x720, 50 frames per second), with 4:2:2 chroma subsampling and 10 bits per component, and selected fragments, with varying characteristics, were coded using different bit rates (ranging from 30 Mbps to 100 Mbps). In each case, the quality of the decoded video material was measured using the peak signal-to-noise ratio (PSNR). Figure 4 presents a typical PSNR graph for high resolution, high quality video compression.

Figure 4: Typical PSNR graph for high resolution, high quality video compression (scenario 1)

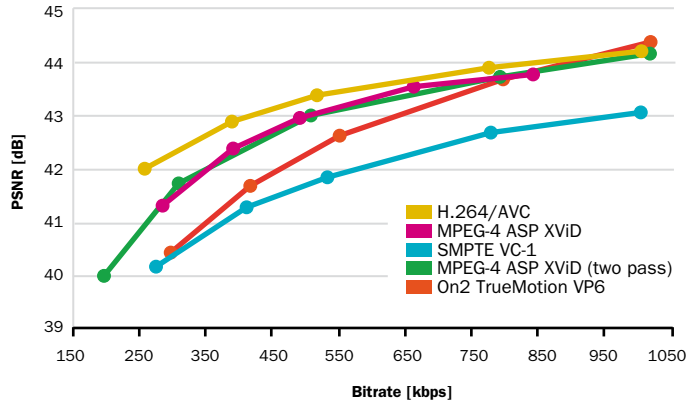


The results show that H.264/AVC Intra performs slightly better than MJPEG2000. However, the differences in objective quality at the same bit-rate are so small that they will not be visible in general. In terms of compatibility with existing and future professional production systems, the situation is not entirely clear yet. This is most certainly the case in broadcast environments.

However, considering its massive adoption in a multitude of other markets, H.264/AVC is likely to receive better hardware and software support. These reasons lead us to conclude that H.264/AVC Intra is the best choice in this scenario.

For the internet distribution scenario, the following coding techniques were evaluated: H.264/AVC, MPEG-4 ASP (XViD, DivX), VC1 (Windows Media 9 Advanced Profile) and Flash (On2 VP6). The first three codecs are open international standards, while the last one is a proprietary solution, which has become a de facto standard in the last few years. All codecs employ temporal prediction to obtain optimal rate-distortion performance. To evaluate the compression performance of these codecs, selected fragments from a recording of the opera *Dialogues des Carmélites* at the Vlaamse Opera in SD format were converted to QVGA format (320x240, 25 frames per second) using professional editing equipment and thereafter compressed at different bit rates ranging from 256 kbps to 1 Mbps. The resulting video quality was again measured using the PSNR (see Figure 5 for a typical PSNR graph). The results show that H.264/AVC generally achieves the best compression efficiency, while VC-1 typically yields the worst performance. The results for MPEG-4 ASP lie somewhere in the middle between those of the latter two codecs. The results for VP6 are harder to interpret. For some sequences, VP6 shows the worst performance, while for others it demonstrates the highest efficiency. The relative performance also seems to vary depending on the target bit-rate. In general, the performance of VP6 should be placed somewhere between that of MPEG-4 ASP and VC-1. The best performing codec, H.264/AVC, is still relatively new. Despite this fact, end-user support for this standard is rapidly growing. On one hand, mature open-source solutions are already available (ffmpeg). On the other hand, a H.264/AVC decoder has recently been added to Adobe's popular and freely available Flash Player 10, ensuring high-quality support for the broad end-user market. Additionally, Apple's popular iPhone and iPod Touch and several mobile phones from Nokia and other manufacturers also offer H.264/AVC support. As a conclusion, based on its performance and end-user support, H.264/AVC is our recommended choice for the internet distribution scenario.

Figure 5: Typical PSNR graph for video compression (scenario 2)



Conclusion compression formats

For the content aggregation scenario, the compression technologies of choice are the following: FLAC for audio content, JPEG-LS for still images and H.264/AVC Intra Only for video content. For the internet distribution scenario, the compression technologies of choice are the following: MPEG-2/MPEG-4 Advanced Audio Coding (HE-AAC v2) for audio content, JPEG XR for still images and H.264/AVC for video content.

CONTAINER FORMATS

While compression formats are designed to compress the multimedia data, container formats, also called wrapper formats, are meta-formats that specify how the (compressed) data is stored in a file or a stream in order to support functionalities such as multiplexing, synchronization, indexing and the addition of metadata. Container formats are typically tailored to a specific type of multimedia material, be it audio, still images, video or a combination of these. Some multimedia container formats, like AIFF, WAV and XMF are exclusively designed to contain audio data. Other containers, like FITS, JP2, JFIF, EXIF and TIFF, are exclusive to still images. Other containers are more flexible and can simultaneously hold many types of audio, video and other data, such as subtitles, metadata, tags, timeline information, and synchronisation information for the playback of the interleaved streams. The most commonly used are 3GP, ANIM, ASF, AVI, CDXL, DVR-MS, IFF, Matroska, MPEG-2 TS, MP4, MOV, Ogg, OGM and Realmedia.

The choice of a multimedia container format requires the thorough evaluation of different aspects of container formats: market support, overhead of the metadata, support for the (advanced) coding features of the intended compression format, support for multiplexing, synchronization and indexing, and finally, the support for streaming media – which requires the data to be stored in chunks inside the container. In the following subsections we elaborate on the choice of the proper container format for the compression formats selected in the previous section.

Still image containers

According to the previous section, the compression technology of choice is JPEG XR for the internet distribution scenario and JPEG-LS for the content aggregation scenario.

The JPEG XR standard defines a feature-complete container format organized as a table of Image File Directory (IFD) tags, similar to a TIFF 6.0 container. A standard JPEG XR file contains image data, an optional planar alpha channel, basic HD Photo metadata stored as IFD tags, optional descriptive metadata stored as IFD tags, optional Extensible Metadata Platform (XMP) metadata encoded in XML and stored as a single IFD tag with extended data, optional Exchangeable Image File Format (EXIF) metadata stored as a sub IFD table linked by an IFD tag, an optional ICC colour profile stored as an IFD tag with extended data. The image data is a monolithic self-contained, self-describing JPEG XR compressed data structure. The optional alpha channel, if present, is stored as separately compressed single channel image data, referenced by the appropriate IFD tags; enabling decoding of the image data independently of transparency data in applications which do not support transparency. In an effort to remain compatible with software designed to decode IFD table-based TIFF files, the largest possible HD Photo file is 4 GB in length. Even though this limit should not raise any concerns in real-life applications, it will be addressed in a future update. Taking all these elements into consideration, it can be concluded that JPEG XR coded images do not need an extra container format for the internet distribution scenario.

JPEG-LS uses a file format that is similar to the JPEG interchange format (JFIF), as it consists of frames, scans, and restart intervals. In fact, JPEG-LS uses the same markers as JPEG (except for a few that do not apply). Moreover, it adds new marker segments containing JPEG-LS specific information, namely specific start-of-frame and start-of-scan marker segments, and an LSE marker segment for preset parameters. In fact, unlike JPEG, parameters have default values that can be overridden by other marker segments. JPEG-LS supports single- and multi-

component scans; in this latter case, a single set of context counters is used throughout all components, whereas prediction and context determination are done independently on each component. The data in the component scan can be interleaved either by lines or by samples. Since JPEG-LS has its own file format, which can foresee most of the required functionality, it is unnecessary to use an additional container format.

Audio containers

Some containers are exclusively designed for audio. The most widespread audio-only container formats are FLAC, WAV, AIFF, and XMF, of which WAV is the most widely used. However, audio is also very often wrapped in multi-purpose multimedia containers, such as Ogg, MP4 or Matroska. This is usually also the case for the compression formats that were withheld before: FLAC for the content aggregation scenario and MPEG-2/MPEG-4 Advanced Audio Coding (HE-AAC v2) for internet distribution.

The open source FLAC development community proposes two alternative containers. The first, also called FLAC, is a very minimalistic audio container, designed to be very efficient at storing single audio streams. The second is the Ogg multimedia container, which enables the mixing of audio, video, metadata, etc. The overhead is slightly higher than that of the native FLAC container format. The FLAC community advises the use of FLAC if only archiving of compressed audio is required. For more advanced purposes it advises the Ogg container. Evidently, other containers also support FLAC. Sometimes, the open source, feature complete multimedia container Matroska is chosen for FLAC encoded audio.

MPEG-2/MPEG-4 Advanced Audio Coding (HE-AAC v2) is supported by many audio and multimedia containers such as 3GPP, Flash Video, Matroska, MP4, MPEG-2 TS, NUT, Ogg, Quicktime, and RMVB. The most interesting of these are the multimedia container MPEG-4 Part 14 (MP4), formerly known as ISO/IEC 14496-14:2003 and Matroska. Both container formats offer a wide variety of functionality and support for many different multimedia compression formats (see further).

Video containers

Video material is usually stored in combination with the corresponding audio tracks, subtitles and metadata in a single container format. Commonly used formats are AVI, MP4, Matroska (MKV/MKA), and MXF.

Audio Video Interleaved (AVI) is a multimedia container designed by Microsoft. AVI containers can store multiple audio and video streams. The format supports nearly all the audio and video formats supported by DirectX and Video for Windows. Subtitles and chapters can also be stored inside the container via modifications outside Microsoft. An AVI container consists of a header with information about the video, e.g., the frame rate, and the actual data.

MPEG-4 Part 14 (MP4) is a multimedia container format that is part of the MPEG-4 standard. MP4 can store multiple audio and video streams. It supports the standard video formats MPEG-1, MPEG-2, MPEG-4, and MPEG-4 AVC and the audio formats (HE-) AAC, MP3, MP2, MP1, CELP, TwinVQ, Vorbis, and Apple Lossless. Except for these audio and video compression formats, MP4 containers can also store private streams. These private streams can hold any kind of information. MP4 also supports storing images, hyperlinks, subtitles, and chapters.

Matroska is an open-source multimedia container format. It is based on EBML (Extensible Binary Meta Language). This is a binary byte-bonded format, based on the principles of XML. Matroska has two versions: MKV, that stores audio and video streams, and MKA,

that can only store audio streams. Matroska containers can hold an unlimited number of audio and video streams. It supports nearly all the current audio and video codecs. Besides audio and video streams it can also store images, subtitles, chapters, DVD-like menus, and even fonts for the subtitles. It also allows streaming.

Material eXchange Format (MXF) is a standard container format for professional audio and video. The format is specified by a set of SMPTE standards. It is an open file format especially designed for exchanging audiovisual material together with the associated data and metadata during the production process. Interoperability is the main goal of MXF. It can be used as a streaming format and as a transferring format. MXF supports nearly all the current audio and video codecs and also permits storing random files. This allows storing transcriptions, images, etc. The MXF container consists of a header, footer and body, which actually holds the data. The header of the container format stores timing parameters, synchronisation information, and metadata. The MXF metadata can store information about the file structure, the title and keywords, subtitles, reference numbers, annotations, version numbers, location, date, etc. To manage the complexity and all the degrees of freedom of the MXF container format, MXF offers some 'operational patterns', or templates.

In the previous section H.264/AVC Intra Only was chosen for the content aggregation scenario while H.264/AVC was the codec of choice for the internet distribution scenario. H.264/AVC Intra Only can be combined with any of the above-mentioned video containers but for maximum professional support this compression format should be combined with the MXF container format. H.264/AVC can also be combined with any of the above-mentioned video container formats. However, a combination with AVI or MXF is not ideal. AVI limits the coding options, e.g. B-frames cannot be supported in a straightforward manner, which results in suboptimal compression and support for MXF on the end-user market is limited. This limits the practical choice of a video container for H.264/AVC to MP4 and Matroska.

Conclusion container formats

FLAC, JPEG-LS and H.264/AVC Intra Only were advised as the compression formats for the content aggregation scenario for audio content, still images and video content respectively. For the internet distribution scenario, the compression technologies of choice are the following: MPEG-2/MPEG-4 Advanced Audio Coding (HE-AAC v2) for audio content, JPEG XR for still images and H.264/AVC for video content. The encoded data should be wrapped in container formats that are fully compatible with the compression formats and support the requirements of the different scenarios. The advised container formats for the chosen compression formats for the content aggregation scenario are:

- Ogg or Matroska for FLAC,
- JPEG-LS does not need an extra container format,
- MXF for H.264/AVC Intra Only.

The container formats of choice for the internet distribution scenario are:

- MP4 or Matroska for MPEG-2/MPEG-4 Advanced Audio Coding,
- JPEG XR does not need an extra container format,
- MP4 or Matroska for H.264/AVC.

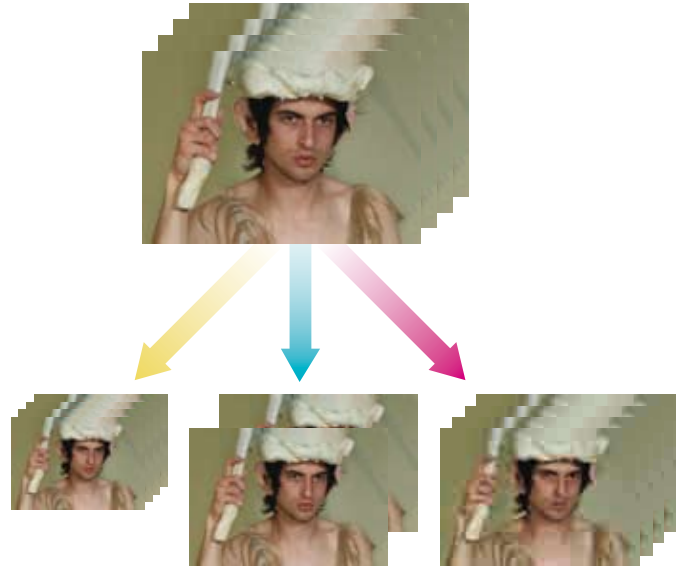
Video transcoding

In order to match the properties and constraints of transmission networks and terminal devices, video transcoding is necessary. Video transcoding can be regarded as the process for *efficient adaptation of video streams*. The information of the incoming video stream is efficiently reused while, at the same time, the quality loss due to the transcoding process is minimized. A number of properties and constraints can be the subject or reason for the transcoding process, such as bandwidth limitations, packet loss, bit rate variation, buffer constraints, display resolution, battery life, etc. The properties and constraints implied by the network or the device typically have an impact on the bit rate, the frame rate, or the spatial resolution. Other types of transcoding exist such as the insertion of new information, i.e. a company logo or a watermark.

A straightforward solution for transcoding is the concatenation of decoder and encoder. Since decoding and encoding is a computationally very demanding operation, this solution is very time-consuming. To overcome this problem, different alternative transcoding solutions have been introduced in the literature that try to 'shortcut' the transcoding process. Reducing the computational efficiency has been a major driving force behind the development of new transcoding solutions.

Efficient transcoding of video streams can be performed by re-using as much of the information as possible from the incoming video stream, and by only changing the required data in the video stream. This means for example, that the motion vectors will be re-used while changes will be made to the residual data (transform coefficients). Another way of reducing complexity is by avoiding algorithmic operations in the transcoding solution. An example

Figure 6: Spatial resolution, temporal resolution and bit rate reduction transcoding



is to work in the frequency domain instead of the pixel domain, hereby avoiding inverse and forward transform operations.

In order to obtain higher compression performance, standardization committees have pushed the limits of coding algorithms in order to identify spatial, temporal, and statistical dependencies in the video stream. As a result, the amount of dependencies in the video stream is severely increased. This means that by changing one syntax element of the video stream, several other elements

can be harmed. Because of the resulting mismatch between the transcoder and decoder, drift can arise in the video stream, and video quality can degrade. Because of this reason, a significant effort related to the development of transcoding algorithms was dedicated to assuring visual quality of the transcoded video streams. Ideally, the transcoded video stream should have the quality of a stream encoded directly with the required parameters. The *problem of drift* together with techniques to stop degradation has been extensively studied in literature.

In this project, we have been investigating bit rate reduction transcoding and temporal resolution reduction transcoding for H.264/AVC video streams.

Bit rate reduction transcoding

The objective of bit rate reduction transcoding is to reduce the bit rate of a video stream while maintaining low complexity and achieving the highest possible quality. Ideally, the quality of the transcoded video stream should have the quality of the video stream directly generated at the reduced bit rate.

There are two classes of techniques for bit rate reduction transcoding, namely requantization transcoding and dynamic rate shaping. Requantization transcoding uses a coarser quantizer while dynamic rate shaping discards high-frequency transform coefficients. In the scope of the project, we selected requantization transcoding for bit rate reduction transcoding.

Different transcoding techniques are proposed in the literature: open-loop requantization, requantization with compensation and the cascade of decoder and encoder. Problems of drift for MPEG-2 transcoding are extensively discussed in the literature. New coding tools in H.264/AVC cause extra problems for transcoding. An evaluation of different techniques for H.264/AVC transcoding is presented in the literature. The main transcoding techniques are briefly discussed and their strengths and weaknesses are indicated:

1) Open-loop requantization is a low-complexity transcoding technique that consists of the following operations: entropy decoding, requantization, and entropy encoding. A number of time-consuming operations are eliminated and fast transcoding becomes possible. The main disadvantage is that the requantization errors propagate and accumulate, which results in increased quality loss. Drift plays an important role in H.264/AVC transcoding and its effect on visual quality will become more severe. Open-loop requantization as such is practically not usable for transcoding.

2) Requantization with compensation is an extension of open-loop requantization. This single-loop architecture calculates the requantization errors and compensates with the requantization errors for both spatial and temporal prediction in order to restrain drift propagation and accumulation. As a result, more processing power and memory buffers are required compared to open-loop requantization; however, this transcoding technique is still faster compared to the cascade of decoder and encoder.

3) The cascade of decoder and encoder is the only drift-free solution for transcoding. This is the most straightforward solution since this is the concatenation of decoder and encoder. In most cases, this solution is not desirable due to the computational complexity as a result of the double-loop architecture. One way to reduce the computational complexity is to reuse the mode and motion data from the incoming video stream. This way, complex processes, such as mode decision and motion estimation, are avoided and significant savings can be made in complexity.

Since none of the transcoding techniques satisfied the requirements, we developed a hybrid architecture that combines different transcoding techniques depending on the picture and macroblock type. This provides a fast transcoding solution that minimizes quality loss due to transcoding. Transcoding tests on performing arts video content have shown that the performance of the hybrid architecture is close to the cascade of decoder and encoder with a significant reduction in transcoding complexity.

Temporal resolution reduction transcoding

The objective of temporal resolution reduction transcoding is to reduce the frame rate of a video stream. In the past, this was often achieved using motion vector mapping. The mapping operation introduces small propagating errors. More recently, other approaches have appeared. These approaches make an appropriate choice for the coding structure. The coding structure allows certain pictures to be dropped without causing errors in other pictures.

Before we explain how to compose these coding structures, we need to further elaborate on picture types. There are three types of pictures: I pictures, P pictures, and B pictures. The I pictures are coded independently and only exploit spatial correlation. These pictures require more bits compared to P or B pictures and are used as random access points in the video stream. The P and B pictures exploit both spatial and temporal correlation. The P pictures only refer to past reference pictures while the B pictures refer to both past and future reference pictures.

In the past, mainly IBBP coding structures were used. The P picture is coded before the two consecutive B pictures. As a result, there is a structural delay on the encoder side of two pictures. When the B pictures are not used as a reference, they can be discarded from the video stream. This leads to a temporal resolution reduction. More recently, hierarchical coding structures have been used. Pictures are organized in temporal layers. A temporal layer only depends on lower temporal layers. As a result, the highest temporal layer can be removed without harming the other pictures.

Conclusion transcoding

Transcoding of H.264/AVC has become more difficult due to new coding algorithms. The transcoding operation should be carefully designed in order to have an optimum trade-off between the computational complexity of the transcoding solution and the visual quality of the transcoded video streams. For bit rate reduction transcoding, a hybrid transcoding architecture is presented that combines different transcoding algorithms. These algorithms are selected based on the picture and macroblock type. This results in a fast transcoding solution that minimises quality loss. We found that the hybrid architecture performs well for performing arts video content. For temporal resolution reduction transcoding, the H.264/AVC specification allows flexibility in the selection of picture types. This way, different temporal resolution reductions can be obtained using different coding structures. This technique does not harm the other pictures in the video stream.

FUTURE WORK

Future work in the domain of digital recording of performing arts will probably be driven by the evolution in compression techniques. In the short term in particular, the evolution in video compression is likely to have a significant influence. More specifically, a lot of work is being put into the development of Scalable Video Coding (SVC) and Multiview Video Coding (MVC).

SVC is the efficient combination of the same video content with a different resolution, frame rate and/or quality into a single encoded video stream. Some of the applications of SVC include a scalable production format, offering both a high resolution, a high-quality editing format and a low resolution, a lower quality browsing format, and a scalable format for distribution over channels with different characteristics (bandwidth, error rate, etc.). MVC allows the efficient representation of multiple views of the same video content into a single encoded video stream. It offers interesting possibilities in applications, such as 3D video and immersive experiences where the spectator can virtually walk on stage during a performance. However, new techniques like SVC and MVC and their possible applications will pose new requirements on container formats and they will increase the need for efficient conversion – transcoding – to application and/or network specific content.

CONCLUSIONS

The goal of this project is a fluent digital dissemination of performing arts content. Two different scenarios were identified to realize this goal, a content aggregation scenario and an internet distribution scenario. This chapter tries to formulate recommendations concerning the optimal choice of compression and container formats for audio, video and still images in the context of these two scenarios. The following table presents the advised combinations of compression formats and container formats for the different scenarios.

Figure 7: Compression and container formats

Scenario	Multimedia files	Compression format	Container format
Content aggregation	Audio	FLAC	Ogg or Matroska
	Still Images	JPEG-LS	/
	Video	H.264/AVC Intra Only	MXF
Internet distribution	Audio	MPEG-2/MPEG-4 Advanced Audio Coding	MP4 or Matroska
	Still Images	JPEG XR	/
	Video	H.264/AVC	MP4 or Matroska

It should be noted that the optimum solutions for the two scenarios differ because the requirements of the scenarios are different. This implies that there is a need for conversion if aggregated content needs to be distributed. The most efficient method for this conversion is transcoding, optimized for specific compression formats and/or applications. In this chapter, a hybrid architecture for transcoding H.264/AVC encoded video with low quality loss was presented.