Efficient Stream-Reassembling for Video Conferencing Applications using Tiles in HEVC

Christian Feldmann
Institut für Nachrichtentechnik
RWTH Aachen University
Aachen, Germany
feldmann@ient.rwth-aachen.de

Christopher Bulla
Institut für Nachrichtentechnik
RWTH Aachen University
Aachen, Germany
bulla@ient.rwth-aachen.de

Bastian Cellarius
RWTH Aachen University
Aachen, Germany
Bastian.Cellarius@rwth-aachen.de

***Abstract*—In a modern video conferencing application, the people participating at each client can be detected, tracked and placed in a virtual scene where all persons are of equal size and occupy a predefined rectangular space. This virtual scene can then be rendered on screen instead of a whole room with several people. As a result, a more immerse video conferencing impression is created. At a Multipoint Control Unit (MCU) it is beneficial to disassemble and reassemble the video streams so as to create a custom video stream for each client that only includes people that will be rendered by that client in order to use the available bandwidth to full capacity. In a conventional video coding approach, this video reassembling operation is only possible by decoding all incoming video streams, mixing in pixel domain and then encoding all outgoing video streams. However, this operation makes very high demands on the computational power of the MCU. In this paper, we demonstrate how in the upcoming video coding standard High Efficiency Video Coding (HEVC) the encoder can be modified to enable a reassembling operation that is HEVC compliant and works on a high syntax level in the bitstream. Hereby, no entropy en- or decoding is necessary which makes the operation very low complex.**

Keywords- HEVC; video conferencing; video mixing; coded domain; tiles; slices;

I. INTRODUCTION

Current video conferencing systems have the ability to perform high quality, real time conferences between different parties all around the world. However, the demand for high video quality and a more immersive experience is often opposed by the available bandwidth and computational power at the central Multipoint Control Unit (MCU). In order to achieve these goals, an immerse conference scenario is considered in this paper that allows for a low complex video reassembling operation at the MCU.

In classical video conferencing approaches each connected endpoint has one camera. The captured video is then encoded into two video streams: One with a high resolution (e.g. 720p) and a second one with a lower resolution which is used as a thumbnail. Both streams are transmitted to the MCU, that decides which is the most active party and forwards the high resolution video stream of this party to all the other parties. The thumbnail views are always routed to all endpoints. Of course, the active party does not receive the high resolution video of itself but the high resolution video of the last active party. Hereby, each party can see a high resolution video of the active speaker and thumbnails of the other parties.

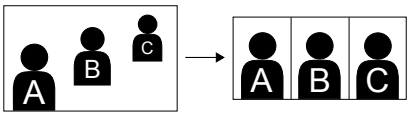

Figure 1. Each person in the scene is extracted from the captured video, scaled and placed side by side.

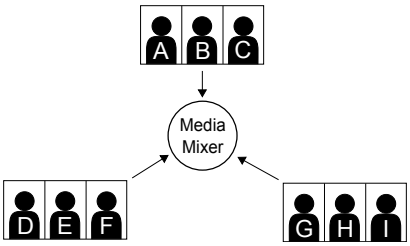

Figure 2. Three clients transmit their processed video stream to the MCU. The last most active people in the conference are E, B, C and I, descending in this order.

In this scenario, a combination of face detection, tracking and audio analysis is used in order to process the input video and extract persons from the captured video. Each person is then scaled and placed side by side (See Figure 1). This video is then only encoded in high resolution and transmitted to the MCU and from there to all clients (See Figure 2). Each client decodes the incoming video streams from the other clients and crops out only the last most active people to render them on screen. In Figure 3 an example is shown in which each client only renders the last most active speakers.

However, in this scenario the MCU transmits a lot of information to each client that the client discards after cutting out the people that it is going to render on screen. It is obvious that the required bandwidth can be significantly reduced, if the MCU supports a reassembling operation that allows outputting individual streams for each client containing only

 ISBN: 978-1-61208-265-3

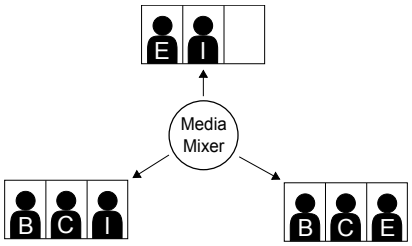

Figure 3. Each client receives and renders only the last most active and relevant people on screen. The last most active people in the conference are E, B, C and I, descending in this order.

relevant parts of the video. In the conventional approach the MCU would decode the incoming video streams, rearrange the video in the pixel domain and re-encode a new stream for each client. While this approach is simple, it has two disadvantages: Running decoders and encoders requires a lot of computing power at the MCU and has a negative impact on the overall compression performance [1] [2].

In the following Sections, we introduce a method that uses the upcoming video coding standard High Efficiency Video Coding (HEVC) [3] and the concept of Tiles [4] in order to logically split the video stream into sub-streams, with each sub-stream containing exactly one person. The video stream for each client can then be easily assembled in the MCU by only copying packets from the input video streams and altering some flags in the headers. After this operation, the output streams are still compliant to the HEVC standard and can be decoded by and device supporting HEVC.

Naturally, the proposed approach is not limited to video conferencing applications. It can be utilized in all applications where a video can be logically split into separate areas and only some of these areas need to be transmitted or the stream needs to be reassembled during transport.

In the following Section II, selected topics from HEVC will be presented that are used in the scope of the approach, before stream reassembling operations (Section III) and required encoder restrictions carried out in this approach are explained in Section III and IV. Afterwards, the arising compression loss due to the usage of Tiles and Slices and due to the encoder restrictions is measured in Section V. Finally, a conclusion about the approach is drawn in Section VI.

II. HEVC

In order to split the video stream into sub-streams, Slices and Tiles are combined in this approach to enable a high syntax level reassembling operations. This Section will give a brief overview of the HEVC tools and techniques that are used and/or modified in the proposed method.

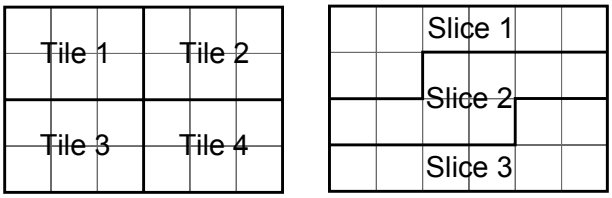

Figure 4. Subdivision of a picture with 24 CTUs into Tiles (left) and Slices (right).

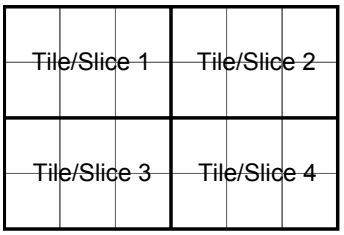

Figure 5. The picture is divided into 24 CTUs and into 4 Slices as well as 4 Tiles of equal size. Each Tile and each Slice contains 6 CTUs.

A. Slices and Tiles

Both Slices and Tiles can subdivide a frame into logically separate parts that can be decoded independently. Tiles are defined via a number of Coding Tree Units (CTUs) for the width and the height of the Tile and are therefore always rectangular, while Slices simply contain a number of CTUs that are laid out in raster scan order (See Figure 4). Slices and Tiles can be used at the same time so that a Slice can contain Tiles or a Tile can contain Slices. A special situation can occur, when Slices and Tiles contain the same number of CTUs. In this case, each Tile contains exactly one Slice and the borders of Slices and Tiles match (See Figure 5).

Since Slices and Tiles do not allow prediction across Tile/Slice boundaries or entropy coding dependencies, they are independent with respect to the encoding and decoding process [4]. Thus, Slices and Tiles can be processed in parallel which can be utilized in parallel implementations and lower the latency of the en-/decoding process [5] [3].

B. Bitstream Syntax

As in H.264/AVC, in HEVC all coded content is embedded into Network Abstraction Layer (NAL) units, which are byte aligned and have a header identifying the kind of payload. A NAL unit can contain a Slice, but also different kinds of parameter sets. Several NAL units form an Access Unit (AU), where decoding an AU results in one decoded picture and must thus contain at least all Slices of that picture. Parameter sets contain information about the whole sequence or one picture and are not entropy coded. Each bitstream must contain at least one Sequence Parameter Set (SPS) and one Picture Parameter Set (PPS), which are valid until another parameter set of the same kind is referenced (example in Figure 6) [3].

 ISBN: 978-1-61208-265-3

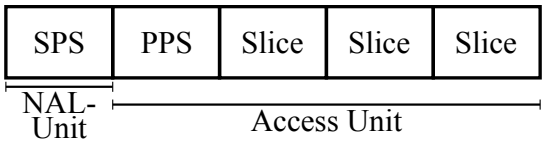

Figure 6. Bitstream containing a Sequence Parameter Set, a Picture Parameter Set and some Slices. More Access Units can follow of course.

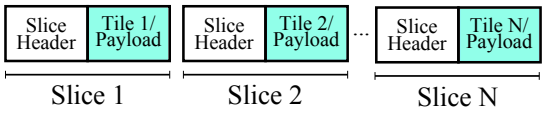

Figure 7. Structure of the Bitstream when using one Slice in each Tile (colored parts are entropy coded).

In the bitstream each Slice is composed of a header and a payload. While the header contains general information about the Slice in high level syntax, the payload is entropy coded and may contain several Tiles, if used. The end of a slice is signaled using the *end_of_slice_flag* which is the last symbol coded for each CTU and is set if the current CTU is the last CTU in the Slice. If the Slice contains Tiles, all Tiles are either separated by fixed byte sequences in the Slice payload or the byte positions in the Slice payload are given in the corresponding Slice header. When using one Tile per Slice, the payload of each Slice NAL unit contains exactly one Tile, but no information about the Tile entry points in the Slice payload is necessary (see Figure 7). Thus, the bitstream appears to contain rectangular Slices that are laid out in raster scan order in the Frame. Still, the definition of column and row boundaries of the Tiles is present in the sequence and/or picture parameter set and the Slices are identified using the *slice_address* given in the Slice header, which is the raster scan index of the first CTU in the Slice [3].

C. Inter-Prediction in HEVC

In HEVC, each frame is subdivided into Coding Tree Units (CTUs) and each CTU can be subdivided into Coding Units (CUs) of different sizes. Each CU can then be split into one, two or four Prediction Units (PUs), which are predicted using either motion compensation or intra prediction. An example for the partitioning of a CTU into CUs and PUs is shown in Figure 8 [3].

There are two different ways of signaling motion information for inter prediction to the decoder:

1) A PU can use the so called merge mode where the reference frame index and the motion information for the current PU are inferred from a neighboring PU. In order to merge a PU, a candidate list is filled and only the index of this list is encoded into the bitstream. The order in which neighboring PUs are added is standardized so the decoder can build an identical merge candidate list. A candidate is only added when the corresponding PU exists, has inter prediction related information and fulfills several more conditions (e.g. if it is located within the same CTU, Tile or Slice). There are two types of candidates (See Figure 9): The ones taken from a PU within the same frame (spatial candidates) and the ones taken from a PU in a collocated frame (temporal candidates). While the spatial candidates need to be in the same Slice/Tile as the current PU, temporal candidates do not have this restriction in general; the only restriction is, that the candidate must be located in the same CTU line [3].

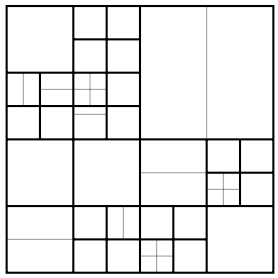

Figure 8. Example for a partitioning of a CTU in CUs (black) and PUs (grey).

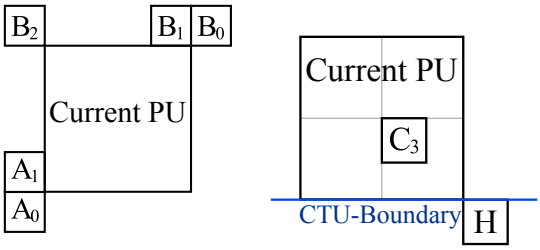

Figure 9. The spatial (left) and temporal (right) candidates that are checked for motion vector prediction as well as merge mode.

2) When the merge mode is not used for a PU the reference indices and motion information needs to be explicitly encoded into the bitstream. However, also in this case the motion information is not directly encoded but it is predicted and only the motion vector difference is encoded. In HEVC, Advanced Motion Vector Prediction (AMVP) is used which works quite similar to the merge mode. A list of possible prediction candidates is created using neighboring candidates as well as temporal candidates. Afterwards the chosen index as well as the motion vector difference is encoded into the bitstream [3].

III. Stream Reassembling

In this Section a high level reassembling operation is proposed. The usage of Tiles and the reassembling operation are similar to the proposed method in [6]. However, in [6] the definition of Slices and Tiles is changed in order

 ISBN: 978-1-61208-265-3

to enable the reassembling operation. This results in an output videostream that is not HEVC compliant and requires changes on the decoder side. The reassembling operation that is presented in this Section however, only utilizes changes on the encoder side and thus is always HEVC conforming. This allows any HEVC compliant decoder to decode the resulting bitstream.

As described in Section II, Tiles can be used to split the video stream into rectangular areas with each Tile containing one person. However, if only multiple Tiles are used, the Tiles cannot be rearranged as freely as the application requires. The problem is the *end_of_slice_flag*, which is only set for the last CTU in the last Tile. If we were to insert the last Tile with the *end_of_slice_flag* set at a different position than that of the last Tile, the set *end_of_slice_flag* would be received before all CTUs of the Slice are decoded. This results in a bitstream that is not conforming to the HEVC standard and might not be decodable. In addition the *end_of_slice_flag* is entropy coded in the bitstream and cannot be changed without entropy de-/encoding the Slice payload.

In order to circumvent this limitation we use Tiles that contain exactly one Slice as described in Section II-A. Since the concepts of Slices and Tiles coincide in this situation, we will use them synonymously hereafter. This way, each person is contained in exactly one NAL unit that contains one Tile/Slice. The people in the video stream can now be reordered by simply inserting the correct NAL units into the new bitstream while modifying the Slice headers and some parameter sets. In the Slice header the *slice_address* has to be modified to match the position of the Tile in the new video stream. Also the *first_slice_in_pic_flag* in the Slice header has to be set or reset. Furthermore, several parameters in the sequence and/or picture parameter set have to be adjusted for the new arrangement of Tiles. Concretely these are the *pic_width_in_luma_samples* and *pic_height_in_luma_samples* values as well as the *num_tile_columns_minus1*, *num_tile_rows_minus1*, *uniform_spacing_flag*, *colum_width* and *colum_height* syntax elements [3].

Overall, this reassembling operation is very low complex. Only a few values in the slice headers have to be changed and the entropy coded slice payload is simply copied to the output stream while in the conventional approach a full encoder as well as a full decoder is needed to create a similar result.

IV. Required Encoder Restrictions

The definition of Tiles and Slices in HEVC allows for independent decoding of each person. The entropy coder is reset after each Tile and prediction is generally not allowed across Tile boundaries. However, this is only true for dependencies within one frame. Some dependencies on other Tiles in past frames that are in the reference buffer can still exist. If these remaining dependencies are not removed, decoding errors as shown in Figure 10 can occur when information is referenced that changed in the reassembling operation.

Figure 10. Decoding error after switching two Tiles due to dependencies between the Tiles (right) and the original sequence (left).

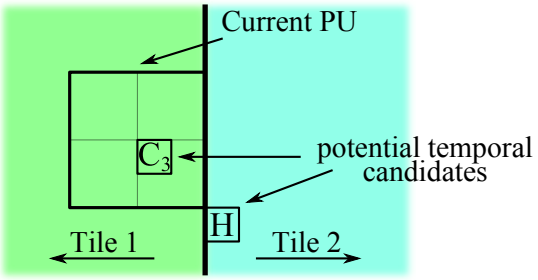

Figure 11. Temporal candidates of PU located at the edge of a Tile.

A. Modified Candidate Lists in AMVP and Merge

While creating the candidate lists for either AMVP or Merge, the possible temporal candidate H is located outside of the current PU and may be located outside of the current Tile (See Figure 11). If information from candidate H is used and H is located outside of the current Tile and that Tile was removed or replaced by the reassembling process, the information from candidate H cannot be determined by the decoder. In this case, candidate H must not be used for prediction. Also, it is possible that no Tile is present at position H while decoding. This makes all candidates after H unusable as well since the encoder cannot know if H will be available at the decoder or not and thus cannot predict how the candidate list after H is constructed at the decoder. If no Tile is present at the position of candidate H at the encoder, all candidates following the potential position of candidate H are unusable as well, since a Tile at the position of candidate H could be added during the reassembling process.

A special case occurs when the PU is located near a Tile boundary and merge mode is used to merge the motion information from a neighboring PU. In this case the encoder must not choose a PU for merging that has a motion vector which would cause the current PU to be predicted from a different Tile in the reference Frame.

 ISBN: 978-1-61208-265-3

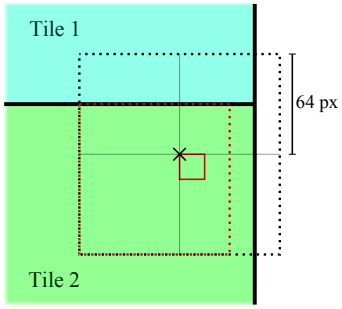

Figure 12. Motion vectors allowed by the standard (black) and the limited motion range taking into account the PU size (red).

B. Resticting Inter Prediction to Tiles

In HEVC, motion vectors are allowed to cross Tile or Slice boundaries. However, the information that is referenced by a motion vector that crosses these boundaries may have changed after reordering the Tiles. In order to only utilize information that is also available at the decoder side we limit the encoder search range near Tile boundaries. At the boundary the dimension of the PU has to be taken into account so that no part of the PU crosses the Tile boundary (See Figure 12).

V. Experimental Results

Using Tiles in a video as well as our restrictions on motion vectors and prediction candidates results in a loss in compression efficiency. In this Section we will evaluate the loss resulting from these modifications.

A. Sequences

Corresponding to the scenario described in Section I, the test sequences are composed of smaller sub-sequences that each contain one person that has been cropped and scaled from the original sequence. An example can be seen in Figure 13. Each test sequence contains three or four sub-sequences with a spatial resolution of 256×256 pixels which corresponds to 4×4 CTUs.

B. Experiments

The proposed encoder modifications were implemented into the HEVC reference software HM version 6.0 [7]. The reassembling operation of the bitstream file was implemented in Python and the rearranged bitstream was decoded using the reference decoder to test HEVC conformance. For the test set, three different configurations were tested:

1) The whole sequence without Tiles
2) The sequence using one Tile for each sub-sequence
3) The sequence using one Tile for each sub-sequence and using the encoder modifications as described in Section IV

The simulations were performed using the low delay main configuration from the common test conditions [8], which

Figure 13. The test sequence Vidyo2 after cropping and scaling.

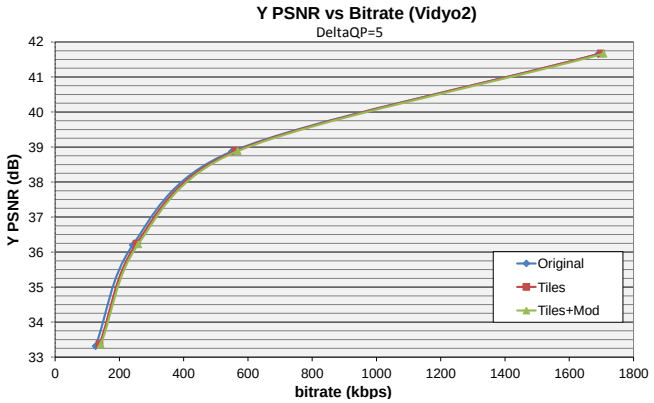

Figure 14. RD-plot of the results for the simulations of sequence “Vidyo2”.

defines a set of coding parameters that are suited for low latency video conferencing applications. The Quantization Parameter (QP) range of 22, 27, 32 and 37 was used, which spans the range of bitrates used in video conferencing applications. In Figure 14, the Rate-Distortion (RD) plot of the result for the simulations of the sequence from Figure 13 is displayed, where the other test sequences exhibit very similar results. In Table I, the Bjøntegaard Delta-rate (BD-rate) [9] overhead for the different scenarios, sequences and color components are displayed and Table II shows the average BD-rate overhead.

C. Evaluation

In general, Tiles as well as our modifications have a higher impact on the performance at lower rate points (high QP). This can be explained by the distribution of the bitrate in the coded stream. While at high QP values a high percentage of the available bitrate is used for the prediction information, this distribution is shifted for low QP values where the

 ISBN: 978-1-61208-265-3

Table I. BD-rate overhead when using Tiles or Tiles and our proposed encoder modifications for each test sequence.

Sequence		Y	U	V
SideBySide	Tiles	5.26%	4.13%	6.96%
	Tiles+Mod	8.69%	8.39%	7.28%
Vidyo1	Tiles	4.73%	2.13%	4.81%
	Tiles+Mod	8.47%	6.03%	8.31%
Vidyo2	Tiles	2.92%	0.06%	0.10%
	Tiles+Mod	5.02%	3.07%	1.66%

Table II. Average BD-rate overhead when using Tiles or Tiles and our proposed encoder modifications.

	Y	U	V
Tiles	4.3%	2.11%	3.95%
Tiles+Mod	7.39%	5.83%	5.75%

main part of the available bitrate is used for coding of the transform coefficients. Table II shows the average rate losses for using Tiles and for using Tiles in combination with our encoder restrictions.

VI. Conclusion

In this paper, a method for reordering of Tiles in an HEVC coded bitstream is proposed, that works on a very high syntax level and does not require any entropy de-/encoding of the bitstream. The resulting bitstream again conforms to the HEVC standard and can be decoded by any conforming decoder. In order to achieve this flexibility, Tiles were used in combination with Slices and some modifications to the encoder were applied to remove all remaining dependencies between neighboring Tiles.

Using Tiles and the proposed encoder modifications yields a small compression loss as shown in Section V. However, it adds the ability to arbitrarily reorder Tiles in a low complexity manner to create new bitstreams with different layouts that conform to the HEVC standard. In addition, it allows for parallel processing at the encoder as well as the decoder.

The limitations of this method result from the definition of Slices and Tiles. Since Tiles always contain a defined number of CTUs, the resolution of the Tiles has to be a multiple of the CTU size (usually 64×64 or 32×32). In addition, Tiles are defined using a grid layout, so with the proposed reordering operation, all Tiles must have the same dimensions.

Although the implementation and experiments were done using HEVC draft 6 [3], the key concepts used in the scope of our approach underwent only small changes up to the latest draft 9 [10] in a way that implementing the described approach using draft 9 is still possible. This will most likely also be true for the finished standard since only small changes are to be expected from working draft 9.

References

[1] M. Willebeek-LeMair and Z.-Y. Shae, "Videoconferencing over packet-based networks," IEEE Journal on Selected Areas in Communications, vol. 15, no. 6, pp. 1101–1114, August 1997.

[2] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," Signal Processing Magazine, IEEE, vol. 20, no. 2, pp. 18–29, March 2003.

[3] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 6," document JCTVC-H1003, JCT-VC, San Jose CA, USA, Tech. Rep., February 2012.

[4] A. Fuldseth, M. Horowitz, S. X. A. Segall, and M. Zhou, "Tiles," document JCTVC-F335, JCT-VC, Torino, Italy, Tech. Rep., July 2011.

[5] K. Misra and A. Segall, "New results for parallel decoding for Tiles," document JCTVC-F594, JCT-VC, Torino, Italy, Tech. Rep., July 2011.

[6] P. Amon, M. Sapre, and A. Hutter, "Compressed domain stitching of HEVC streams for video conferencing applications," in Packet Video Workshop (PV), 2012 19th International, pp. 36–40, May 2012.

[7] I.-K. Kim, K. Sugimoto, K. McCann, B. Bross, W.-J. Han, J.-R. Ohm, and G. Sullivan, "High efficiency video coding (HEVC) test model 6 (HM 6) encoder description," document JCTVC-H1002, JCT-VC, San Jose CA, USA, Tech. Rep., February 2012.

[8] F. Bossen, "Common test conditions," document JCTVC-H1100, JCT-VC, San Jose CA, USA, Tech. Rep., February 2012.

[9] G. Bjøntegaard, "Calculation of average PSNR differences between RD curves," document VCEG-M33, ITU-T Q6/16, Austin TX, USA, Tech. Rep., April 2001.

[10] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 9," document JCTVC-K1003, JCT-VC, Tech. Rep., October 2012.

 ISBN: 978-1-61208-265-3