Motion Compensation-based Low-Complexity Decoder Side Depth Estimation for MPEG Immersive Video
Patrick Garus, Felix Henry, Thomas Maugey, Christine Guillemot

To cite this version:
Patrick Garus, Felix Henry, Thomas Maugey, Christine Guillemot. Motion Compensation-based Low-Complexity Decoder Side Depth Estimation for MPEG Immersive Video. MMSP 2022 - IEEE 24th International Workshop on Multimedia Signal Processing, Sep 2022, Shanghai, China. pp.1-6. hal-03833539v2

HAL Id: hal-03833539
https://inria.hal.science/hal-03833539v2
Submitted on 7 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract—Decoder-Side Depth Estimation (DSDE) is a system firstly enabled in the novel MPEG Immersive Video (MIV) coding standard. In DSDE, only texture components are coded, while the depth is estimated at the decoder-side. This is motivated by previous work, which has shown high coding gain and pixel rate savings in DSDE. However, the computational complexity remains a concern, as high quality depth search has a high runtime and memory requirement. In this work we extend the concept of depth estimation to depth recovery. Using this mode, the decoder-side depth information is recovered through motion compensation utilizing the displacement vectors contained in the texture bitstream. This strategy enables us to replace most of the complex depth estimation processes with a simple motion compensation step, a decision that is drawn on the encoder-side and signaled per coding unit. With only minor losses in terms of synthesis PSNR and similar perceptual quality in terms of MS-SSIM, the complexity is significantly reduced. Depending on the acceptable loss, up to 80% of the moving objects depth may be motion compensated instead of estimated by a depth estimator translating into a speed-up of a factor of 104 for inter-frames compared to the reference depth estimator.

Index Terms—Decoder Side Depth Estimation, MPEG Immersive Video, Immersive Video Coding

I. INTRODUCTION

The Motion Pictures Experts Group (MPEG) finalized a novel immersive video standard denoted as MPEG Immersive Video (MIV) [1] [2]. It enables the user to freely navigate through captured or computer-generated volumetric video, i.e. enabling 6 Degrees of Freedom (DoF). It is compatible with the common multiview plus depth (MVD) format as well as with the emerging Multi-Plane Image format [3]. While it may be considered as a successor of 3D-HEVC, it does not follow the technical foundations of 3D-HEVC [4], namely inter-view and inter-component prediction, and specialized depth coding tools. Instead, the MIV is designed to operate on top of existing 2D video codecs and serves to describe the atlas format, which is the output of the MIV encoder. The atlas construction includes a pruning step which extracts images patches considered non-redundant among the views. This process is followed by the coding of depth and texture atlases independently using a 2D video codec. The decision to remain codec agnostic particularly impacts the coding of depth maps, as they cannot benefit from depth coding tools impacting the coding gain and perceptual quality of the rendering. Consequently, a Geometry Absent (GA) profile was introduced, which enables Decoder-Side Depth Estimation (DSDE) bypassing all issues related to the depth coding of MIV. In DSDE, the coding efficiency is significantly improved at the cost of increased client-side complexity.

In this paper, we propose a solution to the complexity problem of the GA profile, which is based on motion compensating the depth maps using the motion information provided by the texture decoder. This allows to omit the majority of depth estimation processes for Inter frames, which significantly reduces the runtime. All modern video codecs, such as AVC [5], HEVC [6] or VVC [7], will apply some form of motion compensation on the texture components. We re-use this available information to perform motion compensation on the depth component on a block-basis. This decision is drawn on the encoder-side and signaled. We avoid the transmission of residual information, as depth coding and modifications to the 2D codec is restricted by the MIV design. Consequently, depth estimation will be used only if the motion compensation fails given a certain error tolerance. Furthermore, this proposal presents an opportunity to think of depth estimation as part of the whole decoding framework by involving the texture decoder. Given the high potential in reducing the complexity of DSDE, we refer to our proposal as Low-Complexity DSDE (LC-DSDE).

The remainder of this paper is structured as follows: in section II, we introduce the MIV standard and the latest research related to DSDE, in section III we describe our proposed Low-Complexity DSDE approach, in section IV we discuss the results and section V concludes the paper.

II. DECODER SIDE DEPTH ESTIMATION IN MPEG IMMERSIVE VIDEO

The choice and design of the atlas construction tools of MIV are mostly motivated by reducing pixel rate. Pixel rate is defined as the number of luma pixels that can be processed by a hardware video decoder. A test sequence with a resolution of 2048 x 1088 and 16 views requires 64 Megapixels at 30
frames per second and therefore, only 8 out of 16 views can be coded into the atlases. Given that the CTC of MIV contains sequences with up to 64 views, the example clearly shows the demand for reducing pixel rate and why DSDE has become an attractive system.

The basic DSDE (B-DSDE) system is shown in Fig. 1a), which has been firstly evaluated in [9]. Textures $T$ are coded using any 2D video encoder. At the decoder side, decoded textures $T^*$ are used to estimate corresponding depth maps $D^+$. A synthesizer is then rendering intermediate views $S_{BDSDE}$. The DSDE system has been studied throughout the MIV standardization process and adopted as the Geometry Absent profile [15]. Given the high coding efficiency, the single concern of this system is the client-side complexity. In order to support the depth estimator in estimating depth maps faster and with higher accuracy, a dedicated Geometry Assistance SEI message has been adopted based on [14]. This approach allows to move significant amount of complexity to the encoder-side. However, more tools are necessary to further reduce the complexity of DSDE. One way is to exploit synergies with other modules of the DSDE system. Our proposal falls in this class and is therefore only possible in a coding system, taking advantage of the presence of the 2D texture decoder.

Beyond architectural and signaling related improvements, the depth estimator itself can be adapted to the context of DSDE. A point-to-block matching strategy improves the performance if depth are estimated from decoded textures [10]. Furthermore, a depth refinement process can be used to further improve the quality of depth maps estimated from decoded textures [11].
III. PROPOSED DEPTH RECOVERY

It is known from 3D-HEVC studies that the motion in textures often correlates with the motion in the depth maps and therefore may serve as a useful predictor in the context of inter-component coding [16]. In LC-DSDE, we try to utilize the motion information already coded in the texture bitstream to motion compensate the depth maps and therefore, reconstruct depth for Inter frames, the proposal is used. Frames do not benefit from our approach. For subsequent Inter frames, the depth recovery module on the decoder-side is utilizing the modified block-based depth estimator. The bitstream contains the result of the depth analysis and communicates to the depth recovery, which prediction units shall be compensated using the decoded depth at a previous time instance and which shall be computed using depth estimation. The first frame is estimated as usual with the original depth estimator, as Intra frames do not benefit from our approach. For subsequent Inter frames, the proposal is used.

After coding all textures, the depth analysis is performed. For all inter-coded CUs a displacement vector $[mv_x, mv_y]$ is fetched from the texture bitstream. The FFmpeg opensource library provides tools to extract these displacement vectors from the bitstream in a simple way for h.264/AVC [5]. As TMIV is compatible with any 2D codec the usage of AVC in the following experiments is possible. Furthermore, the partitioning associated with all displacement vectors is available. The displacement vectors, the decoded Textures $T^*$ as well as the reference depth maps $D_{ref}$ are provided to the Depth Analysis module. This module performs the following tests for all inter CUs:

1) Perform block-based depth estimation on the CU and derive $D_{bbDE}$.
2) Perform motion compensation utilizing the previous intra-coded depth frame and the associated displacement vectors $[mv_x, mv_y]$ and derive $D_{mc}$.
3) Compute the $L_2$-distance between the reference depth $D_{ref}$ and the two depth map candidates $D_{mc}$ and $D_{bbDE}$.
4) Select the method based on the evaluation $L_{2,bbDE} > sL_{2,mc}$, with the error tolerance factor $s$. If the statement is true, the CU is recovered at the decoder-side through motion compensation. If the statement is false the CU is estimated by the block-based depth estimator.

The decision is signaled as part of the MIV bitstream and does not require normative changes to the 2D encoder.

IV. EXPERIMENTAL SETUP

We use the Common Test Conditions (CTC) of MPEG Immersive Video [12] as a starting point to set up our evaluation framework. In order to study the potential of our proposal, the 2D encoder configuration has been modified in two ways. Firstly, Intra encoder is disabled in inter-frames to maximize the availability of motion data. Secondly, we disable B-Frames, as it will simplify the interpretation of the extracted motion information. This enables us to focus the evaluation of the depth recovery on the quality preservation and speed-up capabilities. Without this modification, the speedup as well as the quality degradation is expected to be smaller. However, the accuracy of the prediction can also be improved if B-frames were enabled. Furthermore, more recent coding standards than AVC may provide displacement vectors with higher accuracy or affine motion information. Consequently, the CTC modifications are not solely to our advantage but necessary to isolate the impact of the proposed depth recovery tool more clearly. Finally, we will investigate the high-bitrate scenario ($Q_T = 25$) for 5 sequences, excluding the still images of the CTC, as they do not benefit from this approach.

We utilize the depth estimation reference software (DER8), developed and continuously improved by MPEG [8]. The configuration and settings are refined for the sequences of the CTC and we adopt them without modification. In order to utilize our proposal, DER8 has to be adapted to support very small blocks, i.e. estimate depth information independently for each CTU. In AVC, the block size of motion compensated CUs can vary between $4 \times 4$ to $16 \times 16$. We enable DER8 to extract the partitioning from the AVC bitstream and to estimate each motion compensated patch individually. It is therefore an extension to the block-based approach presented in [14]. Estimating depth for very small blocks naturally has an impact on the estimation quality. An example of the resulting depth quality is shown in Fig. 2. The left depth map originates from the unmodified full-frame DER8 (fDE), while the right depth map was generated using the block-based DER8 (bbDE). The “blockiness” can be clearly seen, surprisingly however, the depth map appears globally correct.

The amount of motion compensated blocks is controlled by tolerating a higher error, i.e. we define a block as a motion compensated block if $L_{2,bbDE} > sL_{2,mc}$ with an error tolerance $s$ and the $L_2$ distance to a reference depth $D_{ref}$. Here, we use the unmodified DER8 depth maps as a reference. We test the values $s = [0.1, 0.25, 0.50, 0.75]$.
An example of the chosen blocks is shown in Fig. 3. For each value $s$, we compute the amount of motion compensated blocks as well as the resulting synthesis PSNR and MS-SSIM [13]. Furthermore, the depth estimation runtime is provided comparing the block based DERS8 with different thresholds $s$ and the unmodified DERS8. Computation was performed on an Intel Xeon (R) CPU E5-2520 @ 2.00 GHz. As the bitrate required to signal this decision is negligible in the immersive video context, we focus on the objective and subjective quality of the depth maps and the synthesizes views. The more blocks are motion compensated, the better the reduction in complexity. Simultaneously, more errors are introduced into the depth maps.

V. EXPERIMENTAL RESULTS

We first investigate the impact of our proposal on the depth maps quality, secondly on the synthesis quality and finally investigate the benefit on the main goal of the proposal: the complexity reduction. Fig. 4 visualizes depth maps as well as corresponding synthesis for all thresholds as well as the ffDE reference. We show the last frame of the GOP, frame 16, which reflects the highest error. Furthermore, we chose to show patches containing motion, in order to make the artifacts of our proposal visible. In the following, we refer to Fig. 4 when discussing the visual performance.

A. Depth Maps Quality

In general, a quality degradation is expected in all cases, since no residual is allowed to be send, as otherwise the approach would steer towards depth coding. The impact of the last frame in the GoP is most severe for $s = 0.1$, also due to error propagation. However, it is apparent that even for $s = 0.75$ an increase in artifacts may appear. This is due to erroneously estimated depth in a block-based approach. As expected, most artifacts become visible if previously occluded objects appear because they cannot be recovered by motion compensation.

B. View Synthesis Quality

The degradation of view synthesis quality is quite severe for $s = 0.1$, with strong artifacts and visible "holes". Nevertheless, we observe that many artifacts are not due to the motion-compensation of depth, but rather due to the block-based approach of DERS8, because several artifacts appear at areas without any motion. As expected, most artifacts become visible if previously occluded objects appear because they cannot be recovered by motion compensation. If a high amount of error is tolerated ($s = 0.1$), the typical motion compensation artifacts in depth maps translate into an increased level of noise-like artifacts in the synthesized textures. Yet, even for $s = 0.1$ most of the synthesized objects are reconstructed accurately.

C. Complexity-Distortion Compromise

The main goal of the proposed LC-DSDE is to minimize the complexity (with minor impact on quality), as it is the main concern of DSDE. The objective metrics and the runtimes are shown in Tab. I for all tested sequences and thresholds $s$. The unmodified DERS8 is denoted as ffDE. We observe the following: first, the synthesis PSNR degrades with a lower threshold, and therefore, with more motion compensated blocks, as expected. Naturally, this is dependent on the amount of motion in sequence. Therefore, Frog has a loss of $1.63 \, dB$, Painter and Chef2 losses of around $1 \, dB$. In contrast, sequences with little motion have a lower degradation at $s = 0.1$. This is confirmed by the MS-SSIM metric. However, $s = 0.1$ is rather an extreme-case. Another interesting observation is, that the synthesis PSNR is rather similar between $s = 0.5$ and $s = 0.75$. This indicates that with $s = 0.5$, the majority of "safe to use" motion vectors are found, which do not significantly impact the quality in contrast to depth estimation. Again, this is confirmed by the MS-SSIM metric, which indicate the same value of 0.960 for $s = 0.5$ and $s = 0.75$. The next observation is, that despite the saturation of quality from $s = 0.5$, there is still a large gap towards the unmodified, full frame DERS8 synthesis PSNR. On average, a loss of 0.44 dB is seen, with the smallest loss of 0.12 dB for Kitchen and the largest loss of 0.73 dB for Chef2. Due to the saturation of quality starting from $s = 0.5$, we conclude, that large parts of the degradation starting from this threshold are due to the block-based DERS8 approach and not due to the applied motion compensation.

Tab. I also summarizes the amount of motion compensated blocks, taking only into account the blocks with non-zero motion vectors. In case of $s = 0.1$, on average about 80% of blocks with indicated motion are used to motion compensate the depth blocks, instead of estimating them. The fraction gradually increases with the threshold to around 34% with $s = 0.75$. While we observed a saturation of quality loss from $s = 0.5$, there is indeed more blocks being compensated from $s = 0.75$ to $s = 0.5$. The runtime is also shown in this Table for all thresholds and compared to the unmodified DERS8.

### TABLE I

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Motion Compensated [%]</th>
<th>Depth Estimation runtime [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Painter</td>
<td>0.1 0.25 0.5 0.75</td>
<td>88.71 77.37 60.32 41.32</td>
</tr>
<tr>
<td>Kitchen</td>
<td>0.1 0.25 0.5 0.75</td>
<td>56.41 42.48 29.19 20.29</td>
</tr>
<tr>
<td>Dancing</td>
<td>0.1 0.25 0.5 0.75</td>
<td>79.57 65.62 49.19 37.45</td>
</tr>
<tr>
<td>Chef2</td>
<td>0.1 0.25 0.5 0.75</td>
<td>89.81 76.33 55.11 42.39</td>
</tr>
<tr>
<td>Frog</td>
<td>0.1 0.25 0.5 0.75</td>
<td>76.65 51.97 31.09 21.70</td>
</tr>
<tr>
<td>Painter and Chef2</td>
<td>0.1 0.25 0.5 0.75</td>
<td>35 98 149 170</td>
</tr>
<tr>
<td>Average</td>
<td>0.1 0.25 0.5 0.75</td>
<td>76.65 51.97 31.09 21.70</td>
</tr>
</tbody>
</table>

**AVERAGE SYNTHESIS PSNR AND MS-SSIM AS WELL AS THE AMOUNT OF MOTION COMPENSATED BLOCKS AND RUNTIME DIFFERENT THRESHOLDS $s$ AND THE ffDE REFERENCE. THE PERCENTAGE REFERS ONLY TO BLOCKS WHICH HAVE AN IDENTIFIED MOTION.**
Fig. 4. Depth maps and synthesized view of four CTC sequences. Frame 16 is shown for the unmodified reference in the B-DSDE system using ftDE as well as the LC-DSDE proposal for different thresholds.
Naturally, the runtime of the unmodified DERS8 is much larger, as it is applied on the full frame. Nevertheless, focusing the depth estimation on blocks, which are not compensated, can provide a speed-up from 105 ($s = 0.1$) to 18 ($s = 0.75$). Considering the quality degradation, the optimal point among the tested thresholds is $s = 0.5$, for which a speed-up of 22.2 for inter-frames is measured.

VI. CONCLUSION

In this paper, LC-DSDE was presented, which takes advantage of motion information contained in the h.264 coded texture bitstream in order to motion compensate the depth information instead of estimating it. It was shown that a significant speed-up of factor 105 to 18 can be achieved for varying levels of tolerated error using a non-CTC configuration. In future work, more recent codecs with advanced motion modeling may be used to perform the depth recovery. Finally, this proposal is further a proof of concept for the potential of DSDE systems benefiting from jointly operating the texture decoder and the depth estimator.

REFERENCES


