Multi-rate encoding for HEVC-based adaptive HTTP streaming with multiple resolutions

Damien Schroeder, Adithyan Ilangovan and Eckehard Steinbach
Chair of Media Technology, Technische Universität München, Munich, Germany
(damien.schroeder, adithyan.ilangovan, eckehard.steinbach @tum.de)

Abstract—Adaptive HTTP streaming requires a video to be encoded at different rates and qualities called representations. The encoding of multiple representations with the new video coding standard HEVC is computationally complex. In this paper, we propose a multi-rate encoding method which reduces the complexity of encoding a video at multiple spatial resolutions. We first examine block structure similarities at different resolutions and propose a method to derive the block structure for a low resolution representation from a reference high resolution encoding. The derived block structure is used to speed up the encoding of low resolution representations. We further consider the content of the videos in order to achieve a rate-distortion (RD) performance similar to independent HEVC encoding. Experimental results show that the encoding time can be reduced by 50% on average for a low resolution video without degrading the RD performance.

I. INTRODUCTION

Today’s video streaming systems are mostly implementing the adaptive HTTP streaming paradigm [1], where the compressed video content is stored on an HTTP server at different bitrates called representations. The streaming clients request the representations based on a client-side adaptation algorithm, e.g., in order to match their current throughput. The bitrate of a compressed video can be modified by changing either its spatial resolution, its temporal resolution (i.e., frame rate) or its signal fidelity (i.e., different levels of distortion introduced by lossy compression). Scalable codecs (such as [2]) can encode videos at different qualities using multiple layers in order to have multiple bitrates, but the rate-distortion (RD) performance of scalable codecs is lower than in the single layer coding case, especially when the number of layers is high. Another possibility to obtain multiple bitrates is to transcode the video content from one compressed stream to another. However, the RD performance is also degraded compared to single layer coding due to requantization during the transcoding process.

Different from these methods, this paper concentrates on the simultaneous encoding of multiple independently decodable representations of a video, also called multi-rate encoding [3]. In particular, we investigate the case where a video has to be encoded at different spatial resolutions. In adaptive HTTP streaming, a video is generally encoded at different resolutions in order to match the various screen resolutions of the clients. How to choose the optimal set of video resolutions and bitrates has been the subject of recommendations from vendors and has been investigated in the case of a network with specific constraints [4].

Encoding a video into many representations is a computationally complex process, especially when using High Efficiency Video Coding (HEVC) [5], a recent standard which offers improved RD performance compared to previous standards, at the cost of increased encoding complexity. At the encoder side, the available computational power limits the number of representations that can be encoded simultaneously in the case of live streaming, while the storage place is the main limiting factor for the number of representations in the case of video-on-demand.

In multi-rate encoding, the redundancy introduced by encoding the same video at different bitrates can be exploited in order to reduce the overall encoding complexity. The idea was first introduced in 2002 in [6], where motion estimation is performed only once for multiple streams, but the encoding time gain of the method is not assessed. A similar idea is implemented in [3] in the context of a VP8 encoder, but the resulting RD-performance of the multi-rate encoder is severely degraded compared to the single rate encoder. In our previous work [7], we presented a method which reuses the HEVC block structure from a reference representation at a high quality in order to reduce the encoding time for lower quality representations. However, all of this previous work only considered videos at a single spatial resolution.

In contrast, in this paper we consider the case where a video is encoded with HEVC at multiple spatial resolutions. Our contributions are the following:

• We examine the similarities of the block structure of an HEVC encoded video across different resolutions.
• We propose a method to derive the block structure at a low resolution from a reference high resolution encoding. We use this derived block structure to speed up the encoding at a low resolution.
• We characterize the video content based on the high resolution reference encoding and adapt the parameters of the proposed method accordingly.

The rest of the paper is organized as follows: In Section II, we analyze the block structure similarities across different resolutions and propose a method to derive a low resolution block structure from a high resolution reference encoding. In Section III, we explain how to reuse the derived block structure in order to speed up the encoding process. In Section IV, we...
consider the content dependency of the proposed method. We present our experimental results in Section V and conclude the paper in Section VI.

II. BLOCK STRUCTURE AT DIFFERENT RESOLUTIONS

A. HEVC block structure

Compared to previous standards such as H.264/AVC, HEVC introduces a new block structure based on a quadtree partitioning of the frames [8]. The frame is first divided into equal-sized Coding Tree Units (CTUs), which can have a size between $8 \times 8$ and $64 \times 64$ pixels. A CTU is then composed of Coding Units (CUs), at which level the prediction mode (intra/inter) is determined. A quadtree is used to partition a CTU into CUs. The size of a CU is given by its depth in the quadtree. Depth 0 corresponds to the root of the tree and thus the largest possible CU size (which is equal to the CTU size). As the depth increases, the CU size decreases. The block structure is determined by the HEVC encoder during the rate-distortion-optimization (RDO) process.

B. Block structure similarities

To determine the similarities in the block structure of a video encoded at different resolutions, we encode a test video with the HEVC reference software HM 16.4 [9] at three different resolutions (original at 1920 $\times$ 1080 pixels and two downsampled versions at 1280 $\times$ 720 and 640 $\times$ 360 pixels). Fig. 1 shows the resulting block structure for the 20th frame of the ParkScene sequence [10] at quantization parameter (QP) 22 and with a CTU size chosen as $64 \times 64$ at all three resolutions. Although the $64 \times 64$ CTUs do not cover the same image area at different resolutions, we can notice that certain areas of the frame will be encoded similarly at different resolutions, in the sense that homogeneous regions such as the tree on the left tend to be coded with large CUs, whereas frame regions with high detail level tend to be coded with small blocks.

C. Block matching across resolutions

If we want to reuse the block structure information from a high resolution encoding of the video to speed up lower resolution encodings, we need to match the block structure at a high resolution to the block structure at a low resolution. In the case where the downsampling ratio is a multiple of 2, the block structure can be easily matched across resolutions. E.g., we can see in Fig. 2 that a block at depth 1 at 360p corresponds to a block at depth 0 at 720p. However, there is no direct correspondence between the blocks at different resolutions if the downsampling ratio is different than a multiple of 2. As an example, we can see in Fig. 2 that a CTU at 720p covers a frame area which is larger than one CTU but smaller than 4 CTUs at 1080p.

D. Proposed block structure derivation algorithm

We propose an algorithm which enables us to extract block structure information from a high resolution video for a lower resolution with an arbitrary downsampling ratio. In the following we will use these definitions:

- The reference encoding is the high resolution encoding with an unmodified encoder, which gives the reference block structure.
- The derived block structure is a low resolution block structure which is derived from the reference block structure.
- An original encoding is an independent encoding with an unmodified HEVC encoder.
A dependent encoding is a low resolution encoding with a modified HEVC encoder which reuses the derived block structure in order to speed up the encoding process. This part is explained in Section III.

The proposed derivation of the block structure information is done at the CTU level, i.e., we compute the derived block structure for the low resolution representation CTU by CTU. First, the CU at depth 0 from the first CTU is selected in the low-resolution video. Then, we select the area $A$ in the reference encoding that corresponds to the current CU. Next, we measure the percentage $p_0$ of $A$ encoded at depth (less than or equal to) 0, i.e., highest possible CU size. In general, the percentage $p_i$ with $i \in \{0, 1, 2\}$ is defined as the percentage of the corresponding area in the reference encoding with depth less or equal to $i$. If $p_0$ is greater than or equal to a threshold $\tau$, then the current CU is not split and the process moves on to the next CTU. On the contrary, if $p_0$ is less than $\tau$, then the current CU is split into four smaller CUs at depth 1 ($CU_1$, $CU_2$, $CU_3$ and $CU_4$). This process is recursively repeated for all the CUs in order to traverse the quadtree, until the process is finished for each CU or the minimum CU size is reached. As an example, we continue with the decision whether to split/not split each of the depth 1 CUs. Starting with $CU_1$, we select the area $A_1$ in the reference encoding that corresponds to $CU_1$. Now, we measure the percentage $p_1$ of $A_1$ at depth less than or equal to 1. If $p_1$ is greater than or equal to $\tau$, then we do not split $CU_1$ and move on to the next CU. However if $p_1$ is less than $\tau$, we further split $CU_1$ into four smaller CUs at depth 2 ($CU_{11}$, $CU_{12}$, $CU_{13}$ and $CU_{14}$). The algorithm is represented graphically in Fig. 3.

The threshold $\tau$ determines how conservative the algorithm is. For instance, for $\tau = 100$, a CU will be derived at highest possible CU size only if 100% of $A$ is at depth 0. In comparison, for $\tau = 60$, a CU will be derived at highest possible CU size only if at least 60% of $A$ is at depth 0, which means that a part of the area $A$ can have a higher depth. Fig. 4 shows the derived block structure for a frame of a 720p video (derived from a 1080p reference encoding), for two different thresholds. It can be seen that for $\tau = 60$, more blocks tend to be derived at larger CU size compared to $\tau = 80$. The original block structure obtained from an independent encoding with an unmodified encoder is also shown for reference in Fig. 4c.

E. Similarity quantification

In order to quantify the similarity between the derived block structure and the block structure of the original encoding, we calculate the percentage of the area of the frames where the blocks have the same depth. If the block depth is not identical, it can either have a greater depth (i.e., a smaller block size) or lower depth (i.e., a larger block size). Fig. 5 shows the comparison between Fig. 4b and 4c. The yellow region is where both have same block depths, the dark green
Fig. 5: Areas in the original block structure of Fig. 4c with greater (dark green), same (yellow) or lower (light green) depth when compared with the derived block structure shown in Fig. 4b ($\tau = 60$).

Fig. 6: Average percentage of areas in the original encoding with depths lower, identical or greater than the derived block structure, obtained by averaging over 11 video sequences.

indicates greater depth and the light green indicates lower depth. To generalize the results, we repeat the methodology for 11 sequences and average the results and show the percentages of these regions for two different $\tau$ in Fig. 6.

When $\tau$ decreases, the number of blocks of the derived structure at lower depth increases. Thus, the percentage of area having greater depth in the original structure increases, which can be seen by the increase in the dark green region in Fig. 6, when $\tau$ goes from 80 to 60. The percentage of the area where the block depth is not identical (sum of dark and light green regions) is not negligible. Therefore, we cannot directly reuse the derived block structure for the dependent encodings. Still, a majority of blocks of the original structure will have lower or same depth as the derived block structure. E.g., for $\tau = 80$, roughly 95% of the frame area will have lower or same depth. We will exploit this fact when we reuse the block structure information, as explained in the next section.

III. DERIVED BLOCK STRUCTURE REUSE

The block structure is determined recursively in HM 16.4 starting with the largest CU size, that is, depth 0. Different modes are examined by the encoder and the best candidate in the RD sense is chosen. Then the block is split into 4 subblocks (depth + 1) and the RDO is applied to these subblocks, and so on. The block structure combined with the modes which give the smallest RD cost are chosen in order to maximize the encoding RD performance.

As shown in Section II-E, most of the area in the original encoding either has lower or same depth as the derived block structure. Combining this observation and the fact that the RDO process of the HEVC encoder is implemented recursively starting from the lowest depth, similar to [7], we propose to stop the RDO process of the dependent low resolution encoding at the depth given by the derived block structure, as shown in Fig. 7. The depths above the green dotted line are checked during RDO of low resolution encodings. On the contrary, the proposed modified encoder does not check the depths below the green dotted line which leads to significant encoding time savings. As shown in Fig. 6, there is a small percentage of the area where the depth is greater in the original encoding than in the derived block structure. As the RDO process is stopped at the depth of the derived block structure, a suboptimal block size in the RD sense will be chosen for the dependent encoding. However, the overall RD loss due to such cases should be small because this concerns only a small percentage of the frame area, e.g., roughly 5% (dark green region) in Fig. 6a for $\tau = 80$.

In general, when $\tau$ is decreased, the percentage of the area at greater depth (dark green region) increases (cf. Fig. 6). Consequently, there will be a higher RD loss. In the case of lower $\tau$, on average, the RDO process is stopped earlier than for higher $\tau$ and so there will be higher encoding time savings. Thus, there is a trade-off between the encoding time savings and the RD-loss, which can be balanced by $\tau$.

IV. THRESHOLD DETERMINATION

A. Observations

We next discuss how to choose the threshold $\tau$ for the proposed method. To gain first insights, we evaluate the proposed method for different $\tau$'s. We compare our implementation with the unmodified HM 16.4 encoder [9]. The RD-performance difference is measured using the Bjontegaard delta rate (BD-rate) [11] and the Bjontegaard delta PSNR (BD-PSNR) [12]. We first encode the 1080p reference video sequence using the unmodified HM 16.4 encoder. Next, we derive the block structure depending on $\tau$ as explained in Section II-D and reuse it for the dependent encodings as explained in Section III. All sequences are encoded with the
random access, main profile defined in [10]. The CTU size is fixed to $64 \times 64$ pixels and thus the maximum CU depth is 3.

We use six video sequences for this initial comparison: Sunflower, RushHour, Kimono, ParkScene, RiverBed and DucksTakeOff ([13], [10]). We evaluate the proposed method for four $\tau$ (60, 70, 80 and 90). The resulting BD-rate is shown in Fig. 8. We also calculate the average depth $d_{\text{avg}}$ of the 1080p reference encoding. Two observations can be made. First, for all sequences, a greater $\tau$ leads to a lower BD-rate than for a smaller $\tau$, as explained in Section III. Second, for a fixed $\tau$, sequences with lower $d_{\text{avg}}$ tend to have a higher BD-Rate than the ones with higher $d_{\text{avg}}$. This can be intuitively explained by the impact of stopping the RDO before actually reaching the optimal depth. The probability of not reaching the optimal depth increases if the reference depth, and thus the derived depth, are low. In order to balance this effect, we should use a high $\tau$ for sequences with a low $d_{\text{avg}}$. Similarly, we can afford to use comparatively small $\tau$ for a high $d_{\text{avg}}$ sequence.

However, in the multi-rate encoding process, the parameter $\tau$ should be available before the entire sequence is encoded at the reference high resolution in order to encode the dependent low resolution representations in parallel. Determining $\tau$ on a frame by frame basis allows us to encode the dependent frame directly after the reference frame has been encoded. Fig. 9a shows the average depth $d_{\text{avg}}$ from each 1080p reference frame, and the resulting BD-rate at 720p for 50 frames of the RiverBed video for two different $\tau$ values. We fit a linear curve to the point cloud as a first order approximation in order to get a coarse estimation of the BD-rate depending on the average depth. In Fig. 9b, we repeat this process and average over 11 sequences.

### B. Proposed threshold choosing method

In order to keep the BD-rate at a low level while reducing the encoding time, we propose to adapt the value of $\tau$ for every frame based on the $d_{\text{avg}}$ of the current frame of the reference encoding. We choose an arbitrary low BD-rate value of 1.5% and based on Fig. 9b, we propose a simple mapping described in Table I for 720p dependent encodings, where the average depth $d_{\text{avg}}$ of the current reference frame is mapped to a threshold $\tau$ for the corresponding dependent encoding frame. We do not consider a threshold of 100 as our preliminary investigation showed us that this results in a negligible encoding time gain. We perform the same methodology to find a mapping for the 360p dependent encodings as shown in Table II.

### V. RESULTS

#### A. Settings

The RD performance and the encoding time are used as comparison metrics. The difference in encoding time ($\Delta T$) between the original HM encoder and our implementation is measured as the difference of the total encoding time for 4 representations at different qualities (fixed QP 22, 27, 32

<table>
<thead>
<tr>
<th>$d_{\text{avg}}$</th>
<th>[0, 1.01)</th>
<th>[1.01, 1.45)</th>
<th>[1.45, 2.16)</th>
<th>[2.16, 3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$</td>
<td>90</td>
<td>80</td>
<td>70</td>
<td>60</td>
</tr>
</tbody>
</table>

**TABLE I: Mapping between $d_{\text{avg}}$ and $\tau$ for 720p.**

<table>
<thead>
<tr>
<th>$d_{\text{avg}}$</th>
<th>[0, 1.5)</th>
<th>[1.5, 1.65)</th>
<th>[1.65, 1.79)</th>
<th>[1.79, 3]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$</td>
<td>90</td>
<td>80</td>
<td>70</td>
<td>60</td>
</tr>
</tbody>
</table>

**TABLE II: Mapping between $d_{\text{avg}}$ and $\tau$ for 360p.**

![Fig. 8: BD-rate for different thresholds for the 720p sequence and average depths of the reference encoding.](image1)

![Fig. 9: Linear fit of the BD-rate for each frame of the dependent 720p sequence. The $d_{\text{avg}}$ is calculated, per frame, from the reference 1080p sequence.](image2)
4. 4 representations are needed in order to be able to calculate BD-PSNR and BD-rate. All encodings are performed on an Ubuntu server with a Xeon X5690 CPU at 3.47 GHz. 10 video sequences defined in [10], [13] with original resolution of 1920 × 1080 pixels and frame rates between 24 fps and 50 fps are used.

### B. Encoding results

Tables III and IV show the performance of the proposed encoding method compared to the original HM 16.4 reference for 720p and 360p, respectively. The encoding time for the 4 representations is reduced on average by 50.77% and 47.69% for 720p and 360p, respectively. Fig. 10 illustrates the encoding time gains with the proposed multi-rate encoding method in the case of the Parkscene sequence. In terms of RD performance, the dependent 720p encodings show an average BD-rate increase of 1.20% and a BD-PSNR decrease of 0.05%. The dependent 360p encodings have an average BD-rate increase of 2.16% and a BD-PSNR decrease of 0.11%.

### VI. Conclusion

In this paper, we consider the multi-rate encoding of a video sequence at different spatial resolutions. We first examine the similarities of the block structure of HEVC encoded videos across different resolutions. Then, we propose a method to derive the block structure for a low resolution representation based on the reference block structure of a high resolution representation. Furthermore, we describe how to reuse the derived block structure in order to speed up the dependent encoding of low resolution representations. Finally, we consider the content of the video sequences and propose a method to choose the parameter for the block structure derivation based on the content characteristics. Encoding results show that we can speed up the encoding of 720p and 360p representations by 50.77% and 47.69% on average based on a reference encoding at 1080p, while the BD-rate increase is kept low at 1% and 2%, respectively.

### REFERENCES


