Received: 1 September 2021 | Revised: 1 December 2021 | Accepted: 17 December 2021

DOI: 10.4218/etrij.2021-0308

SPECIAL ISSUE

ETRI Journal WILEY

# Recursive block splitting in feature-driven decoder-side depth estimation

**Błażej Szydełko**[1] | **Adrian Dziembowski**[1] | **Dawid Mieloch**[1] | **Marek Domański**[1] | **Gwangsoon Lee**[2]

[1]Institute of Multimedia Telecommunications, Poznan University of Technology, Poznań, Poland

[2]Immersive Media Research Section, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea

**Correspondence**
Adrian Dziembowski, Institute of Multimedia Telecommunications, Poznan University of Technology, Poznań, Poland.
Email: adrian.dziembowski@put.poznan.pl

**Funding information**
Institute of Information & Communications Technology Planning & Evaluation, Grant/Award Number: 2018-0-00207

**Abstract**

This paper presents a study on the use of encoder-derived features in decoder-side depth estimation. The scheme of multiview video encoding does not require the transmission of depth maps (which carry the geometry of a three-dimensional scene) as only a set of input views and their parameters are compressed and packed into the bitstream, with a set of features that could make it easier to estimate geometry in the decoder. The paper proposes novel recursive block splitting for the feature extraction process and evaluates different scenarios of feature-driven decoder-side depth estimation, performed by assessing their influence on the bitrate of metadata, quality of the reconstructed video, and time of depth estimation. As efficient encoding of multiview sequences became one of the main scopes of the video encoding community, the experimental results are based on the "geometry absent" profile from the incoming MPEG Immersive video standard. The results show that the quality of synthesized views using the proposed recursive block splitting outperforms that of the state-of-the-art approach.

**KEYWORDS**
decoder-side depth estimation, immersive video

## 1 | INTRODUCTION

Many contemporary multimedia systems, for example, ones that utilize head-mounted displays to present visual data to their users, make it possible to virtually navigate through three-dimensional scenes [1]. Unfortunately, such a possibility yields the requirement of acquiring precise geometric information to provide a sufficient level of immersion, which is impossible to achieve if the quality of a displayed image is low [2]. For natural scenes, such a detailed description of the scene can be acquired using densely arranged multiview camera systems [3–6] because the quality of the estimated geometry increases with the number of used cameras [7]. It also increases the size of data required to process.

Sending a high amount of uncompressed data to a decoder is highly impractical. Thus, in any practical immersive video system, efficient encoding of an immersive video has to be applied. As immersive video applications gain widespread attention in the video processing community, there has been a noticeable increase in standardization efforts in this field [8]. One of the currently prepared standards, that is, MPEG Immersive video (MIV) [9], provides state-of-the-art

This is an Open Access article distributed under the term of Korea Open Government License (KOGL) Type 4: Source Indication + Commercial Use Prohibition + Change Prohibition (http://www.kogl.or.kr/info/licenseTypeEn.do).
1225-6463/$ © 2022 ETRI

methods for efficient encoding of multiview sequences. Although its main profile focuses on removing interview redundancy and packing of remaining data, the "geometry absent" (MIV GA) profile follows the idea of estimating the geometry of the scene at the decoder side [10]. In MIV GA, only a set of input views and their parameters are compressed and packed into the bitstream. However, in most cases, the geometry information (eg., in the form of a depth map for each view [11]) is available on the encoder side. This situation enables the encoder to send a set of features that could make it easier to estimate geometry in the decoder [12,13]. These features include block-wise information, such as a range of depth in a block or a skip flag that indicates that the depth in a block did not change from the previous frame.

This study evaluates different scenarios of feature-driven decoder-side depth estimation. We assess the influence of these scenarios on the bitrate of metadata, quality of final video reconstruction, and time of depth estimation performed at the decoder side. We also present a novel extension of the method supported in the MIV standard—the introduction of recursive block splitting in the feature extraction process. The proposed method allows the feature extractor to better fit to the edges of the objects. Thus, it improves the quality and decreases the computational time of depth estimation.

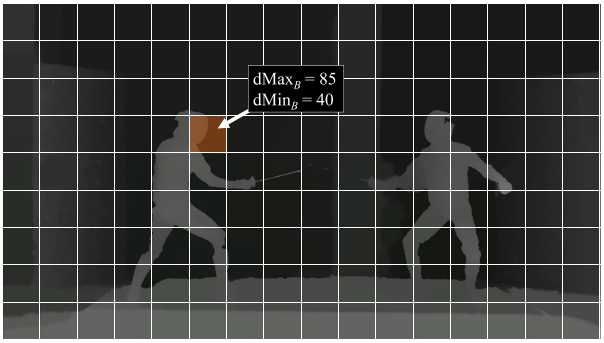


**FIGURE 1** Basic idea of depth range features: division of depth maps according to a 128 × 128 grid and sending two normalized disparity values (the smallest and largest ones) for each block *B*

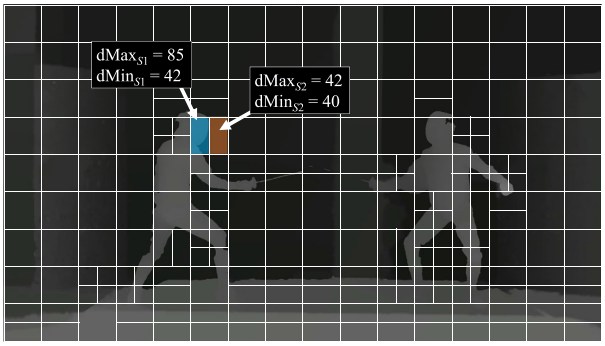


**FIGURE 2** Concept of block splitting: Block *B* from Figure 1 was split into two subblocks (*S*1 and *S*2) with a smaller depth range

## 2 | FEATURE-DRIVEN DSDE

The features derived from depth maps, which are available in the encoder, helps in solving the fundamental issues that arise from decoder-side depth estimation. First, such estimation highly influences the complexity of the decoding process as depth estimation becomes its part. Even the fastest approaches are computationally expensive and cannot provide high-quality depth maps in real time [14]. Moreover, as depth maps are estimated using compressed views, some degradation of their quality can be seen [15,16]. This compression-induced degradation can be decreased using refinement methods for compression artifacts reduction [17] or depth map enhancement [18,19]. However, the use of additional postprocessing steps increases the valuable time from acquiring the bitstream to present the virtual view to a final viewer.

A different approach is shown in feature-driven depth estimation [12], which assumes three different features already within the encoded bitstream. The first type is the depth range within each block of input depth maps (Figure 1). Sending this information to the decoder increases the quality of estimated depth by reducing possible estimation errors, which can be the effect of views compression, as depth within each block will be cropped to the provided range. Moreover, the estimation process can be significantly speeded up as the number of depth values that have to be considered possibly true by the estimation becomes much smaller, thereby reducing the processing time of the entire decoder.

The other features are the above skip flag, which reduces the number of pixels for which the depth has to be estimated, and depth estimation parameters, including regularization parameters that would help the depth estimator properly estimate smooth gradients of depth. The latter parameters highly depend on the used estimator. Therefore, to provide the most overall conclusions of our research, these parameters are not considered further.

The size of blocks is usually not fixed. For example, the blocks can be divided into four square subblocks [12] (Figure 2) or two rectangular subblocks (earlier proposed and evaluated by Szydełko and others [20]). All of these division types are shown in Figure 3.

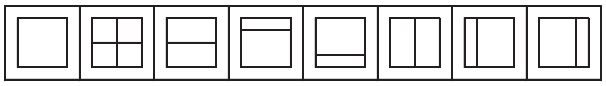

**FIGURE 3** Eight types of block splitting (from the left): no split, quad split, and six rectangular splits (three vertical and three horizontal): symmetrical, asymmetrical with a ratio of 25/75, and asymmetrical with a ratio of 75/25

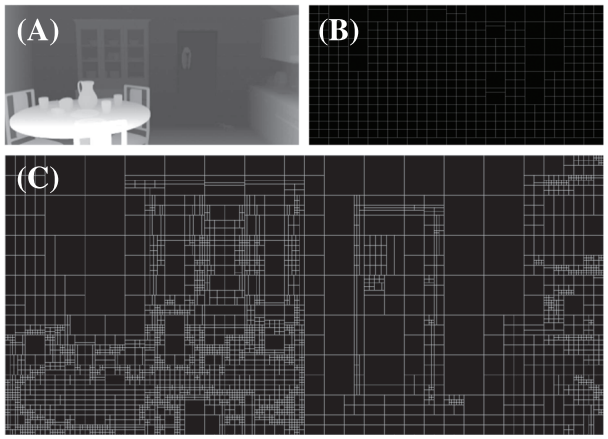


**FIGURE 4** (A) Depth map for one view of sequence kitchen, (B) block grid obtained using a state-of-the-art approach (max block size: 128, min block size: 64), and (C) block grid obtained using a recursive approach (max block size: 128, min block size: 8)

## 3 | PROPOSED BLOCK SPLITTING

State-of-the-art approaches [12,21] split the block only once. Therefore, they cannot fit the more complicated structure of the depth map. To address this, we propose the recursive solution, where each block can be recursively split several times to fit its edges to the edges in the depth map (Figure 4).

The motivation of fitting the block grid to the objects' edges is simple; the depth estimator does not check all possible disparity values but only the values from the range [$\mathrm{dMin}_B$, $\mathrm{dMax}_B$] (Figure 1). Therefore, if the block contains only pixels representing a single object (eg., a fragment of the wall), the range of checked disparities for each pixel within that block is very narrow, resulting in faster depth estimation and a lower possibility for estimating wrong depth. In the optimal case, if $\mathrm{dMin_B}$ and $\mathrm{dMax_B}$ for block $B$ are equal (thus, the block contains a single object of constant depth), there is no need for depth estimation for that block.

Before proceeding with recursive splitting, the entire frame undergoes an initial split into a grid of square blocks of the selected initial size. This allows using simple splitting in a quad-tree (or binary-tree) fashion. For further processing, finding the depth range represented by each block is required. The distance between the farthest and nearest objects in the block, along with the defined split threshold, determines the condition for subsequent block splits:

$$\text{if } (\mathrm{dMin}_B - \mathrm{dMax}_B > splitThresh) : \text{split block } B, \tag{1}$$

where $\mathrm{dMax}_B$ and $\mathrm{dMin}_B$ are the maximum and minimum values of the normalized disparity within block $B$, respectively, directly derived from $\mathrm{zNear}_B$ and $\mathrm{zFar}_B$ values [22] within block $B$:

$$\mathrm{dMin}_B = \frac{\frac{1}{\mathrm{zFar}_B} - \frac{1}{\mathrm{zFar}}}{\frac{1}{\mathrm{zNear}} - \frac{1}{\mathrm{zFar}}} \cdot \left(2^b - 1\right), \tag{2}$$

$$\mathrm{dMax}_B = \frac{\frac{1}{\mathrm{zNear}_B} - \frac{1}{\mathrm{zFar}}}{\frac{1}{\mathrm{zNear}} - \frac{1}{\mathrm{zFar}}} \cdot \left(2^b - 1\right). \tag{3}$$

Here, zNear and zFar are the global values of the nearest and farthest distances for the entire depth map, and $b$ is the bit depth of the depth map (typically equals to 10 or 16).

In other words, if the range of depth within the block is very large, the block should be narrowed down to smaller blocks. As mentioned in the previous section, the current approach allows splitting blocks into seven ways (Figure 3). Basic splitting into quads (four subblocks) does not require additional processing. However, it is necessary to select the best possible variant when rectangular splitting is enabled. For all possible variants (except quads), each subblock is checked using (1). If the depth range in each subblock of a single variant exceeds the set threshold, the tested variant is rejected. This is because it may increase the number of splits, thereby increasing metadata bitrate. For each variant (including quads), the sum of the cost volumes for all subblocks is calculated as follows:

$$\mathrm{costVol}_B = \sum_{S \in B} \left( \left( \mathrm{dMax}_S - \mathrm{dMin}_S \right) + 1 \right) \cdot w_S \cdot h_S, \tag{4}$$

with width $w_S$ and height $h_S$ of each subblock $S$ of block $B$. The variant with the lowest cost volume was assumed as the best way to split the block; thus, it was applied.

The entire above process is repeated recursively for the blocks created in previous splits (Figure 5). Finally, recursion ends when a block does not meet criterion (1), or one of the subblock dimensions becomes smaller than the set minimum block size.

The final step is to exploit the temporal consistency information from the depth maps. For this, we compare the corresponding blocks from the current and previous

frames. The similarity of the blocks is determined by the similarity of the depth range and the ratio of the average SAD (per pixel) to the maximum possible depth value. If the similarity value exceeds the set threshold (*skipThresh*), a skip flag is sent for that depth block. Therefore, to reduce the metadata bitrate, a skip flag is sent for the parent block (subblocks are also skipped). Additionally, dMin and dMax values can be quantized with the provided width of quantization levels (*quantWidth*).

## 4 | EXPERIMENTS

### 4.1 | Methodology

All 18 configurations of block splitting (Table 1) were evaluated under common test conditions (CTC) for MIV [23] on a test set containing 15 miscellaneous test sequences, including 9 computer-generated (CG) sequences (Figure 6) and 6 sequences with natural content (NC) captured by real multicamera systems (Figure 7). The sequences differ in scene characteristics, camera arrangement (linear, planar, and spherical), and camera type: 5 omnidirectional sequences with views represented in equirectangular projection (ERP) and 10 sequences with typical perspective views.

As the name indicates, the feature-driven DSDE approach requires the depth estimation process to be conducted at the decoder side. Therefore, all tested configurations were compared against the DSDE anchor in CTC called "G17 anchor." The G17 anchor was generated using the GA profile of the MIV standard [9] and the test model for MIV 8 (TMIV 8) [24]. When using the GA profile, the MIV encoder selects a subset of input views to decrease the total bitrate and pixel rate [25] of the transmitted video. Then, the selected views are concatenated to decrease the number of video streams and encoded using VVC (or another video encoder, as the MIV is codec-agnostic). The input views are restored from

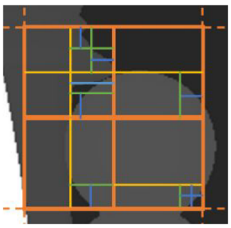

FIGURE 5 Idea of recursive block splitting. Orange: first split ($128 \times 128 \geq 64 \times 64$) as in Clare et al. [21]; additional recursive splits: second (yellow), third (green), and fourth (blue)

FIGURE 6 Computer-generated sequences. Left column: ClassroomVideo, Hijack, Kitchen, Fan, Group, and Mirror; right: Museum, Chess, and ChessPieces

TABLE 1 Tested configurations

| Block split types | Max/min block size | | |
|---|---|---|---|
| Quad only | 128/64[a] | 128/16 | 128/8 |
| | 64/32[a] | 64/16 | 64/8 |
| Quad + symmetrical rectangle | 128/64 | 128/16 | 128/8 |
| | 64/32 | 64/16 | 64/8 |
| Quad + symmetrical & asymmetrical rectangle | 128/64[b] | 128/16 | 128/8 |
| | 64/32[b] | 64/16 | 64/8 |

[a]State-of-the-art approach described in Garus et al. [12] and Clare et al. [27].
[b]State-of-the-art approach [21] adapted by the ISO/IEC WG4 MPEG VC group for the MPEG Immersive video SEI message (Annex F.2.6 of [28]).

FIGURE 7 Natural sequences. Left column: Frog, Hall, and Painter; right: Fencing, Carpark, and Street

TABLE 2 Used QP values for five rate points

| Test sequence | R1 | R2 | R3 | R4 | R5 |
|---|---|---|---|---|---|
| ClassroomVideo | 29 | 31 | 36 | 43 | 50 |
| Museum | 39 | 46 | 49 | 50 | 51 |
| Hijack | 18 | 21 | 26 | 31 | 35 |
| Kitchen | 26 | 31 | 35 | 39 | 43 |
| Chess | 23 | 28 | 32 | 37 | 42 |
| Painter | 23 | 28 | 34 | 37 | 41 |
| Frog | 29 | 31 | 34 | 39 | 44 |
| Carpark | 21 | 23 | 28 | 32 | 35 |
| Fencing | 21 | 24 | 26 | 29 | 34 |
| Hall | 12 | 15 | 17 | 20 | 25 |
| Street | 21 | 23 | 25 | 30 | 35 |
| Fan | 29 | 30 | 33 | 39 | 46 |
| Group | 28 | 32 | 36 | 40 | 46 |
| ChessPieces | 22 | 27 | 33 | 39 | 45 |
| Mirror | 25 | 31 | 36 | 42 | 48 |

decoded VVC bitstreams at the decoder side. Then, they are used for depth estimation. This step is performed using Immersive Video Depth Estimation (IVDE) [26], the publicly available depth estimation method that became the MPEG reference software and that is commonly used in MIV-related experiments.

According to the MIV CTC [23], for each sequence, the video streams were encoded using VVC with five QP values to obtain the rate-distortion (RD) curve, presenting the dependency between bitrate and quality. The exact values of QP used for each sequence are available in the MIV CTC document [23]. The QP values were selected by the ISO/IEC WG4 MPEG VC group to represent the useful range of bitrates, which can be used in practical commercial systems: from 5 to 50 Mbps. For the convenience of the readers, the used QP values are presented in Table 2.

As presented in Table 1, the influence of four parameters was evaluated in the experiments: max block size, min block size, rectangular block split enabling, and enabling of asymmetrical splitting. Other parameters of the feature-driven DSDE were common for all configurations:

- $skipThresh = 2\%$,
- $splitThresh = 2562$,
- $quantWidth = 256$.

The two main purposes of the feature-driven DSDE are to increase coding efficiency and to decrease decoding time (including the time required for depth estimation). Therefore, for the evaluation of the tested configurations, the following three parameters were considered:

- quality of synthesized views (estimated by PSNR metric by comparing input and colocated views synthesized using the decoded bitstream),
- bitrate (the total bitrate of all video streams, MIV metadata, and encoder-derived features),
- decoding time (calculated as a percentage of time needed for the decoding of the G17 anchor).

The configurations are denoted as follows: $TYPE - GS - RL$, where $GS$ is the initial grid size, $RL$ is the number of recursion levels, and $TYPE$ is one of three considered block split types: $Q$ is only quad splitting, $S$ is quad + symmetrical rectangular splitting, and $A$ is for square + all rectangular splitting. For example, Q-128-3 means the configuration with no rectangular splitting, initial grid 128 × 128, and three recursion levels. Thus, the minimum size of the block is equal to 16 × 16 pixels.

## 4.2 | Experimental results

The experimental results are divided into four subsections. In Section 4.2.1, the comparison of all tested configurations is presented. Sections 4.2.2 to 4.2.4 present the influence of single tested parameters: maximum block size (Section 4.2.2), minimum block size (Section 4.2.3), and the allowance of rectangular splitting (Section 4.2.4).

### 4.2.1 | Full results

Three figures in this subsection contain scatterplots presenting the dependencies between three considered

parameters: decoding time, bitrate, and quality. Decoding time and bitrate were averaged over all 15 sequences. Quality was averaged over all sequences and views. Moreover, as the decoding time and quality for each sequence were measured five times (for five different QP values), these parameters were averaged over all five QPs. The bitrates presented in this subsection do not include the bitrate of video streams. However, they include only the bitrate of the features as this bitrate is independent of QP.

As shown in Figure 8, configurations with more recursion levels (a smaller minimum size of the block; that is, the min block size is equal to 8) require a much higher bitrate than configurations with bigger blocks do. Such a relationship is expected because for configurations with smaller blocks, the number of blocks is large (compare Figures 4B,C), and for every block $B$, two features ($dMin_B$ and $dMax_B$), and several flags have to be sent in metadata. However, smaller blocks allow obtaining better quality of synthesized views. This is because they better adapt to the edges within depth maps (cf. Figure 4), simplifying the process of depth estimation.

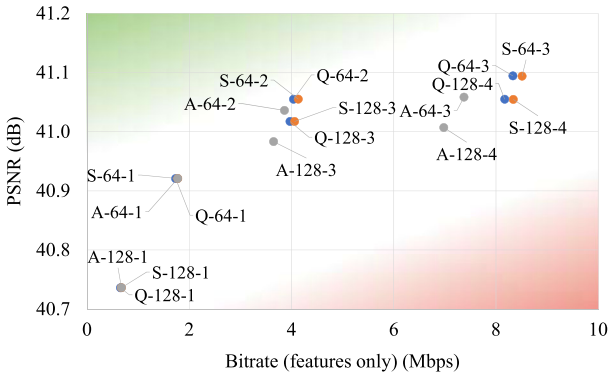


**FIGURE 8** Bitrate of extracted features (averaged over all sequences and views) versus PSNR of synthesized views (averaged over all sequences, views, and QPs) for 18 tested configurations; state-of-the-art approaches: Q-128-1 and Q-64-1 [12] and A-128-1 and A-64-1 [21]

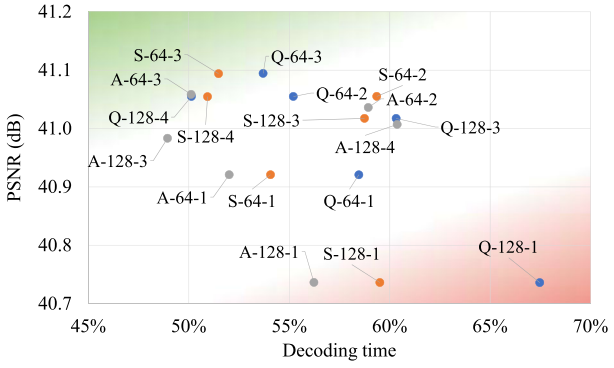


**FIGURE 9** Decoding time versus PSNR of synthesized views (both values are averaged over all sequences, views, and QPs) for 18 tested configurations; state-of-the-art approaches: Q-128-1 and Q-64-1 [12] and A-128-1 and A-64-1 [21]

When analyzing the dependency between quality and decoding time (Figure 9), recursive configurations with smaller min block sizes perform better than nonrecursive ones. As mentioned earlier, using smaller blocks allows the feature extractor to better adapt to the edges of the objects in the depth maps as the majority of the blocks contain pixels from a single object, and the difference between the nearest and farthest depths within many blocks is small. In this case, for a large part of the scene, the depth estimator does not need to analyze a high number of depth levels (being constrained to a narrow range between $dMin_B$ and $dMax_B$), which decreases the computational time required for depth estimation.

Figure 10 shows the dependency between bitrate and decoding time. As shown in the figure, additional split types (symmetrical and asymmetrical rectangular) also decrease the computational time while preserving similar decoding time. Such a difference can be spotted for all block sizes, but it is more visible for the nonrecursive scenario. This is because in nonrecursive configurations, the grid consists of only square blocks that do not fit the edges within depth maps. Meanwhile, edge adaptation is better when rectangular splitting is allowed. In recursive configurations, each block can be split many times, so even quad (square) splitting allows the feature extractor to fit into the depth edges.

### 4.2.2 | Influence of the initial grid size

The results presented in the previous subsection were averaged over all QPs to obtain a single PSNR value for

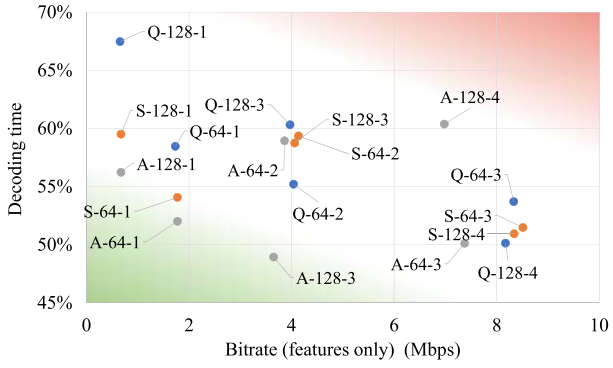


**FIGURE 10** Bitrate of extracted features (averaged over all sequences and views) versus decoding time (averaged over all sequences, views, and QPs) for 18 tested configurations; state-of-the-art approaches: Q-128-1 and Q-64-1 [12] and A-128-1 and A-64-1 [21]

each tested configuration. However, as mentioned in Section 4.1, VVC encoding was repeated five times with different values of QP to allow the analysis of the RD curves.

Figure 11 shows the RD curves obtained for two nonrecursive configurations, with only quad (square) split allowed. The orange curve represents the results obtained for a smaller initial grid size (64 × 64), whereas the blue curve represents the grid size of 128 × 128 pixels. The bitrates shown in Figure 11 are much higher than the bitrates in Figures 8 and 10. This is because the total bitrate includes the features, video streams, and MIV metadata.

As shown in Figure 11, both configurations can be efficiently used in practical immersive video systems, but each configuration should have different applications. Bigger blocks require fewer data to be sent. Thus, they can be used in low-bitrate systems. However, smaller blocks increase the quality of depth maps (Figure 12) and synthesized views (Figure 13), which compensates for the increased bitstream for high-bitrate systems. Moreover, as presented in Table 3, usage of a smaller initial grid size significantly decreases the time of depth estimation and the entire decoding process.

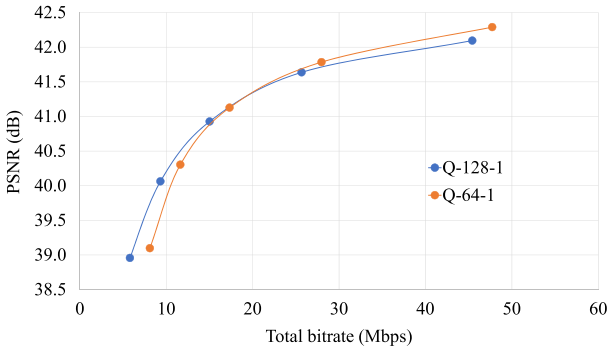


**FIGURE 11** RD curves obtained for different initial sizes of the block grid: 128 × 128 (blue) and 64 × 64 (orange); no recursion, no rectangular splitting; results averaged over all test sequences; both curves present the results of the state-of-the-art approach

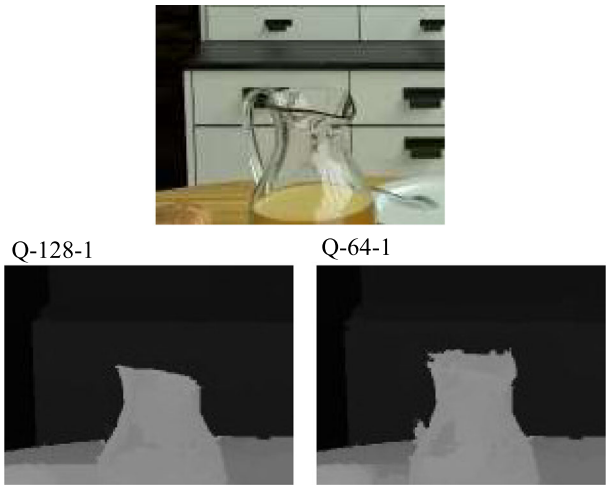


**FIGURE 12** Comparison of tested configurations; fragment of the estimated depth map (fragment colocated to the fragment of the input view presented at the top); sequence Kitchen

### 4.2.3 | Minimum block size

The second considered parameter was the minimum size of the block (or the number of recursion levels). Therefore, to investigate the influence of this parameter on coding efficiency, three configurations were compared. Only quad (square) splitting was allowed, and the initial grid size was set to 64 × 64 pixels. One of the tested configurations was nonrecursive (with the minimum block size equal to 32 × 32), and two were recursive with block sizes of 16 × 16 and 8 × 8, respectively. Figure 14 shows that the relationship between


**FIGURE 13** Comparison of tested configurations; fragment of the virtual view synthesized based on calculated depth maps (fragment colocated to the fragment of the input view presented at the top); please note that the ghosting artifacts at the view are presented at the left

**TABLE 3** Comparison of decoding times for two configurations with different initial grid sizes

| | Configuration | |
|---|---|---|
| **Video type** | **Q-128-1** | **Q-64-1** |
| ERP | 75% | 64% |
| Perspective | 64% | 55% |
| CG | 63% | 56% |
| NC | 74% | 62% |
| Average | 67% | 58% |

*Note*: Presented as a percentage of the decoding time of the G17 anchor [23].

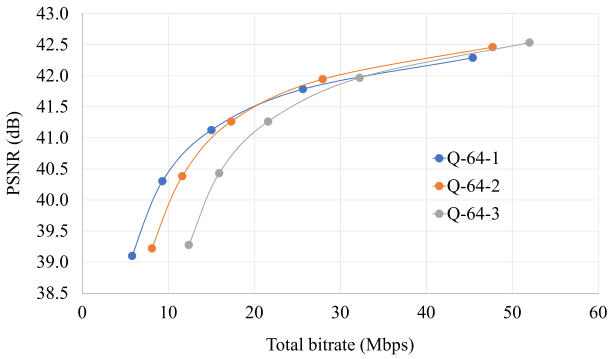


**FIGURE 14** F RD curves obtained for different minimum sizes of the block: 32 × 32 (one splitting level, blue line), 16 × 16 (two splitting levels, orange line), and 8 × 8 (three splitting levels, gray line); initial grid size: 64 × 64, no rectangular splitting; results averaged over all test sequences; Q-64-1 is the state-of-the-art approach [12]

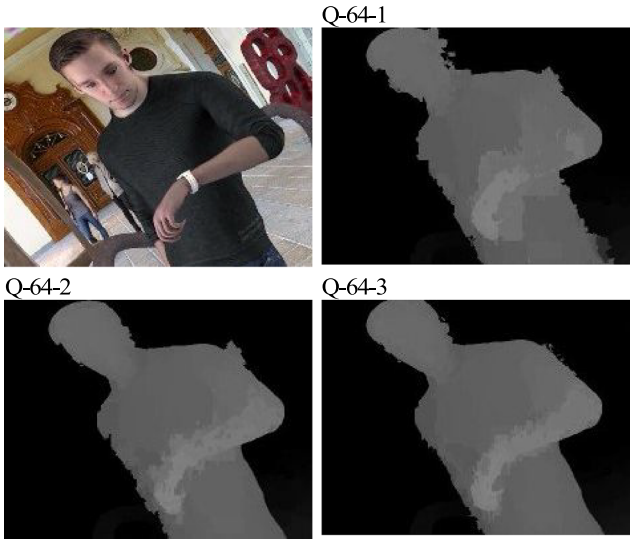


**FIGURE 15** Comparison of tested configurations; fragment of the estimated depth map (fragment colocated to the fragment of the input view presented at the top left); sequence Museum

coding efficiency and minimum block size is similar to the relationship for the initial grid size. Using a smaller number of blocks (no recursion, min block size: 32 × 32), the total bitrate is smaller, making it the best choice for low-bitrate systems. When higher bitrates are allowed, more recursion levels (and smaller blocks) become more effective because they allow obtaining better synthesis quality.

On average, the configuration Q-64-2 outperforms Q-64-1 when more than 20 Mbits per second are used. A configuration with a min block size of 8 × 8 (Q-64-3, three recursion levels) requires higher bitrates to outperform two configurations with fewer recursion levels.

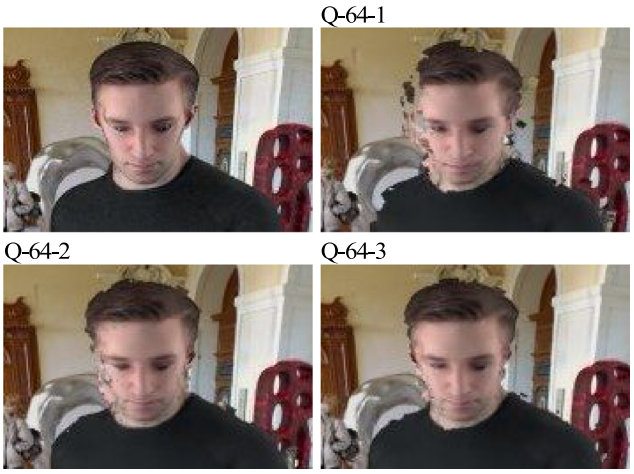


**FIGURE 16** Comparison of tested configurations; fragment of the virtual view synthesized based on calculated depth maps (fragment colocated to the fragment of the input view presented at the top left)

**TABLE 4** Comparison of decoding times for three configurations with different minimum block sizes

| | Configuration | | |
|---|---|---|---|
| **Video type** | **Q-64-1** | **Q-64-2** | **Q-64-3** |
| ERP | 64% | 64% | 64% |
| Perspective | 55% | 51% | 49% |
| CG | 56% | 54% | 54% |
| NC | 62% | 56% | 53% |
| Average | 58% | 55% | 54% |

*Note*: Presented as a percentage of the decoding time of the G17 anchor [23].

The use of more recursion levels increases the objective and subjective qualities of depth maps and synthesized virtual views (Figures 15 and 16).

Table 4 shows that smaller (better-suited) blocks decrease the decoding time, especially for NC. The reason for the discrepancy between CG and NC is simple. For CG, the depth maps are more stable in time. By contrast, the depth maps of NC are algorithmically estimated using input views; thus, they can contain temporal instabilities. When the depth map is temporally stable, many blocks in consecutive frames are skipped (*skip_flag* = 1), and depth estimation is not performed for them. Therefore, the size of the blocks does not significantly affect the decoding time for frames other than the first one. For inconsistent depth maps, depth estimation is performed for the higher number of blocks in all frames; thus, the usage of smaller blocks decreases the decoding time even more.

### 4.2.4 | Allowance of rectangular splitting

In the third experiment, the dependency between coding efficiency and allowance of rectangular splitting was examined. Figure 17 compares the three configurations: only quad splitting ("Q," blue line), quad + symmetrical rectangular splitting ("S," orange line), and all possible split types ("A," gray line). All tested configurations shared a common initial grid size (64 × 64) and the number of split levels (3).

In terms of RD curves, "A" configuration with all possible splitting types outperforms other configurations by significantly decreasing the bitrate required for sending features. Both configurations with no asymmetrical splitting ("Q" and "S") have similar performance, and the results are almost indistinguishable. The objective and subjective qualities are similar for all tested configurations, for depth maps and synthesized virtual views (Figures 18 and 19).

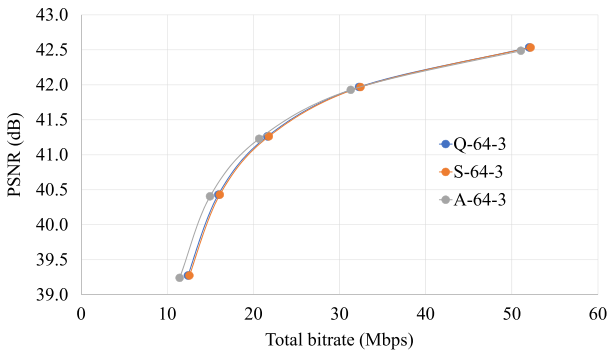


**FIGURE 17** RD curves obtained for different split types: only quad (blue), quad + symmetrical rectangular (orange), and all split types (gray); initial grid size: 64 × 64, number of split levels: 3; results averaged over all test sequences


**FIGURE 18** Comparison of tested configurations; fragment of calculated depth map (fragment colocated to the fragment of the input view presented at the top); sequence Carpark

However, even if the allowance of the symmetrical rectangular splitting does not improve coding efficiency in terms of bitrate and quality, it decreases the computational time of the decoding process (on average by 3 percentage points, as shown in Table 5). The addition of asymmetrical rectangular splitting further decreases the decoding time, but the gain is less noticeable.

Therefore, to illustrate the relationship between coding efficiency and splitting type, all three splitting configurations were compared for the recursive and nonrecursive scenarios, with one and three split levels, respectively. As previously stated, three coding efficiency parameters were examined: quality of synthesized views (Figure 20), bitrate (Figure 21), and decoding time (Figure 22). All presented results were averaged over all tested QP values.


**FIGURE 19** Comparison of tested configurations; fragment of the virtual view synthesized based on calculated depth maps (fragment colocated to the fragment of the input view presented at the top left)

**TABLE 5** Comparison of decoding times for three configurations with different splitting types

| | Configuration | | |
|---|---|---|---|
| **Video type** | **Q-64-3** | **S-64-3** | **A-64-3** |
| ERP | 64% | 62% | 60% |
| Perspective | 49% | 46% | 45% |
| CG | 54% | 52% | 52% |
| NC | 53% | 50% | 47% |
| Average | 54% | 51% | 50% |

*Note*: Presented as a percentage of the decoding time of the G17 anchor [23].

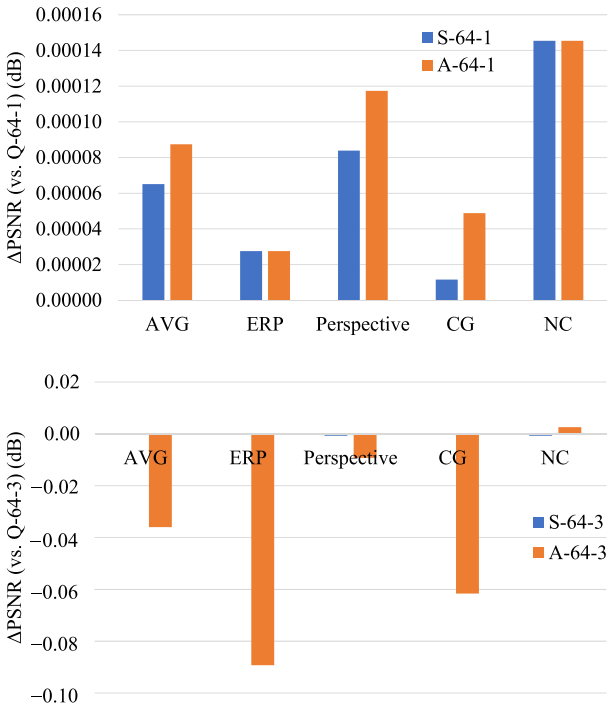


FIGURE 20 Quality gain for different splitting types (vs. quad split only); in a nonrecursive scenario (top) and recursive scenario (bottom); results averaged over all sequences (AVG), all omnidirectional sequences (ERP), all perspective sequences (Perspective), all computer-generated sequences (CG), and all natural sequences (NC); A-64-1 is the state-of-the-art-approach [21]

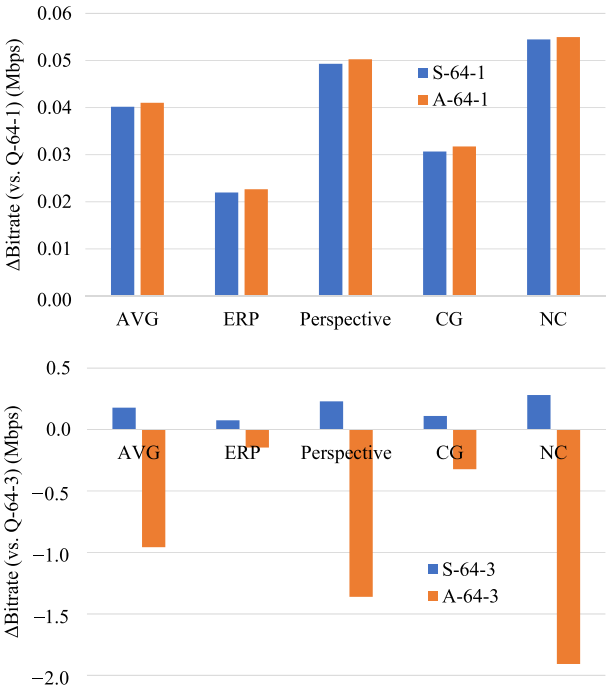


FIGURE 21 Bitrate change for different splitting types (vs. quad split only); in a nonrecursive scenario (top) and recursive scenario (bottom); results averaged over all sequences (AVG), all omnidirectional sequences (ERP), all perspective sequences (Perspective), all computer-generated sequences (CG), and all natural sequences (NC); A-64-1 is the state-of-the-art-approach [21]

Figure 20 shows that the gain of using rectangular splitting highly depends on the number of split levels. For the nonrecursive scenario, both rectangular split types increase the quality of synthesized views, but the difference is negligible.

In the recursive approach, the "S" configuration has a similar performance to configuration "Q" with quad splitting only. The configuration with all split types decreases the synthesis quality because of the minimum size of the block. In configurations "S" and "Q," the minimum width or height of the block, which was split three times, is 8 pixels ($8 = 64 \times ½ \times ½ \times ½$). In the asymmetrical approach, the edge of the block may be shorter, down to 1 pixel, if the smallest of two sub-blocks is recursively asymmetrically split three times ($1 = 64 \times ¼ \times ¼ \times ¼$). In such a case, the shorter edge of the block is smaller than the typical size of super-pixels [29] used in the depth estimator (ie., in IVDE software [26]), which may cause the wrong estimation of depth for these areas.

However, even the highest loss of the quality may be negligible because it is, on average, smaller than 0.1 dB (Figure 20).

The influence of the splitting type on bitrate also depends on the number of recursion levels (Figure 21). When no recursion is used and when each block can be split only once, usage of rectangular splitting slightly increases the bitrate (compared to quad split only). This is because an increase in metadata is needed to signal the splitting type.

In the recursive scenario, the symmetrical rectangular splits slightly increase the bitrate for the same reasons as in the nonrecursive one. However, the influence of asymmetrical splitting is opposite, and it significantly decreases the bitrate. This discrepancy was expected as asymmetrical rectangular splitting fits into the edges in the depth map faster than both symmetrical splitting types (Figure 23). Therefore, fewer splits are needed to fit into the depth edges on average. Thus, fewer blocks are used, and fewer features have to be sent.

The last considered parameter was the decoding time. For this parameter, the results are similar to recursive and nonrecursive scenarios (Figure 22). Generally, rectangular splitting allows the additional decrease of the computational time of the decoder compared to quad splitting only.

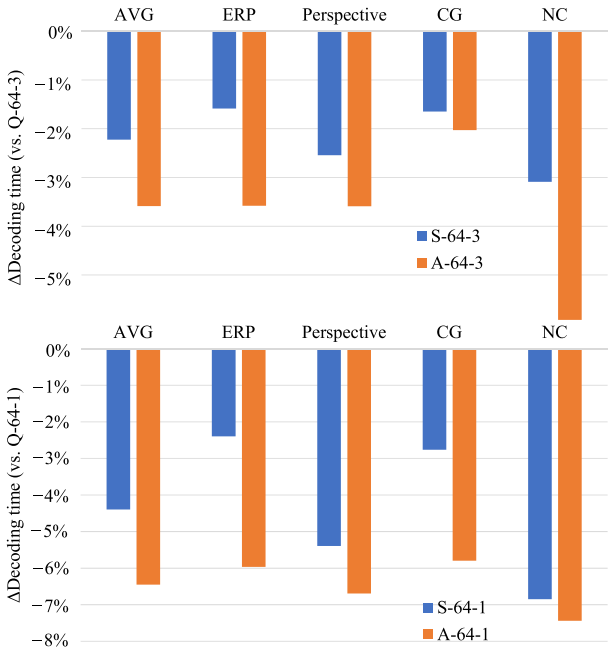


**FIGURE 22** Decoding time change for different splitting types (vs. quad split only); in a nonrecursive scenario (top) and recursive scenario (bottom); results averaged over all sequences (AVG), all omnidirectional sequences (ERP), all perspective sequences (Perspective), all computer-generated sequences (CG), and all natural sequences (NC); A-64-1 is the state-of-the-art-approach [21]

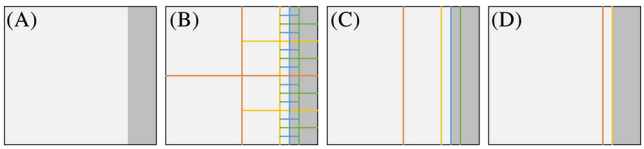


**FIGURE 23** Matching of different splitting types to the vertical edge in the depth map; (A) fragment of the depth map (64 × 64), (B) quad split (four split levels required), (C) symmetrical rectangular split (four split levels required), and (D) asymmetrical rectangular split (two split levels required)

## 5 | CONCLUSIONS

The paper proposes a new approach for deriving depth map features that enhances the performance of decoder-side depth estimation. In the state-of-the-art method, depth features (ie., the nearest and farthest depth values and the flag describing the temporal stability of depth) are derived for blocks of a fixed size (eg., 64 × 64 pixels), which can be split only once to better fit the edges within the depth map.

In the proposed approach, splitting is not constrained, and the block grid can be better adapted to the edges due to the allowance of further splitting of already split blocks. Such an approach requires sending more features, thus increasing the total bitrate. However, it significantly decreases the computational time of the decoder and further increases the quality of synthesized virtual views.

Therefore, comprehensive experiments were performed to present the advantages of the proposed approach. In the experiments, 18 recursive and nonrecursive configurations were evaluated to investigate the influence of the configuration of feature extraction on coding efficiency, including three crucial parameters: quality, total bitrate, and decoding time. These experiments are the first comprehensive test of the GA profile from the incoming MIV in the feature-driven configuration.

The experimental results showed that various use cases require different approaches. For low-bitrate immersive video systems, where the bitrate has to be optimized, the nonrecursive approach performed better because of the smaller feature metadata bitstream. However, in practical systems, where the decoding process has to be performed in real time and the quality of synthesized views has to be as high as possible, the proposed recursive block splitting outperformed the state-of-the-art approach.

## ACKNOWLEDGEMENT

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2018-0-00207, Immersive Media Research Laboratory).

## CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

## ORCID

*Adrian Dziembowski* https://orcid.org/0000-0001-7426-3362

## REFERENCES

1. M. Domański, O. Stankiewicz, K. Wegner, and T. Grajek, *Immersive visual media—MPEG-I: 360 video, virtual navigation and beyond*, in Proc. Int. Conf. Syst., Signals Image Process. (Poznan, Poland), July 2017, pp. 1–9.
2. A. Doumanoglou, D. Griffin, J. Serrano, N. Zioulis, T. K. Phan, D. Jiménez, D. Zarpalas, F. Alvarez, M. Rio, and P. Daras, *Quality of experience for 3-D immersive media streaming*, IEEE Tr. Broadcast **64** (2018), no. 2, 379–391.
3. M. Tanimoto, *FTV (free-viewpoint TV)*, in Proc. IEEE Int. Conf. Image Process. (Hong Kong), Sept. 2010, pp. 2393–2396.
4. A. Schenkel, D. Bonatto, S. Fachada, H. L. Guillaume, and G. Lafruit, *Natural scenes datasets for exploration in 6DoF navigation*, in Proc. Int. Conf. 3D Immersion (Brussels, Belgium) Dec. 2018, pp. 1–8.
5. O. Stankiewicz, M. Domański, A. Dziembowski, A. Grzelka, D. Mieloch, and J. Samelak, *A free-viewpoint television system*

*for horizontal virtual navigation*, IEEE T. Mult. **20** (2018), no. 8, 2182–2195.
6. P. Goorts, M. Dumont, S. Rogmans, and P. Bekaert, *An end-to-end system for free viewpoint video for smooth camera transitions*, in Proc. Int. Conf. 3D Imaging (Liege, Belgium), Dec. 2012, pp. 1–7.
7. D. Mieloch, O. Stankiewicz, and M. Domański, *Depth map estimation for free-viewpoint television and virtual navigation*, IEEE Access **8** (2020), 5760–5776.
8. G. Lafruit, D. Bonatto, C. Tulvan, M. Preda, and L. Yu, *Understanding MPEG-I coding standardization in immersive VR/AR applications*, SMPTE Motion Imaging J. **128** (2019), no. 10, 33–39.
9. J. M. Boyce, R. Doré, A. Dziembowski, J. Fleureau, J. Jung, B. Kroon, B. Salahieh, V. K. Vadakital, and L. Yu, *MPEG immersive video coding standard*, Proc. IEEE **109** (2021), no. 9, 1521–1536.
10. B. Salahieh, and J. Boyce, *MIV geometry absent*, ISO/IEC JTC1/SC29/WG4 MPEG2020/M54874, Online. 2020.
11. K. Müller, P. Merkle, and T. Wiegand, *3-D video representation using depth maps*, Proc. IEEE **99** (2011), no. 4, 643–656.
12. P. Garus, F. Henry, J. Jung, T. Maugey, and C. Guillemot, *Immersive video coding: should geometry information be transmitted as depth maps?* IEEE Trans. Circuits Syst. Video Technol. (2021). https://doi.org/10.1109/TCSVT.2021.3100006
13. P. Garus, J. Jung, T. Maugey, and C. Guillemot, Bypassing depth maps transmission for immersive video coding, in Proc. 2019 Picture Coding Symp. (Ningbo, China), 2019. https://doi.org/10.1109/PCS48520.2019.8954543
14. H. Laga, L. V. Jospin, F. Boussaid, and M. Bennamoun, *A survey on deep learning techniques for stereo-based depth estimation*, IEEE Trans. Pattern. Anal. Machine Intell. (2020). https://doi.org/10.1109/TPAMI.2020.3032602
15. A. Dziembowski, M. Domański, A. Grzelka, D. Mieloch, J. Stankowski, and K. Wegner, *The influence of a lossy compression on the quality of estimated depth maps*, in Proc. Int. Conf. Syst. Image Process. (Bratislava, Slovakia), 2016. https://doi.org/10.1109/IWSSIP.2016.7502730
16. D. Mieloch, D. Klóska, and M. Woźniak, *Point-to-block matching in depth estimation*, in Proc. Int. Conf. Central Eur. Comput. Graph., Visualization Computer Vision, 2021, pp. 153–144
17. X. He, Q. Liu, and Y. Yang, *MV-GNN: multi-view graph neural network for compression artifacts reduction*, IEEE Trans. Image Proc. **29** (2020), 6829–6840.
18. S. Chen, Q. Liu, and Y. Yang, *Adaptive multi-modality residual network for compression distorted multi-view depth video enhancement*, IEEE Access **8** (2020) 97072–97081.
19. D. Mieloch, A. Dziembowski, and M. Domański, *Depth map refinement for immersive video*, IEEE Access **9** (2021) 10778–10788.
20. B. Szydełko, D. Mieloch, A. Dziembowski, G. Lee, and J. Y. Jeong, Rectangular blocks in encoder-derived features for decoder-side depth estimation, ISO/IEC JTC1/SC29/WG4 MPEG2021/M56335, Online, 2021.
21. G. Clare, P. Garus, F. Henry, B. Szydełko, D. Mieloch, A. Dziembowski, M. Domański, G. Lee, and J. Y. Jeong, *[MIV] Combination of m56626 and m56335 for Geometry Assistance SEI message*, ISO/IEC JTC1/SC29/WG4 MPEG2021/M56950, Online, 2021.
22. O. Stankiewicz, G. Lafruit, and M. Domański, *Chapter 1 - Multiview video: Acquisition, processing, compression and virtual view rendering*, in Academic Press Library in Signal Processing, Academic Press, 2018, pp. 3–74. https://doi.org/10.1016/B978-0-12-811889-4.00001-4
23. *Common Test Conditions for MPEG Immersive Video*, ISO/IEC JTC1/SC29/WG4 MPEG2021/N0085, Online, 2021.
24. *Test Model 9 for MPEG Immersive Video*, ISO/IEC JTC1/SC29/WG4 MPEG2021/N0084, Online, 2021.
25. A. Hornbarg, *Handbook of Machine Vision*, Wiley, 2007, pp. 46–47. https://doi.org/10.1002/9783527610136
26. *Manual of IVDE 3.0*, ISO/IEC JTC1/SC29/WG4 MPEG2020/N0058, Online, 2021.
27. G. Clare, P. Garus, and F. Henry, *[MIV] Geometry Assistance SEI message*, ISO/IEC JTC1/SC29/WG4 MPEG2021/M56626, Online, 2021.
28. *Text of ISO/IEC FDIS 23090-12 MPEG Immersive Video*, ISO/IEC JTC1/SC29/WG4 MPEG2021/N0111, Online, 2021.
29. R. Achanta and S. Süsstrunk, *Superpixels and Polygons using simple non-iterative clustering*, in Proc. IEEE Conf. Comput. Vision Pattern Recogn. (Honolulu, HI, USA), July 2017, pp. 4895–4904. https://doi.org/10.1109/CVPR.2017.520

## AUTHOR BIOGRAPHIES

**Błażej Szydełko** was born in 1997. He received his BSc degree in ICT engineering in 2021. He is currently pursuing his MSc degree at Poznań University of Technology. He combines his studies with work at the Institute of Multimedia Telecommunications at PUT, where he contributes to the development of immersive media technologies within the ISO/IEC MPEG group.

**Adrian Dziembowski** was born in Poznań, Poland, in 1990. He received his MSc and PhD degrees from Poznań University of Technology (PUT), Poznań, Poland, in 2014 and 2018, respectively. Since 2019, he has been an assistant professor with the Institute of Multimedia Telecommunications at PUT. He authored and coauthored about 30 articles on various aspects of immersive video, free navigation, and free-viewpoint television systems. He is also actively involved in ISO/IEC MPEG activities toward MPEG Immersive video coding standard.

**Dawid Mieloch** received his MSc and PhD from Poznań University of Technology (PUT), Poznań, Poland, in 2014 and 2018, respectively. Currently, he is an assistant professor at the Institute of Multimedia Telecommunications at PUT. He is actively involved in ISO/IEC MPEG activities, where he contributes to the development of immersive media technologies. He has been involved in several multiview and 3D video processing projects. His professional interests include free-viewpoint television, depth estimation, and camera calibration.

**Marek Domański** received his MSc, PhD, and Habilitation degrees from Poznań University of Technology, Poland, in 1978, 1983, and 1990, respectively. Since 1993, he has been a professor at Poznań University of Technology, where he leads the Institute of Multimedia Telecommunications. He coauthored one of the very first AVC decoders for tv set-top boxes (2004) and highly ranked technology proposals to MPEG for scalable video compression (2004), 3D video coding (2011), and immersive video coding (2019). He authored three books and over 300 papers in journals and conference proceedings. The contributions were mostly on image, video, and audio compression; virtual navigation; free-viewpoint television; image processing; multimedia systems; 3D video and color image technology; digital filters; and multidimensional signal processing.

**Gwangsoon Lee** received his PhD degree in electronics engineering from Kyungpook National University, Daegu, South Korea, in 2004. He joined the Electronics and Telecommunications Research Institute, Daejeon, South Korea, in 2001. He is currently a principal researcher with Realistic-Media Research Section. His research interests include immersive video processing, light field imaging system, and three-dimensional video system.

**How to cite this article:** B. Szydełko, A. Dziembowski, D. Mieloch, M. Domański, and G. Lee, *Recursive block splitting in feature-driven decoder-side depth estimation*, ETRI Journal (2022), 1–13. https://doi.org/10.4218/etrij.2021-0308