

File Format for Scalable Video Coding

Peter Amon, Thomas Rathgen, and David Singer

(Invited Paper)

Abstract—This paper describes the file format defined for Scalable Video Coding. Techniques in the file format enable rapid extraction of scalable data, corresponding to the desired operating point. Significant assistance to file readers can be provided, and there is also great flexibility in the ways that the techniques can be used and combined, corresponding to different usages and application scenarios.

Index Terms—File storage, metadata, scalable extraction, Scalable Video Coding (SVC).

I. INTRODUCTION

SCALABLE Video Coding (SVC) is a technique that has been in the signal processing community for already some time. However, only recently, a simple but yet efficient reincarnation of the idea of providing several qualities within a single hierarchically build stream has been achieved, drafted as an amendment to the H.264/AVC standard [1]. It makes use of mostly well known ideas (e.g., pyramidal prediction structures from MPEG-2 [2]) and combines them with a few new techniques (e.g., residual prediction, the key picture concept and single loop decoding) to achieve high compression efficiency at relatively moderate complexity. The next few years will show whether the market embraces the new technology.

In order to fully exploit the new features of SVC, a dedicated and specialized storage format is needed. This paper provides an introduction to the specific techniques provided for handling scalable video streams in the SVC file format specification. A brief introduction is also given to SVC and the general file format (the ISO Base Media File Format) on which the SVC File Format is based, and the techniques are introduced with use-cases and illustrated by examples. The techniques in both the SVC and ISO Base Media File formats are flexible and can be combined and used in a wide variety of ways.

II. SCALABLE VIDEO CODING AND APPLICATIONS

A. SVC Overview

The ISO/IEC 14496-10:2005/AMD3 SVC standard [1] is being designed as the scalable extension of the existing

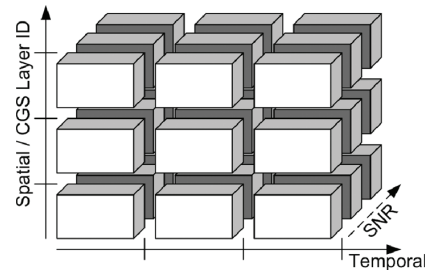


Fig. 1. Data cube model.

H.264/AVC standard. A requirement exists that the SVC base layer shall be compliant to the H.264/AVC standard.

SVC incorporates three scalability modes. Temporal scalability is achieved by hierarchical prediction structure, e.g., using B-frames. If the frames of the highest temporal layer are removed from the SVC stream, then the temporal resolution is reduced (usually by a factor of 2). For spatial scalability, enhancement layers with a higher resolution are coded on top of the H.264/AVC base layer. Inter-layer prediction (e.g., for intra-coded blocks, residual coefficients and motion information) is performed to exploit redundancies between the layers. Fidelity scalability—also referred to as signal-to-noise ratio (SNR) scalability—is achieved in a similar manner as spatial scalability; only the resolution change (downsampling at encoder side and upsampling at decoder side) is omitted and inter-layer prediction is based on coefficients instead of pixel values.

On top of these so-called coarse grain scalability (CGS) layers and spatial layers, medium grain scalability (MGS) layers can be coded. For these MGS layers, the Network Abstraction Layer (NAL) units of one group of pictures (GOP) can be ordered in a rate-distortion optimal way to achieve finer bit rate steps of about 10% [6].

For spatial and SNR scalability, the inter-layer prediction structure is restricted in a way so that only a single motion-compensated prediction loop for the target layer is necessary at the decoder, which reduces decoding complexity. For more details on the SVC standard refer to [4] and [5].

In general, parts of a scalable bit stream can be decoded with reduced quality, i.e., reduced temporal resolution, spatial resolution and/or visual fidelity. The updates from one quality (in one of the scalable directions) to the next higher quality can be seen as elements in a data cube model (Fig. 1). We call all video coding data containing update information from one particular quality to the next quality “belonging to one scalability level.” For scalable video, there are temporal, spatial and SNR levels (see the three dimensions in Fig. 1). A scalability

Manuscript received October 1, 2006; revised July 13, 2007. This work was supported in part by IST European project PHOENIX under Contract FP6 2002 IST 1 001812. This paper was recommended by Guest Editor T. Wiegand.

P. Amon is with Siemens Corporate Technology, Information and Communications, 81730 Munich, Germany (e-mail: p.amon@siemens.com).

T. Rathgen is with the Ilmenau Technical University, Faculty of Electrical Engineering, 98684 Ilmenau, Germany (e-mail: thomas.rathgen@tu-ilmenau.de).

D. Singer works for QuickTime Multimedia Group, Apple, Cupertino, CA 95104 USA (e-mail: singer@apple.com).

Digital Object Identifier 10.1109/TCSVT.2007.905521

level includes the bits for exactly one quality step in exactly one direction. SVC uses a layered coder design to obtain spatial scalability and CGS, indicated by the syntax element *dependency_id*. Temporal scalability is achieved by hierarchical temporal de-composition of each coding layer (indicated by the syntax element *temporal_id*). A picture of a particular coding layer (or *layer picture*) can be refined by up to 15 MGS refinement layers (indicated by the syntax element *quality_id*) to enable SNR scalability.

The coder may choose dynamically which coding layer is used for inter-layer prediction. NAL units that are not used for inter layer prediction of any layer with greater *dependency_id* than the one of the current layer are discardable. Discardable NAL units are signaled with the NAL unit header (see Section II-D).

SVC uses the syntax elements *priority_id*, *dependency_id*, *temporal_id* and *quality_id* (or *PDTQ*) for signaling scalability information within each NAL unit header of an SVC NAL unit. An H.264/AVC NAL unit is preceded by a prefix NAL unit providing this information for the H.264/AVC NAL unit. Generally, *priority_id* may be set by an application according to its requirements representing a valid extraction path; bit stream thinning can be performed by selecting only coded data that satisfies a threshold for *priority_id*. A one dimensional sequence of bit stream operating points represented by successively lower thresholds represents an extraction path. An operating point represents a particular resolution and quality. Each operating point contains a subset of the scalable bit stream that consists of all the data needed to decode this particular resolution and quality.

B. Bit-Stream Representation

A scalable bit stream could be represented in two different ways, as a layered representation (called here “layered scalable”) or providing flexible combined scalability (called here “fully scalable”). In general, there could be more scalability directions, e.g., supporting region of interest (ROI) scalability.

1) *Flexible Combined Scalability*: A scalable bit stream could be organized supporting full scalability. Any valid subset of scalability levels (including the scalability base level) can be extracted from the total bit stream and decoded with the corresponding quality, i.e., any combination of supported resolutions (temporal, spatial or SNR) can be extracted. An SVC elementary stream can be encoded to contain an H.264/AVC compatible base layer (see base layer with *dependency_id* $D = 0$ in Fig. 2). Fully scalable bit streams allow the highest flexibility. The SVC elementary stream itself allows the extraction of any valid substream.

In order to perform an adaptation operation, additional information might be needed to decide which subset out of the available data sets has to be extracted (e.g., depending on the bit rate available). Such adaptation decisions might, for example, be performed based on knowledge of the tradeoff between bit rate and visual fidelity. If such an adaptation operation is performed on a network node, this additional information must be transmitted together with the video data.

2) *Layered Scalability*: Alternatively, a bit stream can be organized in layers. A layer contains all scalability levels to update

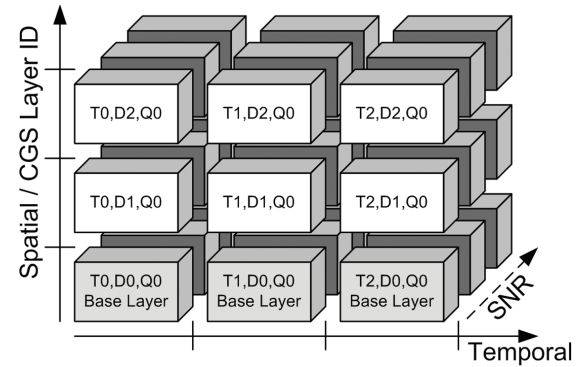


Fig. 2. Fully scalable bit stream representation.

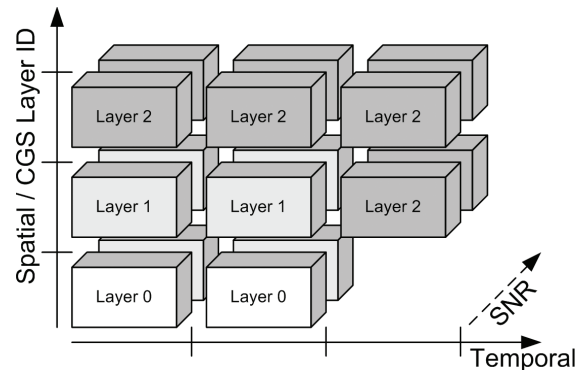


Fig. 3. Layered bit stream representation.

the video from one quality to the next. A layer must enhance the quality in at least one direction—temporal, spatial, or SNR. A layered representation offers simple adaptation operations at defined qualities by discarding unneeded layers. Fig. 3 shows an example of a scalable bit stream organized in three layers. The definition of the operating points is made *a priori* depending on the requirements imposed by an application or by a user or service.

To avoid confusion with the term “layer” as used in the SVC standard, in the SVC File Format scalability layers are referenced as “tiers.”

Since an SVC elementary stream represents the bit stream in the fully scalable representation, a mapping into the layered representation might be performed (e.g., by the streaming server). Depending on the use case, a file reader may choose from one of the offered layered representations and may e.g., set *priority_id* depending on the layer definition (Fig. 4). Adaptation decisions (e.g., during adaptation operations on a network node) can then be based on the scalar layer ID.

C. Usage and Application Scenarios

1) *Direct File Access*: There are three basic access modes to an SVC file: access by an AVC File Reader, bit stream thinning while accessing the file, and accessing the file in order to perform subsequent adaptation operations.

The fact that SVC supports the usage of an H.264/AVC compliant base layer requires the file format to be also AVC compatible. An AVC file reader must be able to access the H.264/AVC base layer when reading an SVC file. Therefore, all AVC File

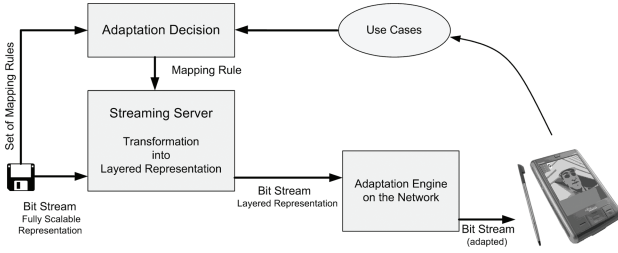


Fig. 4. Mapping into layered representation and adaptation decision.

Format data structures are used as specified for the AVC File Format [10].

A file reader might perform bit stream thinning while accessing the file, e.g., only the data needed for a given operating point is read. The file format provides data to support efficient extraction while accessing the file. This might be necessary when accessing a file with an SVC capable video player to adapt the bit stream to the player's capabilities.

In addition, adaptation operations might be needed on a network node or in a network client. The file format provides data to describe a set of operating points for this purpose. This data can be exported for media transport e.g., using the RTP payload format for SVC [14].

2) *Adaptation Operations*: Adaptation operations consist of an adaptation decision and a thinning operation to discard unneeded data. Depending on the scalability mode, the adaptation decision, e.g., which of the possible operating points gives the best visual quality at a given target bit rate, is a complex operation. An adaptation framework including adaptation decision rules [11] must be provided.

Adaptation decisions for fully scalable bit streams require additional information which needs to be stored in the file format, separate from the video coding data.

Layered scalable bit streams describe a predefined set of operating points on a one-dimensional extraction/adaptation path. Here, adaptation decisions are simple and might be performed easily, e.g., on a simple (i.e., almost stateless) network node. In this case, the information about the layers is e.g., conveyed by the syntax element *priority_id* defined in the SVC specification (see Section II-D).

3) *Erosion Storage*: Surveillance scenarios introduce a special use case. Surveillance video material is often stored on large disk arrays and the quality of the video stream has to be very high. However, after a certain period of time (defined e.g., by legal obligations), the quality may be reduced in order to free storage space. This procedure called "bit stream thinning" can be repeated in order to even reduce further the space used on the storage system. The application taking advantage of such a reduction of the video quality in this step-by-step manner is called "erosion storage."

D. SVC High-Level Syntax

The high-level syntax of SVC obeys similar design criteria as those of H.264/AVC.

Sequence parameter sets (SPS) and picture parameter sets (PPS) containing information for more than one picture are normally transmitted out-of-band using a reliable transmission pro-

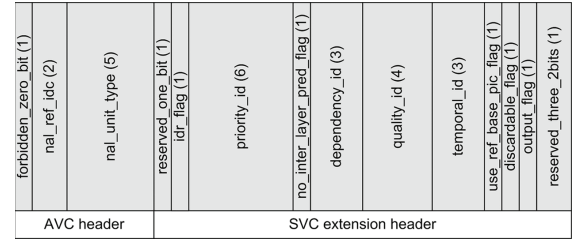


Fig. 5. SVC NAL unit structure [3].

tol (e.g., TCP) in order to ensure that these crucially important pieces of information are available at the decoder.

The pure video data is transmitted in NAL units. The NAL unit syntax of SVC (see Fig. 5) is an extension to the one byte NAL unit structure of H.264/AVC, which mainly contains the NAL unit type distinguishing between e.g., SPS NAL units, PPS NAL units and the video coding NAL units containing different kind of video data (H.264/AVC and SVC NAL units). The first byte of the header extension mainly contains the aforementioned syntax element *priority_id* and also indicates whether the NAL unit belongs to a so-called IDR (instantaneous decoding refresh) access unit (*idr_flag*). The second and third byte provide information on the scalability dimensions represented by the syntax elements *dependency_id*, *temporal_id* and *quality_id*. In addition, the second and third byte of the extension NAL unit header provide information e.g., about the possibility to discard NAL units from the decoding of layers with higher *dependency_id* (*discardable_flag*), whether a NAL unit is coded without using inter-layer prediction (*no_inter_layer_pred_flag*) or if a decoded base picture (i.e., *quality_id* equal 0) is used for inter prediction (*use_ref_base_prediction_flag*).

Most of these pieces of information—especially the scalability information—should also be available at file format level in order to allow adaptation decisions. The mechanisms used for this purpose are described in Section IV.

The NAL unit header is not entropy coded to ensure easy access to the information from a systems layer. It is even used at transport layer as the payload header for the Real-time Transport Protocol (RTP) payload format for H.264/AVC [6] and also for SVC [14], [15].

A further design criterion is the backward compatibility to H.264/AVC. A legacy H.264/AVC decoder regards SVC NAL units as regular NAL units with unknown NAL unit types and therefore discards them while still being able to decode the base layer. However these unknown NAL units might exceed the buffer size indicated by the profile level of the base layer.

III. REVIEW OF FILE FORMAT BASICS

A. ISO Base Media File Format

Within the ISO/IEC MPEG-4 standard, there are several parts that define file formats for the storage of time-based media (such as audio or video). Except from Part 12 itself, they are all based on, and derived from, the ISO Base Media File Format (ISO/IEC 14496-12) [6], which is a structural, media-independent definition and which is also published as part of the JPEG2000 family of standards (as ISO/IEC 15444-12).

The file structure is object-oriented; a file can be decomposed into its constituent objects very simply, and the structure of the objects can be inferred directly from their type and position. The types are 32-bit values and usually chosen to be four printable characters, for ease of inspection and editing.

The ISO Base Media File Format is designed to contain timed media information for a presentation in a flexible, extensible format, which facilitates interchange, management, editing, and presentation of the media. This presentation may be “local” to the system containing the presentation, or may be accessed via a network or other stream delivery mechanism.

The files have a logical structure, a time structure, and a physical structure, and these structures are not required to be coupled. The logical structure of the file is that of a “movie,” which in turn contains a set of time-parallel tracks. The time structure of the file is represented by the tracks containing sequences of samples in time, and those sequences are mapped into the timeline of the overall movie by optional edit lists. The physical structure of the file separates the data needed for logical, time, and structural de-composition, from the media data samples themselves. This structural information is represented by the tracks documenting the logical and timing relationships of the samples, and also containing pointers to where they are located. Those pointers may reference the media data within the same file or within another one, referenced by a URL.

Each media stream is contained in a track specialized for that media type (audio, video, etc.) and is further parameterized by a sample entry. The sample entry contains the “name” of the exact media type (i.e., the type of the decoder needed to decode the stream) and any parameterization of that decoder needed. The name also takes the form of a four-character code. There are defined sample entry formats not only for MPEG-4 media, but also for the media types of other organizations using this file format family. Tracks are synchronized by the media sample’s time stamps. Furthermore, tracks might be linked together by track references.

Finally, tracks may form alternatives to each other, e.g., two audio tracks containing different languages. Tracks which are alternatives have the same nonzero alternate group number in their header, and readers should detect this and make a suitable selection of which one to use. Optional track metadata can be used to tag each track with the “interesting characteristic” that it has, for which its value may differ from other members of the group (e.g., its bit rate, screen size, or language).

Some samples within a track have special characteristics or need to be individually identified. One of the most common and important characteristic is the synchronization point (often a video I-frame). These points are identified by a special table in each track. More generally, the nature of dependencies between track samples can also be documented. Finally, there is a concept of named, parameterized sample groups. These permit the documentation of arbitrary characteristics which are shared by some of the samples in a track. In the SVC File Format, sample groups are used to describe samples with a certain NAL unit structure.

All files start with a file-type box (possibly after a box-structured signature) that defines the best use of the file, and the specifications to which the file complies. These are documented as “brands.” The presence of a brand in this box indicates both a

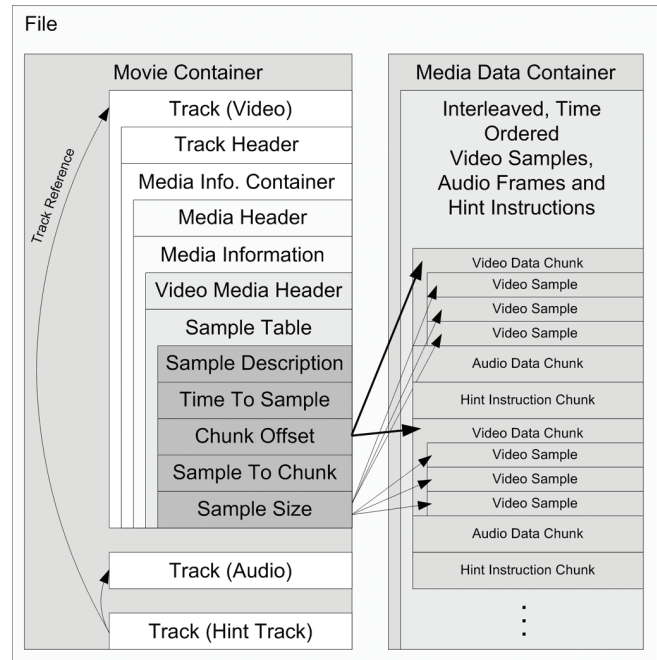


Fig. 6. Example file.

claim and a permission; a claim by the file writer that the file complies with the specification, and a permission for a reader, possibly implementing only that specification, to read and interpret the file.

The “movie” box contains a set of “track” boxes. Each track box contains for one stream:

- 1) its timing information (decoding and composition time tables);
- 2) the nature of the material (video/audio etc.), the coding standard used (H.264/AVC, SVC, etc.), visual width/height information, etc., and the initialization information for that coding standard (sample entry tables);
- 3) information on where the coding data can be found, and its size etc. (sample size and chunk offset tables).

When media is delivered over a streaming protocol, it often must be transformed from the way it is represented in the file. The most obvious example of this is the way media is transmitted over the Real-time Transport Protocol (RTP) [8]. In the file, for example, each frame of video is stored contiguously as a file-format sample. In RTP, packetization rules specific to the video coding standard used must be obeyed to place these frames in RTP packets.

A streaming server may calculate such packetization at run-time if needed. However, there is assistance for the streaming servers. Special tracks called hint tracks, which contain general instructions for streaming servers as how to form packet streams from media tracks for a specific protocol, may be placed in the files. Because the form of these instructions is media-independent, servers do not have to be revised when new codecs are introduced. There is a defined hint track format for RTP streams in the ISO Base Media File Format specification.

The example in Fig. 6 shows a hypothetical file containing three tracks in the movie container (one video track, one audio track and a hint track). Each track consists, among others, of a sample table box with a sample description box. The sample

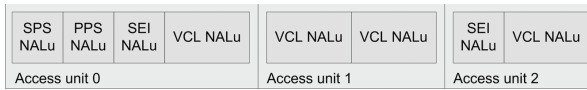


Fig. 7. H.264/AVC elementary stream.

description box holds the information, e.g., contained in the decoder configuration record for H.264/AVC video, needed by the decoder to initialize. Furthermore, the sample table box holds a number of tables, which contain timing information and pointers to the media data. In the example, the video and audio data is stored interleaved in chunks within the media data container. Finally, the third track in the example contains precomputed instructions on how to process the file for streaming.

B. MP4 File Format

The MP4 File Format (ISO/IEC 14496-14) [16] is based on the ISO Base Media File Format. MP4 files are generally used to contain MPEG-4 media, including not only MPEG-4 audio and/or video, but also MPEG-4 presentations. When a complete or partial presentation is stored in an MP4 file, there are specific structures that document that presentation.

MPEG-4 presentations are scenes, described by a scene language such as MPEG-4 BIFS (Binary Format for Scenes). Within those scenes, media objects can be placed; these media objects might be audio, video, or entire subscenes. Each object is described by an object descriptor. Within the object descriptor, the streams that make up that object are described. The entire scene is described by an initial object descriptor (IOD). This is stored in a special box within the movie box in MP4 files. The scene and the object descriptors it uses are stored in tracks—a scene track and an object descriptor track; for files that comprise a full MPEG-4 presentation, this IOD and these two tracks are required.

C. AVC File Format

The AVC File Format (ISO/IEC 14496-15) [5] is based on the ISO Base Media File Format. Not truly a file format in its own, it describes how to store H.264/AVC streams in any file format based on the ISO Base Media File Format, including MP4, 3GPP, etc.

An H.264/AVC stream is a sequence of access units, each divided into a number of NAL units. There are different NAL unit types defined, e.g., video coding layer (VCL) NAL units, Supplemental Enhancement Information (SEI) NAL units (carrying additional information (e.g., on the bitrate) not needed for the decoding process) or parameter set NAL units (Fig. 7). In an AVC file, all NAL units to be processed at one instant in time form a file format sample. The size of each NAL unit (this length indication can be configured as 1, 2, or 4 bytes) is stored within the elementary stream in front of each NAL unit. The size of the entire sample is given in the sample size box.

In the simple use of the AVC File Format, the parameter sets are stored in a configuration record in the descriptive data for the video track (i.e., the sample entry which is contained in the sample description box). Alternatively, if the parameter sets are highly dynamic, a separate parameter set stream may be stored in the file.

H.264/AVC provides means for stream switching. If a sequence is coded to different targets (e.g., bit rates) and these are all stored in one file, then normally one would be able to switch between them at IDR pictures (i.e., I-frames). The H.264/AVC standard also defines switching pictures, which can be used to provide more switching points at lower cost in terms of coding efficiency. The file format contains structures to allow storage of these switching pictures.

IV. SVC FILE FORMAT

A. Design Principles

The SVC File Format is a further specialization of the AVC File Format, and compatible with it. Like the AVC File Format, it defines how SVC streams are stored within any file format based on the ISO Base Media File Format.

Since the SVC base layer is compatible with H.264/AVC, the SVC File Format can also be used in an H.264/AVC-compatible fashion. However, full exercise of the scalability features of SVC encouraged the development of some SVC-specific structures to enable scalable operation.

These extensions fall into three broad groups, differing in the level of detail they cover (and therefore also in the complexity of using them).

- 1) If there are some expected, normal subsets of the scalable stream that will often be extracted, it is possible to define tracks that contain simple instructions on how to form those streams. By following the instructions, a file reader can construct a stream for a particular operating point (i.e., subset) of the scalable stream with very little parsing or structural understanding of the scalable stream itself. These are called *extractor* tracks.
- 2) The data in the stream can be grouped into *tiers*, which contain one or more *scalability layers* (see Section II) of the scalable stream. Each tier has a description, and all the data in the stream can be mapped to a specific tier. If decisions about scalability can be made on the basis of the tier descriptions, then these structures can be used to select the tiers of interest, and discover rapidly the data that is associated with those tiers. The descriptive data in this case is not timed; only the mapping from the coding data to the descriptions is timed. This technique uses sample groups.
- 3) Finally, the data in the scalable stream can have time-parallel data associated with it, providing exact information about the associated video coding data. In this case, the descriptive data itself is timed, and can vary on a time-basis. This technique uses a time-parallel metadata track.

Finally, of course, scalable operations can be performed, if needed, by parsing the SVC coding data itself.

Scalable video data is stored as one or more tracks. There is a set of tracks that contains the entire scalable stream (the complete set). In a simple use of the file format there would be one track that contains the entire scalable stream.

If there is more than one track representing part or all of the SVC stream, then the client is instructed to choose one of them, by placing them all into an “alternate group,” as described above for the ISO Base Media File Format.

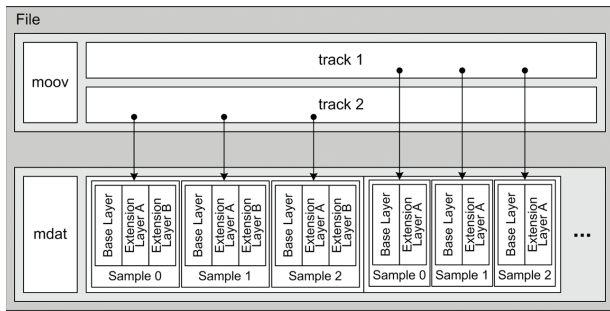


Fig. 8. Two tracks duplicating data.

B. Extractor Tracks

The first technique mentioned above allows “cookbook” construction of expected extractions of the scalable stream. These take the form of tracks within the alternate group.

In the case of nonscalable coding, each track has a unique copy of the video coding information needed for that operating point. Clearly, in the case of scalable coding, that information is already present in the track(s) that form the complete subset. Extractor tracks provide a way to share that data and therefore do not enlarge the file excessively.

These tracks are structured exactly like SVC video tracks. However, they permit the use of an in-line structure, specific to the file format, structured like a NAL unit, called an *extractor*. Extractors are pointers that provide information about the position and the size of the video coding data in the sample with equal decoding time in an other track, which is very much like hint instructions. This allows building a track hierarchy directly in the coding domain. An extractor track is linked to one or more base tracks, from which it extracts data at run-time. An extractor is a dereferenceable pointer with a NAL unit header with SVC extensions. If the track used for extraction contains video coding data at a different frame rate then the extractor also contains a decoding time offset to ensure synchrony between tracks. At run-time, the extractor has to be replaced by the data to which it points, before the stream is passed to the video decoder.

This means that, since the extractor tracks are structured like video coding tracks, they may represent the subset they need in different ways.

- 1) They contain a copy of the data (Fig. 8).
- 2) They contain instructions on how to extract the data from another track (Fig. 9).
- 3) They copy some data and contain instructions on how to extract other data from another track.

The three options above have different characteristics. The first duplicates data, and thus makes the file overall larger but keeps access and extraction simple. The second keeps the storage of media data and the metadata compact, however the reader must load the data for both tracks and dynamically do the extraction.

The third is a hybrid technique, which is also possible. Which choice is appropriate is dependent on the application, usage, file size, and other considerations.

C. Sample Groups

If the “cookbook” extractions offered by the extractor tracks are not sufficient, then use can be made of sample groups.

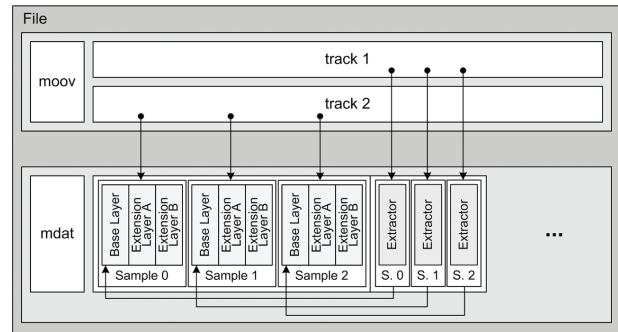


Fig. 9. Track 1 using extractors.

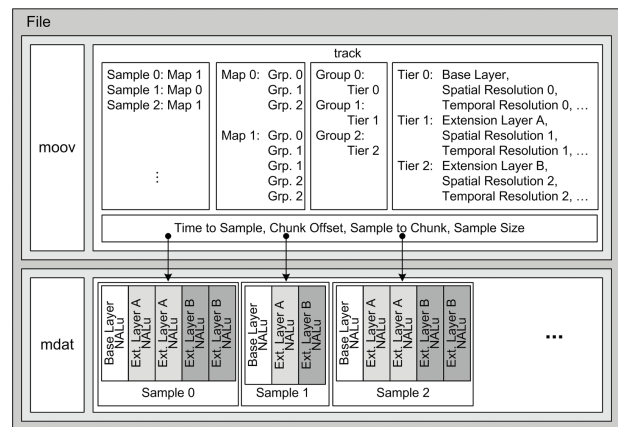


Fig. 10. Double indirection using maps and groups.

As defined in the ISO Base Media File Format, sample groups provide two structures.

- 1) A number of sets of description tables; each set has a grouping or description *type*, and each member of the set contains a description of that type.
- 2) A number of mapping tables. Each mapping table has a grouping or description type, and defines a mapping from each frame in the track to the description of that type (by index).

This enables dividing the samples in a track into a few groups, each of which has a description.

However, in SVC, each file format sample is composed of several layers. Since each layer does not always appear with the same other layers, there is an issue, if the descriptions apply to whole samples: each layer must be described multiple times, in combination with the other layers with which it might appear. This both duplicates descriptions and multiplies the number of entries.

To alleviate this, a second level of indirection is introduced. Instead of associating each file-format sample directly with a description, it is associated with a *map*. Each map describes the group structure of the samples with which it is associated; for example (see Fig. 10) “all samples associated with map 0 start with a NAL unit for group 0, then two NAL units for group 1, and finally two NAL units for group 2.” Each H.264/AVC NAL unit and its corresponding prefix NAL unit are logically treated as a single NAL unit. A second set of tables contains the descriptions of the tiers. Each tier is connected to one or more

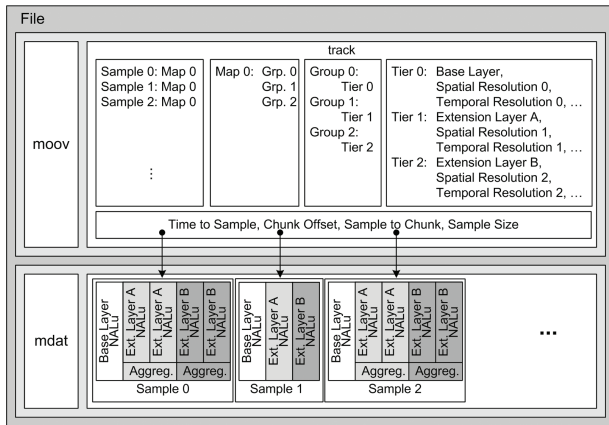


Fig. 11. Using aggregators to build regular structures.

groups. Exactly one of the groups associated with a tier contains the tier description. This group is the primary definition of this tier.

However, there is a remaining issue. Each file-format sample—a scalable coded video frame—is divided into NAL units. It is possible that the number of NAL units used for each tier varies from frame to frame, even though the frames have the same general structure. Representing this in the maps is possible, of course, but may needlessly increase the number of maps.

In order to address this, scalable video streams in the file may contain another in-stream structure, which, like the extractors discussed above, is structured like a NAL unit. This structure, called an *aggregator*, exists to aggregate other NAL units into a single logical NAL unit for the purposes of description. Fig. 11 illustrates the usage of aggregators: sample 0 and sample 2 show virtually the same structure and are described by map 0.

Using these structures, it is now possible to:

- 1) structure the file-format samples (coded video frames) into regular, repeating patterns of groups, using aggregators;
- 2) document those patterns using map sample groups, and associate each file-format sample with the appropriate map. Each map is a series of group indexes;
- 3) assign each group to a tier;
- 4) document the nature of each tier by index with a detailed description.

The tier descriptions may contain a wealth of descriptive data, some of which cannot easily be deduced from the stream itself.

Besides temporal and spatial resolution, detailed bit rate information is available, e.g., the total average and maximum bit rate of the stream including this tier or the additional average and maximum bit rate of this tier. Furthermore, it tells which SVC operating points described by DTQ (see Section II) are contained. Optionally, there are statements regarding region of interest or HRD parameters. Additionally each tier can be individually encrypted to enable layered protection.

Tiers are identified by an increasing tier ID. A larger value of tier ID indicates a higher tier. A value 0 indicates the lowest tier. Decoding and presentation of a tier is independent of any higher tiers but may be dependent on lower tiers. If tiers temporally

subset the track, retiming information is provided to enable a constant frame rate when accessing the temporal subset.

In a special use, tiers are assigned to indicate a number of operating points, which might be of interest during further bit stream adaptations. In this case, tier ID might be reflected by the value of *priority_id*.

D. Metadata

The two techniques above depend on the regular nature of the stream. However, there are some aspects of the stream that may be irregular in nature. For example, it can be useful to know the answer to some simple or more complex questions without scanning the visual data.

- 1) How many NAL units are contained in this file format sample?
- 2) How large are they?
- 3) What are their types?
- 4) How are they aggregated?
- 5) Which NAL unit is predicted from which other NAL units?
- 6) Which region of the image does the current NAL unit cover?

These, and many other questions, can be answered in time-parallel metadata in an SVC metadata track.

An SVC metadata track is structured as a sequence of file format samples, just like a video track. However, each metadata sample is structured as a metadata *statement*. There are various kinds of statement, corresponding to the various questions that might be asked about the corresponding file-format sample or its constituent NAL units.

Statements fall into two broad classes: there are predefined statement types in the SVC File Format specification; and there is explicit provision for third-party or extension statements.

Each statement in the file is identified with an index; for the predefined statements, these indexes are defined in the specification. For extension statements, the sample-entry in the track setup information contains a mapping table from index to a URL. The URL may be (and usually is) dereferenceable, providing documentation, or even a schema, to define that statement type.

An example of the use of this might involve the MPEG-21 bit stream description language [9]. In this case, the URL might address an anchor point in a BSDL XML description.

Some of the predefined statement types concern the structuring of the metadata. Three important ones are as follows.

- 1) Empty statement: This is used when no statement needs to be made about the matching video coding data.
- 2) Group of statements: This is used when more than one statement needs to be made about the matching video coding data; it contains that set of statements;
- 3) Sequence of statements: This is used where the matching video data can be de-composed into a sequence, e.g., it is a video coding sample, an aggregator, or an extractor, all of which are a sequence of NAL units. This statement contains a sequence of statements, in one-to-one correspondence with the sequence in the matching video coding data.

The entire metadata sample is therefore defined as an implicit group of statements about the entire temporally-aligned video coding file format sample.

Other predefined statement types include:

- 1) a copy of the NAL unit header of the matching NAL unit (which gives its type, size, and so on);
- 2) a statement about the contents of an aggregation (how many NAL units it contains, etc.).

The following shows an example (also illustrated in Fig. 12). First, the media sample to be described is shown (in pseudo syntax)

```
SEI NALu
Base-layer Slice NALu 1
aggregator NALu 2 containing {
    enhancement NALu 2.1,
    enhancement NALu 2.2
}
another enhancement NALu 3
```

An example matching metadata sample follows:

```
some statement about the whole sample
sequenceOfStatements {
    empty statement about SEI NALu
    groupOfStatements: {
        NALu header 1 statement;
        some other statement about NALu 1
    }
    groupOfStatements {
        aggregator statement
        sequenceOfStatements {
            NALu header 2.1 statement;
            groupOfStatements: {
                NALu header 2.2 statement
                another statement about NALu 2.2
            }
        }
    }
    some statement about NALu 3
}
```

There is an option to transmit entire metadata samples or parts of it (e.g., a group of statements or just a single statement) within an SEI message (e.g., a user data unregistered SEI message). This enables transport of the metadata together with the related video data.

E. AVC Compatibility

In the SVC File Format, a provision for storing in an AVC compatible manner exists, such that the H.264/AVC compatible base layer can be used by any existing AVC File Format compliant reader.

AVC compatibility can be divided into two major areas.

- 1) File format compatibility: If a track is marked both AVC compatible (“avc1” sample entry) and SVC compatible

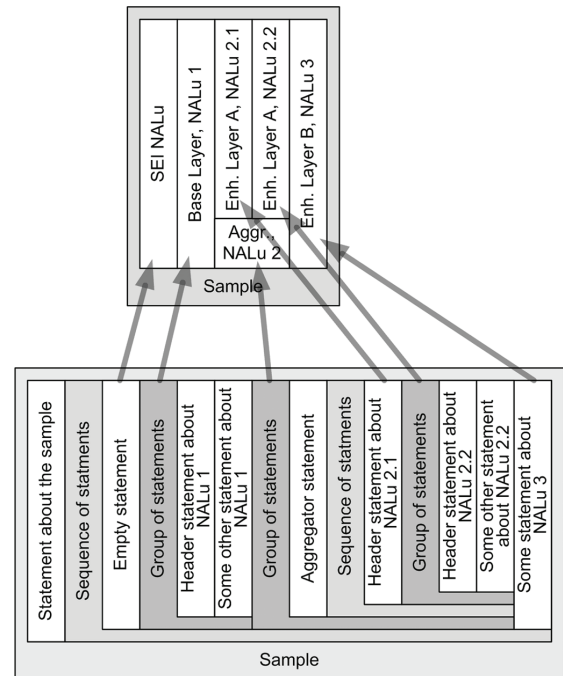


Fig. 12. SVC sample and corresponding metadata sample.

(“svc1” sample entry), all file format structures must be valid for the entire track regardless of whether it is read by a legacy AVC reader or by an SVC reader.

- 2) Video coding layer compatibility: If an SVC track containing an H.264/AVC base layer is also marked AVC compatible, the video data passed to the decoder must fulfill all requirements (e.g., buffer sizes) indicated by the H.264/AVC base layer.

An SVC track may use one of three different sample-entry names.

- 1) “avc1”: used for plain AVC tracks or for SVC tracks with an H.264/AVC base layer but not using data extraction to access the H.264/AVC base layer data. Additionally an “avc1” track must contain an H.264/AVC compliant bit stream. This label is the sample entry name defined in the AVC File Format specification, and is therefore fully backward-compatible.
- 2) “avc2”: used for plain AVC tracks or for SVC tracks with an H.264/AVC base layer but using data extraction to access the H.264/AVC base layer data. An “avc2” track must contain an H.264/AVC compliant bit stream.
- 3) “svc1”: used for SVC tracks which are not or should not be considered as AVC compatible.

Aggregators and extractors can only be used in “avc1” tracks if access to the H.264/AVC base data is not affected by them; in particular, if the SVC data is wrapped in aggregators, this enables easy skipping by an AVC reader.

To ensure AVC compatibility it is recommended to store the H.264/AVC base layer in a separate AVC base track. SVC enhancement layer data should be stored in one or more enhancement tracks, which reference the AVC base track (see Section V-B).

F. Summary

In general, the SVC File Format defines techniques to describe operating points and the resulting grouping of bit stream elements. Furthermore, bit stream structures and the dependencies which exist between bit stream elements are described. Three different kinds of scalability assistance are defined to enable efficient subsetting and extraction.

- 1) Precomputed scalability assistance: a track describes a subset of the total bit stream. The track represents one operating point. This might be achieved by copying the media data or using extraction instructions. A file reader may choose from one of the offered tracks and reads the entire track. These tracks may use *extractors*.
- 2) Scalability assistance through tiers (mainly assistance for layered scalability): A track describes the entire bit stream or a subset of the total bit stream. Additionally, a set of operating points (tiers) is given in this track. After choosing the track, a file reader may choose one of the offered operating points. While accessing the file, data extraction operations are performed (using grouping information, see Section IV-C). Additionally the extraction path defined by the tiers in this track might assist by setting the value of *priority_id* in the NAL unit headers to enable further adaptations at the given operating points. The sample groups provide a summary (grouping), of the layers and their possible extraction.
- 3) Scalability assistance with parallel metadata: The time-parallel metadata provides frame-by-frame assistance and optionally NAL unit by NAL unit assistance for understanding and extracting data from the scalable stream, e.g., when using the full scalability mode.

In all scalability assistance modes, tracks may share media data as described before.

V. EXAMPLES OF USE

A. Simple Extractor Tracks

Consider an application which needs to be able to deliver at three operating points, such as QCIF at 15 fps (frames per second), CIF at 15 fps, and CIF at 30 fps. In this example, we encode all the data in one track. (This single track is then the “complete set” in this case.) This track would therefore (if decoded completely) yield the CIF and 30 fps version of the content.

We can now define two more extractor tracks, operating at CIF with 15 fps and QCIF with 15 fps. These two tracks share data with the first (by referencing it from extractors); they have, of course, only half as many access units to decode; the access units that would have yielded 30 fps are omitted entirely.

There is probably also an associated audio track. All the video media-data is referenced from the first video track, which is the only one that is marked as needed to be kept, if the entire bit stream is to be retained. The normal file layout would interleave all these four tracks together (Fig. 13). This means that typically the file reader reads all the data (e.g., from disc) and then subsets it. (It is more efficient to read large sections of a file at once.) The subsetting, of course, is done in two steps: selecting the data actually referenced by the track in question, and then replacing extractors by the data that they reference.

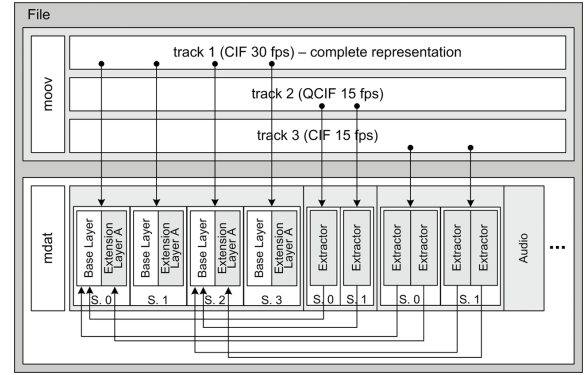


Fig. 13. Extractor usage example.

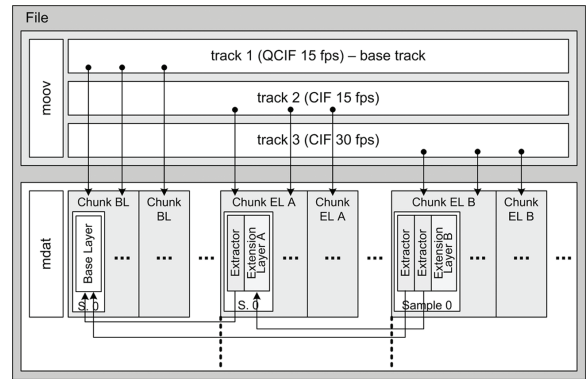


Fig. 14. Chunk layout to support erosion storage.

B. Base Track

A very different application, using the same operating points as in the previous example, can come up when considering erosion storage, as discussed above. In this example, the initial recording is at CIF and 30 fps, for example, but later it is reduced to 15 fps, and later again to QCIF resolution.

In order to achieve this, we organize the file differently. Rather than putting all the media data in one track and subsetting it, we instead place the base quality (QCIF at 15 fps) in one track (i.e., the base track), and then we add two more tracks that use extractors to access that base data, and then contain the enhancement in-line. In this case, all three tracks contribute to the complete scalable bit stream (the “complete set”).

We then interleave the base track with the audio data, at the earliest part of the file. The media data for CIF at 15 fps then follows, all together and following all the base data; this, as said before, contains extractors referring to the needed base data, and also the enhancement video data. Finally, the media data for CIF at 30 fps is similarly placed last in the file (Fig. 14).

This file has the same operating points as the one in the first example. However, storage space is now easily reclaimed. The track structure (the track “box”) can be marked as “free” space simply by changing its signature, truncating the file and adjusting the length of the media-data box to eliminate (and free) the stored bit stream for the CIF 30 fps layer. Then again, later, the CIF 15 fps material can be truncated from the file, and the matching track can be removed by changing its type to “free.” Through this type change, a rewriting of the file (or at least

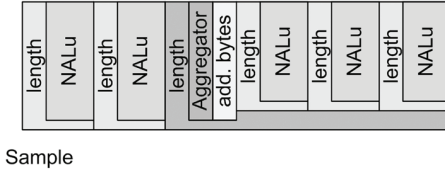


Fig. 15. Including NAL units in an aggregator.

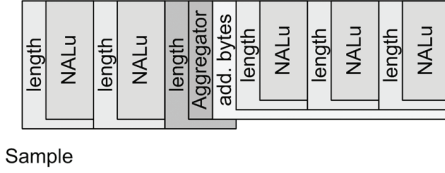


Fig. 16. Referencing NAL units by an aggregator.

the “moov” box) e.g., with updated length information can be avoided in comparison to a deletion of the track.

C. Aggregator Usage

Aggregators may be used to group NAL units belonging to the same sample. Aggregators are special NAL units, which use a NAL unit type from the reserved range. A file reader interprets an aggregator as one NAL unit. This can be used to build regular structures (as described above, see Fig. 11) or to virtually hide SVC contents from an AVC file reader. While accessing a track by an SVC file reader, the aggregator is unpacked and removed.

Aggregators may include NAL units or reference a continuous number of bytes. An including aggregator can be seen as a single large NAL unit (Fig. 15). An AVC file reader ignores the aggregator and skips it as a whole.

A referencing aggregator includes NAL units by referencing a number of additional bytes following the aggregator. An SVC file reader treats the referenced NAL units as if they were included. An AVC file reader ignores the aggregator but accesses the referenced NAL units (Fig. 16). This can help getting a regular structure for H.264/AVC NAL units.

Mixing including and referencing NAL units in a single track is also possible.

D. Reading Map and Group Information

This example shows how to interpret the grouping information. In the example, the bit stream has the following structure (see Fig. 17; dependencies are illustrated by arrows):

- 1) an H.264/AVC base layer with QCIF at 15 fps;
- 2) a spatial enhancement layer to CIF, also providing 30 fps;
- 3) a second spatial enhancement layer to 4CIF, including an MGS layer.

For this stream, four tiers are defined.

- Tier T0: H.264/AVC base layer (QCIF at 15 fps).
- Tier T1: spatial enhancement of T0 to CIF.
- Tier T2: temporal enhancement of T1 to 30 fps.
- Tier T3: spatial enhancement of T2 to 4CIF (including the MGS enhancement).

Fig. 18 shows the NAL unit structure of five samples of this bit stream, also illustrating the tier assignment.

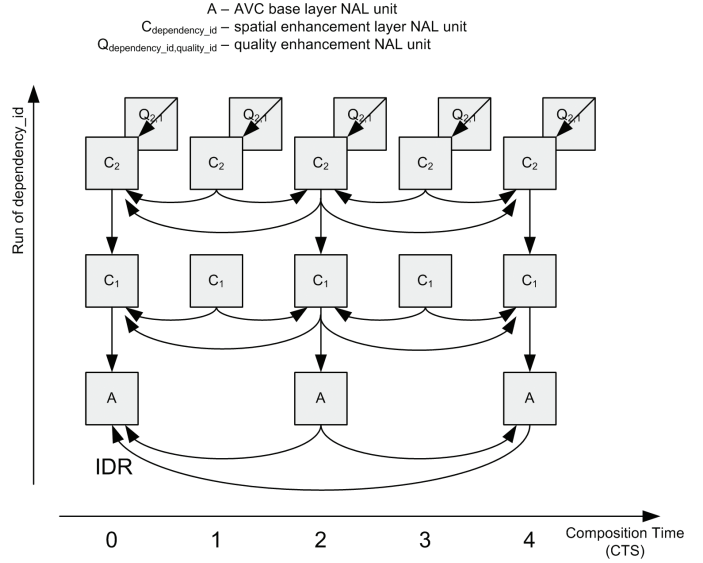


Fig. 17. Example bit stream structure.

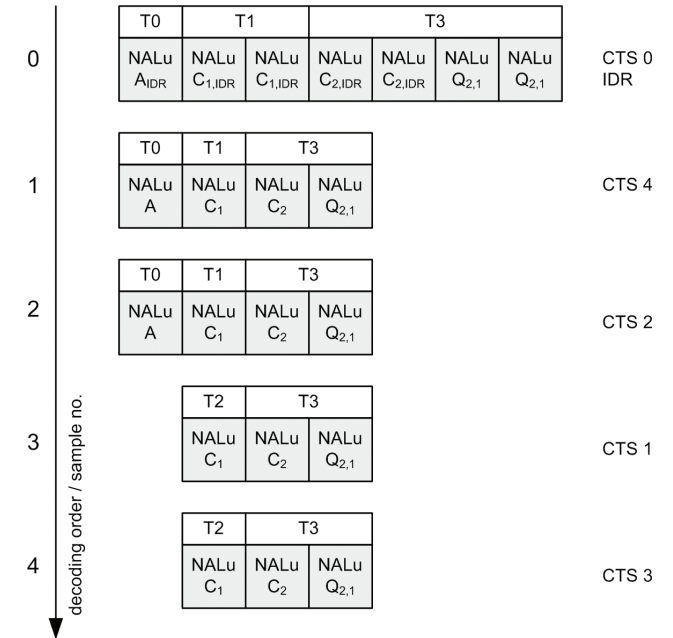


Fig. 18. Sample structure of example bit stream.

Tiers are assigned to a group G, but more than one group might be assigned to one tier to reflect special properties. In the example, one of these properties is IDR picture. IDR (instantaneous decoding refresh) pictures allow for random access into the stream, since all buffers (e.g., previously decoded pictures) are cleared. The “primary definition” contains the tier description. The following illustrates the group assignment of the example in Fig. 17:

- Group G0 → Tier T0, primary definition;
- Group G1 → Tier T0, tier IDR;
- Group G2 → Tier T1, primary definition;
- Group G3 → Tier T1, tier IDR;
- Group G4 → Tier T2, primary definition;

- Group G5 $\rightarrow$ Tier T3, primary definition;
- Group G6 $\rightarrow$ Tier T3, tier IDR.

The SVC file format uses maps to describe the sequence of the scalable properties of the NAL units in a sample. All samples with identical sequences (identical maps) are grouped together. Each NAL unit belongs to exactly one group. There are as many maps as there are different sequences of groups G in the entire track. Maps are defined by a “scalable NALu map entry” in a visual sample group entry of type “scnm.” Maps do not have an ID. Its ID is inferred from the value of entry count in the sample group description. One map (identified by such value) is then assigned to each sample.

In the example the following possible group sequences exist (compare Fig. 18):

- Map M0 $\rightarrow$ G1, G3, G3, G6, G6, G5, G5 (as in sample 0);
- Map M1 $\rightarrow$ G0, G2, G5, G5 (as in samples 1 and 2);
- Map M2 $\rightarrow$ G4, G5, G5 (as in samples 3 and 4).

Finally, each sample is assigned to a map:

- Sample 0 $\rightarrow$ M0;
- Sample 1 $\rightarrow$ M1;
- Sample 2 $\rightarrow$ M1;
- Sample 3 $\rightarrow$ M2;
- Sample 4 $\rightarrow$ M2.

In the example, if a picture of tier 1 is to be extracted, the groups G2 and G3 are needed. Since tier 1 depends on tier 0, groups G0 and G1 are needed, too. The file reader needs to access the sample at a given position (time) if it contains data of groups G0, ..., G3, which are contained in M0 and M1 (Sample 0, ..., Sample 2). After reading the sample, bit stream thinning is performed by counting NAL units. Sample 0 is of map 0, which has a sequence G1, G3, G3, G6, G6, G5, G5. Since only groups G0, ..., G3 are needed, the first three NAL units are copied to the output buffer.

If, in another example, only IDR pictures of tier 1 are desired, the file reader needs to access group G1 and G3 only.

E. Multiple Extraction Paths

In this example, an application needs to send a bit stream over different networks. This includes possible further adaptation operations on the way to the receiver. The extraction path varies on the different routes and adaptations are to be made on basis of *priority_id*. Like in the example above, we consider CIF resolution at 30 fps. These are the desired extraction paths:

- CIF@30 $\rightarrow$ QCIF@30 $\rightarrow$ QCIF@15 $\rightarrow$ QCIF@7.5;
- CIF@30 $\rightarrow$ CIF@15 $\rightarrow$ QCIF@15 $\rightarrow$ QCIF@7.5.

The two extractions paths need to be reflected by the value of *priority_id*, so its value needs to be changed depending on the path. Furthermore, *priority_id* can be used by the application, which means that we don't know, which extraction path is represented by this value in the elementary stream. Therefore, an “over-ride P statement” exists to be used in the parallel metadata. This statement exists for each NAL unit in every sample as described in Section IV-D. The application can rely on the desired extraction path, if the value *priority_id* is replaced by the

“over-ride P” statement value when putting the NAL unit into the output buffer.

VI. CONCLUSION

The SVC File Format uses the flexible features of the ISO Base Media File Format, the coding features of the SVC standard and its compatibility with H.264/AVC and file format structures defined for the SVC File Format in order to achieve a highly flexible, powerful file format. There is provision for a wide variety of use cases. At the simple end, these include AVC compatibility, and rapid “cook-book” extraction of desired subsets of the stream. More flexible techniques might use the descriptive summary information, which divides the bit stream into scalable “tiers” and identifies, to which tier each part of the bit stream belongs. Further extraction assistance is offered by time-parallel metadata. Using these techniques and the data-organization options offered by the base file format, applications can optimize their computation and input/output to achieve rapid, flexible, and scalable operation.

REFERENCES

- [1] *Information Technology—Coding of Audio-Visual Objects—Part 10: Advanced Video Coding*, ISO/IEC 14496-10:2003.
- [2] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information—Part 2: Video*, ISO/IEC 13818-2:1993.
- [3] *Information Technology—Coding of Audio-Visual Objects—Part 10: Advanced Video Coding; Amendment 3 Scalable Video Coding*, ISO/IEC 14496-10:2005.
- [4] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the scalable video coding extension of the H.264/AVC standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, Sep. 2007.
- [5] M. Wien, H. Schwarz, and T. Oelbaum, “Performance analysis of SVC,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1194–1203, Sep. 2007.
- [6] I. Amonou, N. Cammas, S. Kervadec, and S. Pateux, “Optimized rate-distortion extraction with quality layers in the H.264/SVC scalable video compression standard,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1186–1193, Sep. 2007.
- [7] *Information Technology—Coding of Audio-Visual Objects—Part 12: ISO Base Media File Format (Technically Identical to ISO/IEC 15444-12)*, ISO/IEC 14496-12:2005.
- [8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” Jul. 2003, RFC 3550, STD 0064.
- [9] *Information Technology—Multimedia Framework (MPEG-21)—Part 7: Digital Item Adaptation*, ISO/IEC 21000-7:2004.
- [10] *Information Technology—Coding of Audio-Visual Objects—Part 15: Advanced Video Coding (AVC) File Format*, ISO/IEC 14496-15:2005.
- [11] A. Hutter, P. Amon, G. Panis, E. Delfosse, M. Ransburg, and H. Hellwagner, “Automatic adaptation of streaming multimedia content on a dynamic and distributed environment,” in *Proc. ICIP*, Genova, Italy, Sep. 2005, pp. 716–719.
- [12] T. Wiegand, G. J. Sullivan, J. Reichel, and H. Schwarz, *Joint Draft 10 of SVC Amendment*, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6, Doc. JVT-W201, Apr. 2007.
- [13] S. Wenger, M. M. Hannuksela, M. Westerlund, and D. Singer, “RTP payload format for H.264 video,” Feb. 2005, RFC 3984.
- [14] S. Wenger, Y.-K. Wang, and T. Schierl, “RTP payload format for SVC video,” Mar. 2007, IETF internet draft draft-ietf-avt-rtp-svc-01.txt.
- [15] S. Wenger, Y.-K. Wang, and T. Schierl, “Transport and signaling of SVC in IP networks,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 9, pp. 1164–1173, Sep. 2007.
- [16] *Information Technology—Coding of Audio-Visual Objects—Part 12: MP4 File Format*, ISO/IEC 14496-14:2003.



Peter Amon received the Dipl.-Ing. (M.Sc.) degree in electrical engineering from the University of Erlangen-Nuremberg, Germany, in 2001, where he specialized in communications and signal processing.

In 2001, he joined Siemens Corporate Technology, Munich, Germany, where he is currently working as a Research Scientist in the Networks and Multimedia Communications Department. In this position, he is and has been responsible for several research projects. His research field encompasses video

coding, video transmission, error resilience, and joint source channel coding. In that area, he has authored or co-authored several conference and journal papers. He is also actively contributing to and participating at the standardization bodies ITU-T and ISO/IEC MPEG, where he is currently working on scalable video coding and the respective storage format.



Thomas Rathgen received the diploma in electrical engineering from the Ilmenau Technical University, Ilmenau, Germany, focusing on hardware synthesis for image processing.

Currently, he is a member of the Video and Image Processing Group at Ilmenau Technical University, Faculty of Electrical Engineering, where he participates on different national and international research projects related to embedded devices and media technologies. He has been an Editor for the SVC extension of the AVC file format, among others.



David Singer received the B.S. and Ph.D. degrees from the University of Cambridge, Cambridge, U.K., focusing on multimedia systems.

As QuickTime EcoSystem Manager at Apple, Cupertino, CA, he is a member of the QuickTime engineering group, where he performs industry relations and standards work for the QuickTime team. He joined Apple in 1988 and has since held a number of positions in research and product development for the company, related to time-based networking and media technologies. He has been editor for the

MPEG-4 (ISO) file format family of specifications, among others.