Low Complexity Cloud-video-Mixing Using HEVC

Y. Sanchez[1,2], R. Globisch[1], T. Schierl[1], T. Wiegand[2]

Multimedia Communications - Image Processing Dept.[1]
Fraunhofer Heinrich Hertz Institute (HHI)
Berlin, Germany

Image Communication Dept.[2]
Technische Universität Berlin (TUB)
Berlin, Germany

***Abstract*— Real-time video applications, such as video conferencing and video surveillance systems, typically involve the simultaneous transport of multiple video sources to interested parties that consume the content. It may be desirable to mix these multiple source videos into a single video stream at intermediary nodes in the network. This has the advantage of reduced application and transport complexity on the client device and also makes it possible for devices with a single hardware decoder to consume the content. A typical approach is to apply transcoding operations to the original videos, i.e. the videos are decoded, merged and encoded into a single video stream. This paper proposes an alternative solution to video transcoding, which uses the new video coding standard HEVC and has a much lower processing complexity. We consider how our approach can be realized in real-world applications such as a cloud video mixer. Such systems typically require some degree of dynamics and personalization and we provide some insight into how transport signaling complexities can be addressed.**

***Index Terms*— HEVC, video mixing, video conferencing, surveillance, cloud.**

I. INTRODUCTION

The highly available resources and cost-efficient nature of what is nowadays known as the cloud has enabled a great number of new services that were not previously viable due to the high investment costs involved in setting up the infrastructure required to support such resource intensive services. The cloud has the necessary resources per se. Thus, services requiring high computational processing, e.g. video processing services, can easily be developed using a cloud-centric approach.

Video services, such as video surveillance or multi-party video conferencing, entail presentation of multiple videos simultaneously. Applications like multi-party video conferencing or multi-camera surveillance can be implemented in software by running multiple decoders in parallel where one decoder is required per video to be presented. However, in order to target a higher number of devices and set up a service compatible with devices utilizing hardware decoders, the processing of multiple video streams in the network is necessary to generate a single video stream by merging multiple videos into a single video. In many cases, devices only use a single hardware-accelerated decoder to achieve real-time decoding capabilities and therefore sending only a single coded stream is necessary. Mobile devices including many smart phones and tablets fall into this category.

In order to accomplish such an operation, pixel-domain video stitching is typically applied, where the different video streams are transcoded into a single bitstream. Transcoding can be implemented using a cascaded decoder and encoder, which entails decoding the incoming bitstreams, mixing/stitching them in the pixel-domain and encoding them together into a single bitstream. This method is also referred to as uncompressed-domain stitching. Besides having a high computational complexity it also results in quality degradation of the original source videos caused by the double encoding [1]. Even though there are techniques that aim to reduce the transcoding complexity [2], it is still necessary to carry out resource consuming operations that require partial decoding of the videos, e.g. entropy decoding.

Although there may be an abundance of resources in the cloud to perform transcoding for merging multiple videos into a single video, it is desirable to use techniques with a low complexity, making it feasible for real time operation in a scalable manner and saving costs. Additionally, each consumer of the multiple video streams may want to see a different subset or representation of the videos. For example, in a video conferencing application each participant may want to observe different participants. Such personalization makes it even more important that the mixing process is of a low complexity.

It is expected that in a near future, many of the video streams will be encoded with the new HEVC standard [3]. Therefore, this paper proposes a solution for streams encoded with HEVC to be mixed in the cloud. In [4], a similar proposal has been presented, where the technique is referred to "compressed-domain stitching". In [4] changes to the decoder are introduced so that the stream mixed with a low complexity mixing operation, as described in this paper, can be correctly decoded. However, the solution proposed in [4] leads to standard-compliant decoders not being able to correctly decode the mixed streams. Instead, we propose to impose a set of constraints at the encoder side so that a standard HEVC decoder can correctly process the merged video streams.

The remainder of the paper is organized as follows. Section II gives an overview of the system that we name "cloud-mixer". In section III, an overview of the HEVC tools used for video mixing and related work is given. Besides, the differences of the solution in this paper with respect to the solution in [4] are given, explaining the set of constraints proposed for encoders so that the mixed video is correctly

decoded by standard HEVC decoders. Section IV describes the system control protocol necessary for controlling the senders providing the video streams to be merged. In section V, an evaluation of the performance of the techniques described in the paper is made. Finally, section VI summarizes the paper and presents conclusions.

II. System Overview

The functionality carried out in the cloud for the considered application is similar to that what RTP-Mixers [5] or RTCP-Terminating MCU [6] do currently. However, in the case of both Mixers and MCUs, the original streams are either kept as independent streams, identified by different CRSC fields [5], or the various content is mixed together (in the uncompressed domain) and re-encoded at intermediate nodes in the network.

Figure 1: System architecture for video mixing, e.g. traffic video surveillance

Figure 1 shows the system architecture of the "cloud-mixer" for a multi-video application. As can be seen, multiple video streams are sent by different senders and are merged in the cloud into a single encoded video bitstream.

In order to allow for low complexity video mixing, not requiring transcoding of the incoming video bitstreams, incoming bitstreams have to fulfill a set of constraints, as described in section III, and senders have to be synchronized using control messages as described in section IV.

III. HEVC And Video Mixing

High Efficiency video coding (HEVC) [3], which reduces the bitrate by 50% in comparison to its predecessor H.264/AVC while maintaining the same quality, includes new tools that enable low complexity video mixing. In particular, HEVC includes the concept of tiles, which result from partitioning a picture into rectangular regions each referred to as a tile. Figure 2, shows an example of partitioning a picture into several tiles.

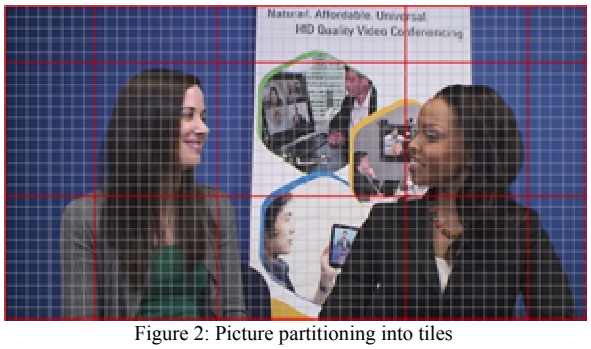

Figure 2: Picture partitioning into tiles

Tiles comprise an integer number of coding tree units (CTUs - small blocks in the figure) and are depicted in the figure as red blocks. Tiles are independently decodable parts of the same picture and are generated by splitting the original picture into a given number of rows and columns of an arbitrary width and height.

Similar to the solution already presented in [4], this paper uses tiles in order to merge different videos into a single video bitstream. The idea is to rewrite the parameter sets to adapt to the new characteristics of the video as well as rewrite the slice segment header of the incoming NAL units in such a way that the outcoming NAL units belong to the same bitstream but to different tiles corresponding to different regions of the picture. Figure 1 depicts an example with four input streams. Thus, each incoming bitstream is converted into a different tile in the output stream.

Although we do not show a comparison of complexity between the solution proposed here and a solution based on transcoding, it is obvious that the solution proposed here has a much lower complexity than trancoding operations. Note that our solution entails only parsing and updating a relatively small number of syntax elements as described in Section III.A, while transcoding additionally entails carrying out some of the most demanding operations in video decoding/encoding, such as entropy decoding/encoding, reconstruction of the decoded image and syntax element (motion vector) prediction update.

In order to merge several videos into a single output bitstream by rewriting NAL unit slice segment headers, the input video streams have to fulfill the constraints described in section III.B.

A. Related Work

In [4], the compressed-domain video mixing, i.e. rewriting of slice segment header of NAL unit and parameter sets, is described. It entails:

- Video Parameter Set (VPS) rewriting: e.g. indicated level.
- Sequence Parameter Set (SPS) rewriting: e.g. picture size set to the resulting size from video mixing.
- Picture Parameter Set (PPS) rewriting: dimension of tiles set based on size of input streams and merging pattern.
- Slice segment header rewriting: segment address update.

In addition to the segment address rewriting in slice segment headers described in [4], it may be also necessary to change the slice_qp_delta syntax element if the initial QPs signaled in the different PPSs (init_qp_minus26) of the different streams differ. In such a case, one of them needs to be selected as a reference and the slice_qp_delta in the slice segment header of the affected NAL units needs to be updated to signal the same resulting QP as the original one.

Rewriting of the mentioned NAL units by itself is not enough for ensuring correct merging of the videos. The described technique may cause drift at the decoder side if certain aspects are not addressed properly. The main issues discussed in [4] were:

- Inter-prediction mismatch due to vectors pointing to previously non-existing areas for which so-called constant border extension was performed.

• Inter-prediction mismatch due to sub-pixel interpolation being performed with newly added pixel values close to the "region" borders where two pictures of different videos are merged/stitched.

• Adaptive Loop Filtering (ALF) performed over the whole picture without possibility of avoiding ALF filtering across tile borders.

The issue with ALF is no longer an issue since ALF has not been included in the final version of the standard.

Furthermore, the Advanced Motion Vector Prediction (AVMP) in HEVC has evolved and Temporal Motion Vector Prediction (TMVP) [7] is now derived from the prediction unit (PU) located in the right-bottom of the collocated PU of the reference frame instead of from the collocated PU (unless the right-bottom collocated PU is not available for TVMP derivation). PUs located at the right border of the picture are examples of such a scenario. Since there are no PUs beyond picture border, in this case, the collocated PU is used for TMVP instead. Therefore, at tile borders, where videos are merged together, an additional mismatch not considered in [4] may happen. When TMVP from a PU from another tile/video is used, a wrong syntax element is predicted, potentially leading to severe drift, which would affect the proposed solution. In this case, there might be a PU from another tile (due to video mixing operation) at the right-bottom of the collocated PU which would be used for TMVP, where the collocated PU was used in the original video since there was no PU at the right bottom.

Contrary to [4], which proposes modifying the decoder behavior so that the decoding process treats tile boundaries as picture boundaries, by e.g. performing border extension when motion vectors point beyond tile borders, we propose a set of constraints on the encoders so that standard-conformant HEVC decoders can be used. These constraints are discussed in the following subsection.

B. Encoding constraints

As mentioned previously, tiles are independently decodable portions of the picture. Thus, it is not necessary to apply any constraints to the merged intra frames, since no dependencies exist between the different tiles. However, when each of the incoming pictures is converted into a tile of the outgoing bitstream, special care has to be taken that each tile (picture region) output at the decoder looks exactly the same as the original, since this picture may be used as a reference picture during decoding. This implies that in-loop filtering across tiles of the output stream needs to be disabled and consequently the input streams must have the same constraints if tiles are used.

Additionally, slices may be used to subdivide the picture into different portions if the desired flexibility is not achievable with tiles (see Figure 3). In the figure, input streams of different sizes are considered, leading to two tiles: one containing the stream with a bigger picture size and the other containing two streams with smaller picture sizes. Here the tile has been subdivided into multiple regions with slices, each containing one video input stream (marked in the picture with the black dashed line). Similarly to the reasoning for cross-tile filtering, in-loop filtering across slices also needs to be disabled. Therefore, if the original video streams contain multiple slices per picture, these have to be generated taking this constraint into account.

Figure 3: Video mixing using tiles and slices

Parameter set values such as i.e. chroma_idc, bit_depth_luma, etc. must be consistent across input video streams. Further, no conformance cropping window can be present in the original streams, since it is not possible to signal it for each of the videos once they have been merged.

Additionally, the encoding process should be constrained to avoid the aforementioned mismatch in the inter prediction. The issue with the sub-pixel interpolation comes from the fact that HEVC supports motion vectors with units of one quarter of the distance of luma samples [3]. An 8-tap filter and 7-tap filter are used for half-sample and quarter-sample interpolation respectively. For sample units close to the picture borders, sample units that are outside the original picture, i.e. non-existing samples, are needed for interpolation. The value of the last sample of the picture is copied for the non-existent samples (by performing the so-called border extension). However, when videos are merged, samples that did not exist before for a given video may exist in the merged video (these samples correspond to another video in the resulting merged video), leading to different interpolated samples in the encoder and decoder. Therefore, encoders have to constrain the sub-sample vector selection for prediction units in which motion vectors point to the edge of the picture, i.e. not selecting sub-sample motion vectors that point to samples that lie within the sample position (x,y) with x (horizontal component of its luma position) or y (vertical component of its luma position) within the interval [0, 2], x within the interval [PicWidth-4, PicWidth-1] or y within the interval [PicHeight-4, PicHeight-1], where PicWidth refers to the width of the picture in number of luma samples and PicHeight refers to the height of the picture in luma samples.

As aforementioned, mismatches may also happen due to motion vectors pointing to previously non-existing areas. Note that the size extension of the picture due to the video mixing operation leads to two issues: clipping of the motion vectors (which depends on the size of the picture) is different in the merged video stream and the original video stream and border extension of original pictures converted into tiles may not happen unless also placed in the border of the merged video. Therefore, special care has to be taken into constraining the possible values of motion vectors, i.e. the selected motion vectors shall not point to luma samples (x,y), for which x (horizontal component of its luma position) does not lie within the interval [0, PicWidth-1] and y (vertical component) does not lie within the interval [0, PicHeight-1].

Finally, as mentioned in subsection III.A, AMVP and more concretely TMVP may be wrongly predicted in PUs on the right border of each of the tiles generated from the input streams. Note that for those PUs on the right border the temporal predictor may not exist in the original video streams, or the motion vector from the collocated PU is used as temporal predictor instead of the motion vector of the right-bottom collocated PU. In either case, the candidate lists for motion vector prediction at the decoder may be different for the PUs on the right border compared to the candidate lists used at the encoder, i.e. this is the case when as a result of merging a temporal predictor exists in the right bottom PU. In the first case, when the temporal predictor does not exist either in the right bottom collocated PU or in the collocated PU, the candidate list at the encoder has a zero candidate direct after the neighbour candidates, while at the decoder, there might be a temporal prediction in the right bottom collocated PU, which belongs to the new region/tile merged in the network, leading to a non-zero candidate directly following the neighbour candidates. In the second case, when the motion vector from the collocated PU is used as temporal predictor instead of the motion vector of the right-bottom collocated PU at the encoder, there might be a temporal prediction in the right bottom collocated PU at the decoder, which belongs to the new region/tile merged in the network, resulting in a different temporal candidate.

The merge index (merge_idx) in skip or merge mode, or the motion vector prediction flags (mvp_l0_flag and mvp_l1_flag), which point to a candidate in the described candidate list, may point to a zero vector that fills the motion vector candidates list for the absent temporal motion vector predictor or to a different motion vector predictor as described above. Additionally, even though the candidate may be the same, in order to correctly infer the value of the reference index refIdx for the skip or merge mode when a zero candidate is selected, it is necessary to know the exact position of the first zero candidate added (since for each new zero candidate refIdx is increased wrapping around at the highest possible value - number of reference frames - while starting with a value of 0 for the first zero candidate). A wrong derivation of refIdx would lead to the selection of the wrong reference picture, resulting in severe drift. Such a problem does not exist for the non skip/merge mode, since the refIdx is indicated as a syntax element.

Taking all this into account, the encoder must be constrained in such a way that, for PUs at the right border, the merge_idx does not point to a candidate that is not a neighbour candidate in the absence of a temporal candidate and may only point to the zero candidates if a temporal candidate exists, i.e. if the collocated PU is used for TVMP. For non skip/merge mode, mvp_l0_flag/mvp_l1_flag shall not point to a zero vector directly following a neighbour predictor or a zero vector that is the first of all candidates, which may happen in the absence of a temporal candidate.

IV. Control Protocol

A signaling protocol is required to enable dynamic multi-source video services, where assignment of the incoming videos to different regions of the outgoing video varies in time. An example would be a multi-party video conference system where active speakers are presented using a higher resolution than passive speakers (see Figure 3) and active speakers dynamically change. Also, such a control protocol would allow users to indicate their interest in bitstreams, so that they could join the bitstreams of interest (different for each user) and these are mixed in the cloud allowing personalized content, e.g. dynamic selection of a subset of cameras in traffic video surveillance. User capabilities and interests can simply be negotiated using the SDP Offer/Answer model [10]. The Real-time Transport Protocol (RTP) and associated RTP Control Protocol (RTCP) [5] are typically used for media transport in real-time media applications.

A. Region (tile) switching

As aforementioned, in some scenarios such as multi-conference services speakers are presented in a higher resolution than non-speakers and a bigger region of the merged stream is assigned dynamically depending on who is the current speaker is at each moment. This entails a dynamic resolution change of some input video streams, which requires signaling between the cloud mixer and the video source(s).

Two signaling approaches allow such functionality, where the first approach renegotiates session using a SIP re-INVITE [11], and the second approach used RTCP feedback messages.

[8] specifies Payload-Specific Feedback messages that could be extended with a Payload Type Selection message where PT=PSFB and FMT=4 (parameter values 4 -14 remain unassigned), e.g. to update the payload type being used, provided that the requested payload types (for different resolutions) have been negotiated previously e.g. using the Offer/Answer model. Thus RTCP could be used to control encoder behavior.

SIP solves this resolution change by renegotiating sessions with existing video sources. In this case, this would be accomplished using a re-INVITE.

The advantage of the RTCP-based approach is that it is faster: it takes half a round-trip time for the RTCP packet to be sent from the cloud mixer to the video source. A SIP re-INVITE would take at least one and a half round-trip times. On the other hand RTCP is typically sent over UDP which means that there is a chance that the RTCP message might be lost. In that case the encoder at the video source would never change the resolution and it would take time for the cloud mixer to notice this and resend the RTCP message. SIP is typically sent over TCP so this is not a concern.

In order to allow such a change of regions (position in the merged streams) affected video streams have to break dependencies from previous pictures sending an intra-only slice and avoiding references to previous pictures. Encoders should keep a consistent Reference Picture Set (RPS) [9] as if no change would have happened, i.e. including pictures not used for reference so that other regions which do not suffer any change are not affected. If this is not the case, the “cloud-mixer” would have to modify RPS signaling in the slice segment header so that regions that are not modified have a consistent RPS. Encoders must also avoid the usage of IDR or CRA pictures for the resolution change so that the mixer does not have to modify syntax elements to convert those NAL units to non-IDR/CRA NAL units.

B. Adding/Removing Video Source Dynamically

Further signaling requirements need to be considered to add and remove video sources dynamically, which would lead to repartitioning of the output picture and most probably to a resolution change of the merged bitstream. In the case, where the resulting resolution of the merged bistreams does not change, only a resolution change needs to be performed by the video senders. In such a case, the discussion above applies.

However, if adding or removing video sources leads to a resolution change of the merged bitstream, it may be necessary for the cloud mixer to renegotiate media sessions with both the video senders and receivers using the SIP re-INVITE mechanism. For such case, it is not possible to use the RTCP approach described before for the receivers, a SIP re-INVITE is required for negotiation, using the Offer/Answer model as explained before.

V. Analysis

We have implemented the constraints mentioned in section III.B at the encoder based on HM9.1. In order to evaluate the efficiency cost due to the constraints, we encoded three streams in the low delay mode with hierarchical P frames, which is an adequate configuration for e.g., video conferencing applications. The “FourPeople”, “Kristen and Sara” and “Johnny” test sequences were used in the following results and were cropped and downscaled to 1280x640 and 640x320 to achieve the setup illustrated in Figure 3 so that the resolutions are multiples of the Coded Treeblock Unit (CTU) resolution (64x64 in our experiments), which is a requirement for tiles.

The encodings were done for QPs from 26 to 34 resulting in the 9 operation points shown in Table I and Table II. In order to compare the rate-distortion characteristics of both the non constrained and constrained encodings, the Bjøntegaard delta measurement [12] has been used to evaluate the encoding penalty incurred by the proposed constraints. We have exemplarily shown the exact bitrate-PSNR values for the test sequence "Kristen and Sara" in Table I and Table II for a resolution of 1280x640p60 and 640x320p60 respectively, as well as the penalty in terms of Bjøntegaard delta measurement. The penalty incurred due to the constraints set at the encoder side is around 1.9% for the higher resolution video, while being higher for the lower resolution video at about 2.9%.

TABLE I. Performance comparison for "Kristen and Sara" sequence at 1280x640p60

Constrained		Non-Constrained	
Bitrate (Kbps)	PSNR (dB)	Bitrate (Kbps)	PSNR (dB)
252.47	41.13	245.49	41.14
218.48	40.49	212.96	40.51
189.40	39.85	185.06	39.85
166.88	39.21	162.14	39.21
145.19	38.56	140.99	38.57
126.59	37.90	123.25	3791
110.61	37.24	107.62	37.24
97.22	36.56	94.37	36.56
85.76	35.86	83.55	35.87
Penalty [%]			
2.87			

TABLE II. Performance comparison for "Kristen and Sara" sequence at 1280x640p60

Constrained		Non-Constrained	
Bitrate (Kbps)	PSNR (dB)	Bitrate (Kbps)	PSNR (dB)
765.13	41.39	754.40	41.40
639.35	40.96	629.57	40.97
536.82	40.51	528.93	40.52
458.07	40.05	450.96	40.06
389.06	39.57	382.46	39.58
332.65	39.07	326.42	39.08
285.57	38.55	280.48	38.56
246.80	38.03	242.70	38.03
214.79	37.49	211.34	37.49
Penalty [%]			
1.86			

Figure 4 shows the incurred penalty for the three test sequences. The bars labeled with "J" correspond to results for the "Johnny" test sequence, while the bars with "K" and "F" correspond to the test sequences "Kristen and Sara" and "Four People" respectively. Additionally, for each of the sequences, the bars labeled without the appended "s", correspond to the 1280x640p60 sequences, while the bars labeled with the "s" correspond to the smaller resolution 640x320p60. In all cases the penalty incurred due to encoder constraints are very low, which may vary from values below 1% as for the "Four People" up to values around 3% for the test sequence "Kristen and Sara" at 640x320p60.

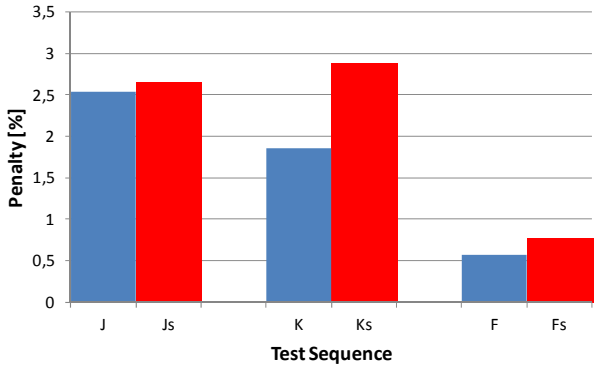

Figure 4: Penalty comparison for the three test sequences

It is also possible to observe that, although still being very moderate values, applying the constraints at the encoder for videos at lower resolution has a bigger impact than applying them at higher resolution videos. This is expected since a higher percentage of PUs of the videos suffers from the applied constraints. However, the efficiency penalty is still relatively low.

VI. Conclusion

In this paper, we have proposed a standard conformant solution for mixing multiple videos into a single video stream at low complexity using HEVC. Benefits of such a solution over demanding transcoding techniques have been discussed. Finally, encoder restrictions necessary to allow the proposed

video mixing have been discussed and comparisons between a non-constrained and constrained encoder have been shown. We consider how these constraints could be signaled in a dynamic real-world application. In conclusion, videos can be merged with low complexity by applying minor high level syntax changes at the cost of a 1-3% loss in coding efficiency. This makes our approach suitable for use in real-time cloud-based video-mixing applications.

REFERENCES

[1] K.-Y. Yoo; K.-d. Seo, "Syntax-based mixing method of H.263 coded video bitstreams," Consumer Electronics, 2005. ICCE. 2005 Digest of Technical Papers. International Conference on , vol., no., pp.403,404, 8-12 Jan. 2005

[2] I. Ahmad, X. Wei, Y. Sun, Y. Zhang, "Video Transcoding: An Overview of Various Techniques and Research Issues" IEEE Transactions on Multimedia, V. 7, N.6, Oct 2005. .

[3] G.J. Sullivan, J. Ohm, W. Han, T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Transactions on Circuits and Systems for Video Technology, V. 22, N. 12, Dec 2012.

[4] P. Amon, M. Sapre, A. Hutter, "Compressed Domain Stitching of HEVC Streams for Video Conferencing Applications", Proc. of 19th Packet Video Workshop, 2012.

[5] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550.

[6] M. Westerlund, S. Wenger, "RTP Topologies", RFC 5117.

[7] J. Lin, Y. Chen, Y. Tsai, Y. Huang, S. Lei, "Motion Vector Coding Techniques for HEVC"

[8] J. Ott, S. Wenger, N. Sato, C. Burmeister, J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF), RFC 4585.

[9] R. Sjöberg, Y. Chen, A. Fujibazashi, M.M. Hannuksela, J. Samuelson, T.K. Tan, Y.-K. Wang, S. Wenger, "Overview of HEVC High-Level Syntax and Reference Picture Management", IEE Transaction on Circuits and Systems for Video Technology, Dec. 2012.

[10] M. Handley, V. Jacobson, C. Perkins, "SDP: Session Description Protocol", RFC 4566, 2006.

[11] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, 2002.

[12] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves", ITU-T SG16/Q.6, Doc. VCEG-M033, Austin TX, April 2001.