INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG2020/m53407
April 2020, Online

Source	**Poznań University of Technology, Poznań, Poland** **Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea**
Status	**Input**
Title	**[MPEG-I Visual] Immersive video depth estimation**
Author	**Dawid Mieloch*, Adrian Dziembowski*, Jakub Stankowski*, Olgierd Stankiewicz*, Marek Domański*, Gwangsoon Lee****, **Yun Young Jeong**** * - Poznań University of Technology, **- Electronics and Telecommunications Research Institute
Contact	dawid.mieloch@put.poznan.pl

Abstract

Omnidirectional video formats are currently considered within MPEG in the context of 6DoF/3DoF+ video technology. Unfortunately, the current version of Depth Estimation Reference Software does not allow depth estimation from video acquired by multiple omnidirectional cameras, needed to create multi-point 6DoF/3DoF+ scene representation. In this document, we present novel depth estimation technique and software developed by Poznań University of Technology (PUT) and Electronics and Telecommunications Research Institute (ETRI), called Immersive Video Depth Estimation (IVDE), which addresses these deficiencies. Full source code of the method and CTC-based comparison with RDE are included in this contribution.

1 Introduction

One of the subjects considered in the current MPEG-I activities in the context of prospective 6DoF/3DoF+ video technology are omnidirectional video formats. Acquisition of video with an omnidirectional camera or with several omnidirectional cameras positioned in distinct locations seems to be a very promising way to capture real, natural 3D content. It is therefore important to develop tools allowing further research.

2 Overview of the method

The particular usefulness of the presented method in virtual navigation, free-viewpoint television and other 6DoF systems, is a result of the joint exploitation of the ideas mentioned below:

- Depth is estimated for segments instead of individual pixels, and thus the size of segments can be used to control the trade-off between the quality of depth maps and the processing time of estimation. Larger segments can be used to attain fast depth estimation, or finer segments can be used to attain higher quality.

- Estimation is performed for all views simultaneously and produces depths that are inter-view consistent because of the utilization of the new formulation of the cost function, developed for segment-based estimation.

- No assumptions about the positioning of views are stated: any number of arbitrarily positioned cameras (both perspective and omnidirectional) can be used during the estimation.

- In the proposed temporal consistency enhancement method, depth maps estimated in previous frames are utilized in the estimation of depth for the current frame, increasing the consistency of depth maps and simultaneously decreasing the processing time of estimation.

- The proposed depth estimation framework uses a novel parallelization method that significantly reduces the processing time of graph-based depth estimation.

The new framework does not use any external libraries for image processing operations. Such libraries offer a very wide spectrum of image processing solutions, but they have a negative impact on the easiness of their use in other frameworks. Moreover, OpenCV does not provide full compatibility between different versions of this library, which provides further difficulties during the development of new software.

2.1 Depth estimation

The estimation of depth in the proposed method is based on a cost function minimization. The cost function is based on two components: the intra-view discontinuity cost $V_{s,t}$ and the inter-view matching cost $M_{s,s'}$, responsible for the inter-view consistency of depth maps:

$$E(\underline{\mathrm{d}}) = \sum_{c \in \mathrm{C}} \sum_{s \in \mathrm{S}} \left\{ \sum_{c' \in \mathrm{D}} M_{s,s'}(d_s) + \sum_{t \in \mathrm{T}} V_{s,t}(d_s, d_t) \right\},$$

where:

$\underline{\mathrm{d}}$ – vector containing depth value for each segment in all views,
C – set of views,
c – view used in the estimation,
D – set of views neighboring to the view c,
c' – view neighboring to the view c,
S – set of segments of the view c,
s – segment in the view c,
d_s – currently considered depth of the segment s, $d_s \in \underline{\mathrm{d}}$,
s' – segment in the view c', which corresponds to the segment s in the view c for the currently considered depth d_s,

$M_{s,s'}$ – inter-view matching cost between segments s and s',
T – set of segments neighboring to the segment s,
t – segment neighboring to the segment s,
$V_{s,t}$ – intra-view discontinuity cost between segments s and t,
d_t – currently considered depth of the segment t, $d_s \in \underline{\mathrm{d}}$.

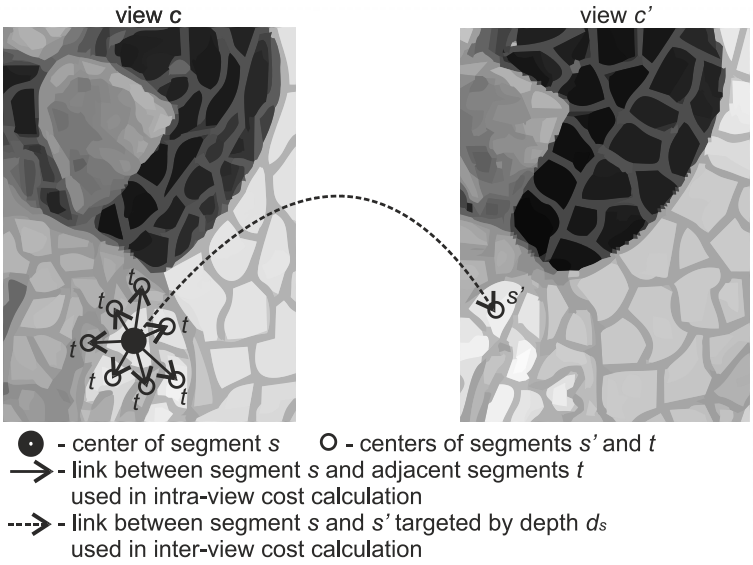


Fig. 1. Inter-view and intra-view costs.

The intra-view discontinuity cost is calculated between all neighboring segments within the same view:

$$V_{s,t}(d_s, d_t) = \beta \cdot |d_s - d_t| \ ,$$

where:
β – smoothing coefficient,
d_s – currently considered depth of the segment s,
s – segment in the view c,
t – segment neighboring to the segment s,
$V_{s,t}$ – intra-view discontinuity cost between segments s and t,
d_t – currently considered depth of the segment t.

In the proposed method the smoothing coefficient β is not fixed for all segments, instead, the smoothing coefficient is calculated using a similarity of two neighbouring segments *s* and *t* and β_0 that is an initial smoothing coefficient:

$$\beta = \beta_0 / \left\| [\hat{Y}\ \hat{C}_b\ \hat{C}_r]_s - [\hat{Y}\ \hat{C}_b\ \hat{C}_r]_t \right\|_1 \ ,$$

where:
β – smoothing coefficient ,
β_0 – initial smoothing coefficient provided by the user,
$\|\cdot\|_1$ – L1 distance,
s – segment in the view c,
t – segment neighbouring to the segment s,
$[\hat{Y}\ \hat{C}_b\ \hat{C}_r]_s$ – vector of average Y, C_b, C_r color components of the segment s,
$[\hat{Y}\ \hat{C}_b\ \hat{C}_r]_t$ – vector of average Y, C_b, C_r color components of the segment t.

The core of the inter-view matching cost, denoted as $m_{s,s'}$, is:

$$m_{s,s'}(d_s) = \frac{1}{count(W)} \sum_{w \in \mathrm{W}} \left\| [Y C_b C_r]_{\mu_s + w} - [Y C_b C_r]_{T[\mu_s]+w} \right\|_1 ,$$

where:

W – set of points in the window of the size specified by the user,
count(·) – size of the window W,
w – vector of coordinates of a point in the window W,
$\|\cdot\|_1$ – L1 distance,
μ_s – vector of coordinates of center of a segment s,
$T[\cdot]$ – 3D transform obtained from intrinsic and extrinsic parameters of cameras,
$[Y\ C_b\ C_r]_{\mu_s+w}$ – vector of Y, C_b, C_r color components of the center μ_s of the segment s,
$[Y\ C_b\ C_r]_{T[\mu_s]+w}$ – vector of Y, C_b, C_r color components of the point in a view c' corresponding to the center μ_s of the segment s in a view c.

In order to achieve the inter-view consistency of depth maps, the value of the inter-view matching cost $M_{s,s'}(d_s)$ is calculated as [6]:

$$M_{s,s'}(d_s) = \begin{cases} \min\{0, m_{s,s'}(d_s) - K\} & if\ d_s = d_{s'} \\ 0 & if\ d_s \neq d_{s'} \end{cases} ,$$

where:

s – segment in the view c,
d_s – currently considered depth of the segment s,
s' – segment in the view c', which corresponds to the segment s in the view c for the currently considered depth d_s ,
$d_{s'}$ – currently considered depth of the segment s',
$M_{s,s'}$ – inter-view matching cost between segments s and s',
$m_{s,s'}$ – core of the inter-view matching cost between segments s and s',
K – a positive constant.

The value of constant K is selected so that the inter-view matching cost $M_{s,s'}$ is not dominated by the intra-view discontinuity cost $V_{s,t}$, as a sum of these two costs constitutes the cost function of the depth optimisation. The chosen final value of K in presented research is 30, as it provides the high quality of estimated depth maps for all tested sequences. The use of both equirectangular and perspective views is included in the 3D transform $T[\cdot]$.

In order to increase the final quality of estimated depth maps, we propose a new segment-based method of the depth enhancement, named neighboring segments depth analysis.

The proposed process is performed for each segment in estimated depth maps. For the currently processed segment, depth values of its neighboring segments are tested as new depth candidates for this segment. A depth value is used if two conditions are fulfilled: use of this depth reduces the inter-view matching cost for the processed segment and a corresponding segment in neighboring view targeted by this depth also has the same value of depth.

The proposed solution increases the quality of depth maps in areas of uncertain depth (e.g., disoccluded areas) and preserves the inter-view consistency of depth maps. Moreover, because the process is performed after estimating the depth for each frame, such enhanced depth is used for all following frames (because of segmentation-based temporal enhancement). Therefore, such an approach increases the quality of depth maps also in terms of temporal consistency.

2.2 Temporal consistency enhancement

In natural video sequences, only a small part of an acquired scene considerably changes in consecutive frames, especially when cameras are not moving during the acquisition of video. The idea of the proposed temporal consistency enhancement of depth estimation is to calculate a new value of depth only for the segments that changed (in terms of their color) in comparison with the previous frame.

The proposed temporal consistency enhancement method allows us to automatically mark segments as unchanged in consecutive frames. These segments are used in the calculation of the intra-view discontinuity and the inter-view matching cost for other segments, but are not represented by any node in the structure of the optimized graph. It reduces the number of nodes in the graph, making the optimization process significantly faster, and on the other hand, increases the temporal consistency of estimated depth maps.

In the first frame of a depth map, denoted as an "I-type" depth frame, the estimation is performed for all segments, as described in the previous sections. The following frames ("P-type" depth frames) can utilize depth information from the preceding P-type depth frame and the I-type depth frame.

Segment s is marked by the algorithm as unchanged in two cases: if all components of the vector $[\hat{Y}\hat{C}_b\hat{C}_r]_s$ of average Y, Cb and Cr color components changed less than the set threshold T in comparison with segment s_B, which is a collocated segment in the previous P-type frame, or, if all components of the abovementioned vector changed less than the threshold T in comparison with segment s_I – a collocated segment in the I-type frame. If any of these two conditions are met, then segment s adopts the depth from the segment s_B or s_I (depending on which condition was fulfilled).

A collocated segment in the previous or the first frame is simply the segment which contains the central point of the segment s. Therefore, even if the segmentation in compared frames is not the same, the algorithm can easily find the corresponding segment in these frames.

The introduction of two reference depth frames has a beneficial impact on the visual quality of virtual navigation. First, the adoption of depth from the previous P-type depth frame allows us to use the depth of objects that changed their position over time. On the other hand, the adoption of depth from the I-type depth frame minimizes the flickering of depth in the background.

2.3 Parallelization of graph-based optimization

In our proposal, each of n threads estimates a depth map with an n-times lower number of depth levels. Depth maps with a reduced number of depth levels that were calculated by different threads have to be merged into one depth map. The merging process is performed in a similar way as depth estimation [using the cost function (1)], but only two levels of depth are considered for each segment – i.e., the depth of a segment from thread t or the depth from thread $t+1$ (Fig. 4). Only two depth maps can be merged into one by one thread during the merging cycle. Therefore, for n threads, $\lceil log2(n) \rceil$ of additional cycles are needed to estimate the final depth map with all depth levels.

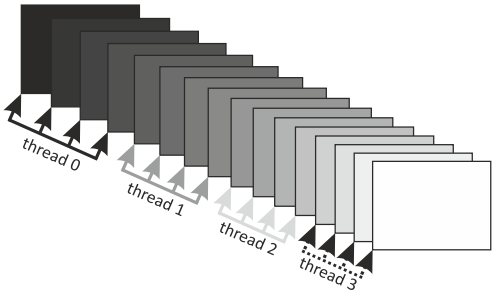


Fig. 2. Depth levels are divided into blocks, each rectangle represents a different level of the depth of a scene.

Of course, even without the use of parallelization, all cores of the CPU can also be used for depth estimation, e.g., each core can perform the estimation of depth for different sets of input views (e.g., for each 5 cameras

of the system), or for different frames of the sequence. Unfortunately, when many standalone depth estimation processes are performed, it results in the loss of inter-view consistency or temporal consistency of estimated depth maps. When the proposed parallelization is used, both inter-view and temporal consistency of depth maps, which are fundamental for the quality of virtual view synthesis, are preserved.

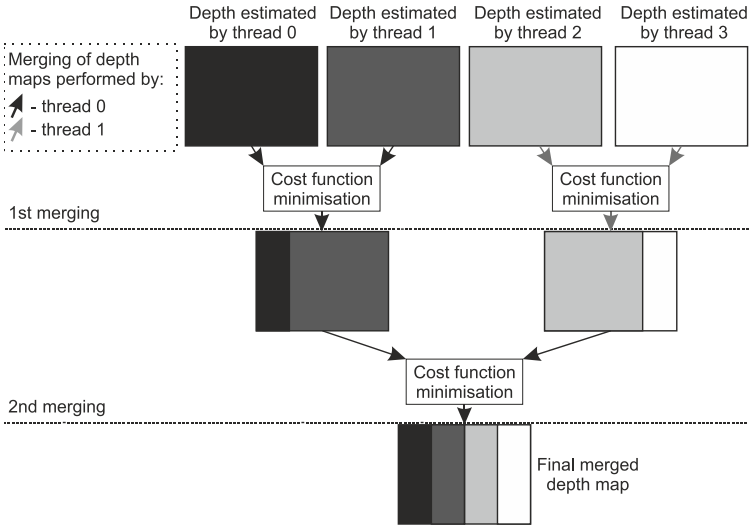


Fig. 3. Depth map merging process for the case of 4-thread parallelization.

2.4 Segmentation in omnidirectional videos

The use of omnidirectional cameras is taken into account during the superpixel segmentation of input views. The superpixel segmentation [2] is based on the calculation of the color and spatial distances of a point to neighboring superpixels.

Fig. 4. shows initial grid of 1000 superpixels used in the beginning of segmentation process. To estimate such initial grid, the overall size of image is divided by the number of superpixels in order to acquire the average size of superpixel. Then, the square root of the resulting superpixel size is used to define the distance between centers of superpixels and, in the end, the whole image is divided evenly as presented in the figure below.

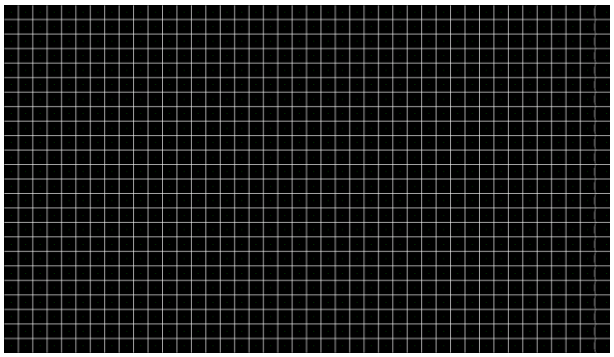


Fig. 4. Initial grid of superpixels used in segmentation.

In next steps, segments' shapes are changed on the basis of color and spatial distances of neighboring points in order to match edges present in a scene. The final segmentation of a omnidirectional sequence can be seen in Fig. 5.

Segments on the top and bottom border of presented image have similar size in the whole image. However, in equirectangular image, areas in the top and the bottom of an image represent much smaller areas of a scene than areas in the middle of an image. Therefore, if the segmentation of the image would be not adapted to the equirectangular images, then the accuracy of estimated depth maps would be not consistent in for the whole image in the proposed method.

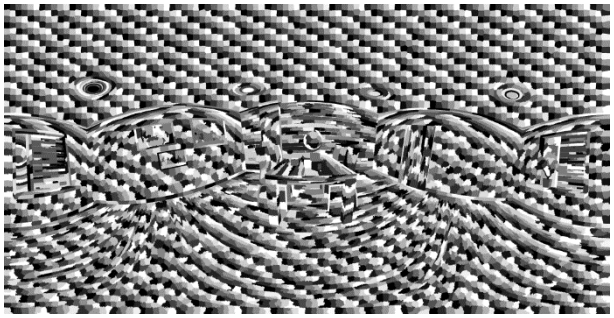


Fig. 5. Result of unmodified superpixel segmentation for an equirectangular image.

First of all, the initial segmentation of 360 video should be based on the equirectangular projection. First of all, as in the process of unmodified segmentation, the average distance between centers of segments is calculated as square root of the average size of a segment. This average distance is used to calculate the number of superpixels on the 'equator' (central row) of an equirectangular image. The number of superpixels in rows that are above or under the equator is proportionally lower, because these rows represent circles on a sphere that are smaller than the circle represented by the equator. The result of such initial grid of superpixels in an equirectangular image is presented in Fig. 6.

The calculation of the spatial distance in case of an omnidirectional image has to be based not simply on the difference of positions of two points in an image, but on the distance between these points before the equirectangular projection, using appropriate formulas.

The final result of such modified superpixel segmentation, adapted to equirectangular images, can be seen in Fig. 7. The size of segments in the center of an image is smaller than in unmodified superpixel segmentation, while the size of segments in the top and the bottom of an image is much larger, therefore, the proposed segmentation better represents real relative sizes of objects present in a scene.

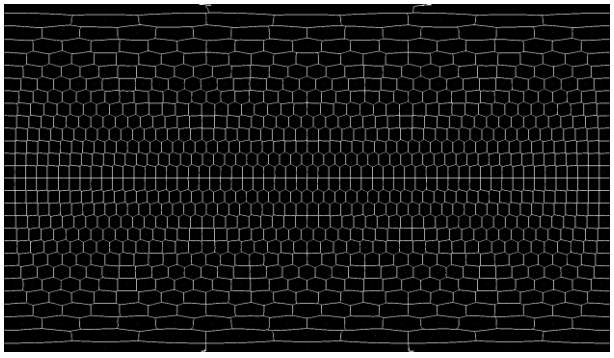


Fig. 6. Proposed initial grid of superpixels used in segmentation of an equirectangular image.

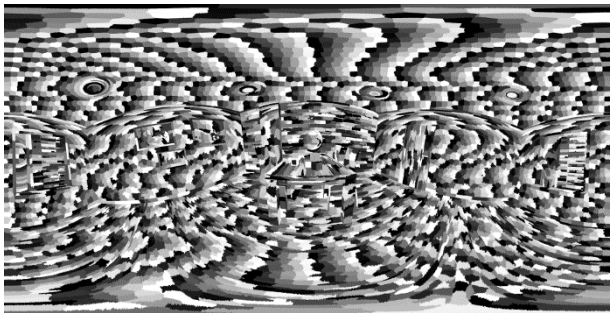


Fig. 7. Result of modified superpixel segmentation for an equirectangular image.

3 Experimental results

In order to fully test the proposed method, a series of experimental tests was performed. Tests followed Common Test Conditions [3] and Exploration Experiments evaluation scheme [4]. The proposal was compared with RDE 0.94 [5].

Only one exception from the CTC was made: in ULB Baby Unicorn sequence the z-near was changed, as the previous one does not include whole range of depth (for views 20 to 24).

The comparison includes:

- softwares with no parallelization, 150 000 segments per view for IVDE (Table 1).
- multi-threaded (8 threads for RDE and 2 for IVDE) versions of softwares, 150 000 segments per view for IVDE (Table 2)
- multi-threaded (8 threads for RDE and 2 for IVDE) versions of softwares, 50 000 segments per view for IVDE (Table 3)

Additionally, comparison of depth estimated for ClassroomVideo (vs. original, synthetic depth) using TMIV4 are presented in document M53567 [7].

Table 1. RDE (no parallelization) vs IVDE/ETRI depth estimation
(no parallelization, 150 000 segments per view)

Test sequence	Method	IV-PSNR (dB)	ΔIV-PSNR (max – min) (dB)	Y-PSNR (dB)	ΔY-PSNR (dB)	U-PSNR (dB)	ΔU-PSNR (dB)	V-PSNR (dB)	ΔV-PSNR (dB)	Time per one view and frame (sec)	Memory per one view (GB)
IntelFrog	RDE	38.26	6.92	27.84	5.62	42.36	3.93	41.18	4.72	188.9	3.4
	IVDE	37.29	7.45	27.06	5.53	41.84	4.60	40.17	5.63	203.8	0.3
	Diff	-0.97	0.53	-0.78	-0.09	-0.52	0.67	-1.02	0.90	14.9	-3.1
Orange Dancing	RDE	42.54	5.91	32.09	3.84	49.75	3.24	51.48	3.73	117.8	6.1
	IVDE	41.85	5.25	30.85	3.52	49.17	2.79	50.96	3.47	77.6	0.3
	Diff	-0.69	-0.66	-1.24	-0.31	-0.58	-0.44	-0.52	-0.26	-40.2	-5.8
Orange Kitchen	RDE	39.17	7.06	31.38	7.15	46.84	8.67	49.20	11.60	150.7	4.9
	IVDE	41.33	7.24	32.93	4.90	47.61	9.13	49.92	12.39	80.7	0.3
	Diff	2.15	0.18	1.55	-2.25	0.78	0.47	0.72	0.79	-70.0	-4.6
Orange Shaman	RDE	46.34	11.12	38.72	6.82	48.46	7.97	46.39	7.56	116.3	4.7
	IVDE	46.76	9.83	38.47	6.16	48.72	6.97	46.68	6.98	89.1	0.3
	Diff	0.42	-1.29	-0.26	-0.66	0.26	-1.00	0.30	-0.58	-27.2	-4.4
Poznan Fencing	RDE	38.34	4.37	30.13	2.76	45.44	2.16	44.56	1.97	337.7	6.1
	IVDE	39.98	5.92	29.58	4.03	45.05	3.26	44.18	2.72	278.4	0.3
	Diff	1.65	1.55	-0.55	1.27	-0.39	1.10	-0.38	0.75	-59.3	-5.8
TechnicolorPainter	RDE	40.38	9.84	31.70	7.30	45.52	5.61	44.46	6.69	188.2	5.8
	IVDE	40.40	8.23	32.38	5.58	45.87	5.42	44.93	6.35	225.0	0.4
	Diff	0.03	-1.61	0.68	-1.71	0.35	-0.19	0.48	-0.34	36.8	-5.4
ULBBaby Unicorn	RDE	34.33	11.70	27.37	8.66	37.70	9.50	36.98	7.49	1290.3	17.1
	IVDE	35.17	10.52	27.83	8.18	38.08	6.46	37.67	4.63	153.8	0.3
	Diff	0.84	-1.18	0.47	-0.48	0.39	-3.03	0.69	-2.86	-1136.5	-16.8
ULBUnicornA	RDE	40.14	2.43	30.96	3.79	43.84	2.65	44.04	2.66	312.6	7.7
	IVDE	39.14	2.21	29.26	3.32	43.12	2.47	43.45	2.40	245.0	0.3
	Diff	-1.01	-0.23	-1.70	-0.48	-0.72	-0.17	-0.58	-0.26	32.4	-7.4
ULBUnicornB	RDE	40.78	2.18	31.96	2.11	44.29	1.68	44.62	1.41	348.0	9.7
	IVDE	40.21	2.14	30.41	3.11	43.88	2.02	43.97	1.62	386.1	0.3
	Diff	-0.57	-0.05	-1.55	1.00	-0.40	0.34	-0.66	0.20	38.1	-9.4
Average	RDE	40.03	6.84	31.35	5.34	44.91	5.04	44.77	5.32	338.9	7.3
	IVDE	40.24	6.53	30.98	4.93	44.82	4.79	44.66	5.13	204.4	0.3
	Diff	0.21	-0.31	-0.37	-0.41	-0.09	-0.25	-0.11	-0.18	-134.5	-6.9

Table 2. RDE (parallelization using 8 threads) vs IVDE depth estimation
(parallelization using 2 threads, 150 000 segments per view)

Test sequence	Method	IV-PSNR (dB)	ΔIV-PSNR (dB)	Y-PSNR (dB)	ΔY-PSNR (dB)	U-PSNR (dB)	ΔU-PSNR (dB)	V-PSNR (dB)	ΔV-PSNR (dB)	Time per one view and frame (sec)	Memory per one view (GB)
IntelFrog	RDE	38.26	6.92	27.84	5.62	42.36	3.93	41.18	4.72	168.8	3.4
	IVDE	37.17	7.19	26.96	5.45	41.81	4.55	40.10	5.48	182.6	0.5
	Diff	-1.09	0.27	-0.89	-0.17	-0.55	0.62	-1.09	0.75	13.8	-2.9
Orange Dancing	RDE	42.54	5.91	32.09	3.84	49.75	3.24	51.48	3.73	107.3	6.1
	IVDE	41.85	5.28	30.85	3.54	49.17	2.77	50.96	3.52	48.8	0.5
	Diff	-0.69	-0.63	-1.24	-0.29	-0.58	-0.47	-0.52	-0.22	-58.4	-5.6
Orange Kitchen	RDE	39.17	7.06	31.38	7.15	46.84	8.67	49.20	11.60	136.5	4.9
	IVDE	41.26	7.21	32.90	4.82	47.55	9.13	49.92	12.41	51.6	0.6
	Diff	2.09	0.15	1.52	-2.32	0.71	0.47	0.72	0.80	-84.9	-4.3
Orange Shaman	RDE	46.34	11.12	38.72	6.82	48.46	7.97	46.39	7.56	108.0	4.7
	IVDE	46.99	9.80	38.58	6.07	48.85	6.95	46.83	6.79	54.7	0.6
	Diff	0.65	-1.31	-0.14	-0.75	0.39	-1.01	0.45	-0.77	-53.3	-4.1
Poznan Fencing	RDE	38.34	4.37	30.13	2.76	45.44	2.16	44.56	1.97	309.8	6.1
	IVDE	39.98	5.92	29.58	4.03	45.05	3.26	44.18	2.72	212.4	0.6
	Diff	1.64	1.55	-0.55	1.27	-0.39	1.10	-0.38	0.75	-97.3	-5.6
TechnicolorPainter	RDE	40.38	9.84	31.70	7.30	45.52	5.61	44.46	6.69	170.3	5.8
	IVDE	40.52	8.65	32.44	5.80	45.88	5.46	44.93	6.39	165.4	0.6
	Diff	0.15	-1.19	0.74	-1.49	0.35	-0.15	0.48	-0.30	-4.9	-5.2
ULBBaby Unicorn	RDE	34.33	11.70	27.37	8.66	37.70	9.50	36.98	7.49	1102.6	17.1
	IVDE	35.32	10.40	27.90	8.04	38.08	6.46	37.67	4.63	92.8	0.5
	Diff	0.99	-1.30	0.54	-0.62	0.39	-3.03	0.69	-2.86	-1009.7	-16.6
ULBUnicornA	RDE	40.14	2.43	30.96	3.79	43.84	2.65	44.04	2.66	285.0	7.7
	IVDE	39.11	2.24	29.24	3.31	43.12	2.45	43.45	2.40	254.0	0.5
	Diff	-1.04	-0.19	-1.72	-0.48	-0.72	-0.19	-0.59	-0.26	-31.0	-7.2
ULBUnicornB	RDE	40.78	2.18	31.96	2.11	44.29	1.68	44.62	1.41	318.1	9.7
	IVDE	40.19	2.18	30.33	2.73	43.89	2.06	44.02	1.67	261.9	0.6
	Diff	-0.59	0.00	-1.63	0.62	-0.40	0.38	-0.60	0.26	-56.3	-9.1
Average	RDE	40.03	6.84	31.35	5.34	44.91	5.04	44.77	5.32	300.7	7.3
	IVDE	40.27	6.54	30.98	4.87	44.82	4.79	44.67	5.11	147.1	0.5
	Diff	0.23	-0.30	-0.37	-0.47	-0.09	-0.25	-0.09	-0.21	-153.6	-6.7

Table 3. RDE (parallelization using 8 threads) vs IVDE depth estimation (parallelization using 2 threads, 50 000 segments per view).

Test sequence	Method	IV-PSNR (dB)	ΔIV-PSNR (dB)	Y-PSNR (dB)	ΔY-PSNR (dB)	U-PSNR (dB)	ΔU-PSNR (dB)	V-PSNR (dB)	ΔV-PSNR (dB)	Time per one view and frame (sec)	Memory per one view (GB)
IntelFrog	RDE	38.26	5.62	27.84	5.62	42.36	3.93	41.18	4.72	168.8	3.4
	IVDE	36.90	5.01	26.80	5.53	41.86	4.21	39.99	4.49	42.6	0.2
	Diff	-1.36	-0.61	-1.04	-0.09	-0.50	0.28	-1.19	-0.23	-126.2	-3.2
Orange Dancing	RDE	42.54	3.84	32.09	3.84	49.75	3.24	51.48	3.73	107.3	6.1
	IVDE	41.42	2.79	30.22	3.52	48.71	2.54	50.65	3.49	12.1	0.2
	Diff	-1.12	-1.04	-1.87	-0.31	-1.05	-0.70	-0.83	-0.24	-95.2	-5.9
Orange Kitchen	RDE	39.17	7.15	31.38	7.15	46.84	8.67	49.20	11.60	136.5	4.9
	IVDE	41.02	4.46	32.29	4.90	47.22	8.64	49.60	12.12	15.4	0.2
	Diff	1.84	-2.69	0.91	-2.25	0.38	-0.03	0.41	0.51	-121.1	-4.7
Orange Shaman	RDE	46.34	6.82	38.72	6.82	48.46	7.97	46.39	7.56	108.0	4.7
	IVDE	46.10	5.28	37.52	6.16	48.08	6.50	46.00	6.17	17.1	0.2
	Diff	-0.24	-1.55	-1.21	-0.66	-0.38	-1.46	-0.39	-1.39	-90.9	-4.5
Poznan Fencing	RDE	38.34	2.76	30.13	2.76	45.44	2.16	44.56	1.97	309.8	6.1
	IVDE	39.46	3.74	29.17	4.03	44.93	3.11	44.11	2.73	42.9	0.2
	Diff	1.12	0.98	-0.96	1.27	-0.51	0.95	-0.45	0.76	-266.9	-5.9
TechnicolorPainter	RDE	40.38	7.30	31.70	7.30	45.52	5.61	44.46	6.69	170.3	5.8
	IVDE	40.57	6.44	32.61	5.58	45.85	5.38	44.91	6.36	35.4	0.2
	Diff	0.19	-0.85	0.91	-1.71	0.32	-0.23	0.46	-0.33	-134.9	-5.6
ULBBaby Unicorn	RDE	34.33	8.66	27.37	8.66	37.70	9.50	36.98	7.49	1102.6	17.1
	IVDE	35.01	7.80	27.67	8.18	38.03	6.33	37.61	4.49	36.5	0.2
	Diff	0.69	-0.86	0.30	-0.48	0.34	-3.16	0.63	-3.00	-1066.1	-16.9
ULBUnicornA	RDE	40.14	3.79	30.96	3.79	43.84	2.65	44.04	2.66	285.0	7.7
	IVDE	38.96	3.43	28.93	3.32	42.89	2.77	43.27	2.48	56.6	0.2
	Diff	-1.18	-0.36	-2.03	-0.48	-0.95	0.12	-0.76	-0.18	-228.4	-7.5
ULBUnicornB	RDE	40.78	2.11	31.96	2.11	44.29	1.68	44.62	1.41	318.1	9.7
	IVDE	40.06	2.83	29.98	3.11	43.88	2.13	43.98	1.79	57.3	0.2
	Diff	-0.72	0.72	-1.98	1.00	-0.41	0.45	-0.64	0.38	-260.8	-9.5
Average	RDE	40.03	5.34	31.35	5.34	44.91	5.04	44.77	5.32	300.7	7.3
	IVDE	39.94	4.64	30.58	4.93	44.60	4.62	44.46	4.90	35.1	0.2
	Diff	-0.09	-0.70	-0.77	-0.41	-0.30	-0.42	-0.31	-0.41	-265.6	-7.1

Table 4. RDE (parallelization using 8 threads) vs IVDE depth estimation (parallelization using 2 threads, 50 000 segments per view) and IVDE depth estimation (parallelization using 2 threads, 150 000 segments per view)

Method		IV-PSNR (dB)	ΔIV-PSNR (dB)	Y-PSNR (dB)	ΔY-PSNR (dB)	U-PSNR (dB)	ΔU-PSNR (dB)	V-PSNR (dB)	ΔV-PSNR (dB)	Time per one view and frame (sec)	Memory per one view (GB)
RDE		40.03	6.84	31.35	5.34	44.91	5.04	44.77	5.32	300.7	7.3
Diff	IVDE (150k)	0.23	-0.30	-0.37	-0.47	-0.09	-0.25	-0.09	-0.21	-153.6	-6.7
	IVDE (50k)	-0.09	-0.70	-0.77	-0.41	-0.30	-0.42	-0.31	-0.41	-265.6	-7.1

The presented results show that the objective quality of RDE and IVDE is very similar. However, results show that IVDE achieves:

- smaller deltas of PSNR – result of simultaneous estimation for all input views, depth maps are also much more inter-view consistent,
- much shorter time of depth estimation (per one view and frame),
- much smaller memory consumption, dependent only on number of cameras and number of segments per view

Moreover, all configuration parameters, except for one (SmoothingCoefficient) are common for all the sequences. Below, the results of TMIV with automatic calculation of this coefficient were added.

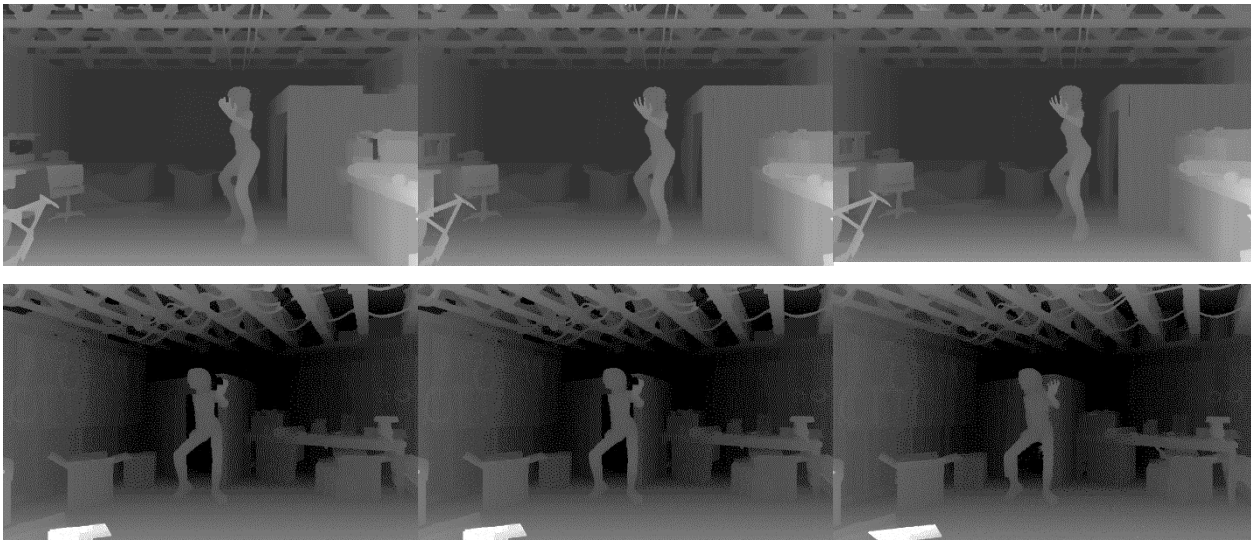
Table 5. IVDE depth estimation (parallelization using 2 threads, 50 000 segments per view) and IVDE depth estimation with automatic smoothing coefficient calculation (parallelization using 2 threads, 150 000 segments per view)

Method	IV-PSNR (dB)	ΔIV-PSNR (dB)	Y-PSNR (dB)	ΔY-PSNR (dB)	U-PSNR (dB)	ΔU-PSNR (dB)	V-PSNR (dB)	ΔV-PSNR (dB)	Time per one view and frame (sec)	Memory per one view (GB)
IVDE	40.27	6.54	30.98	4.87	44.82	4.79	44.67	5.11	147.1	0.5
IVDE auto smoothing	40.11	6.30	31.00	4.82	44.85	4.76	44.67	4.89	166.4	0.5

RDE IVDE (50k) IVDE (150k)

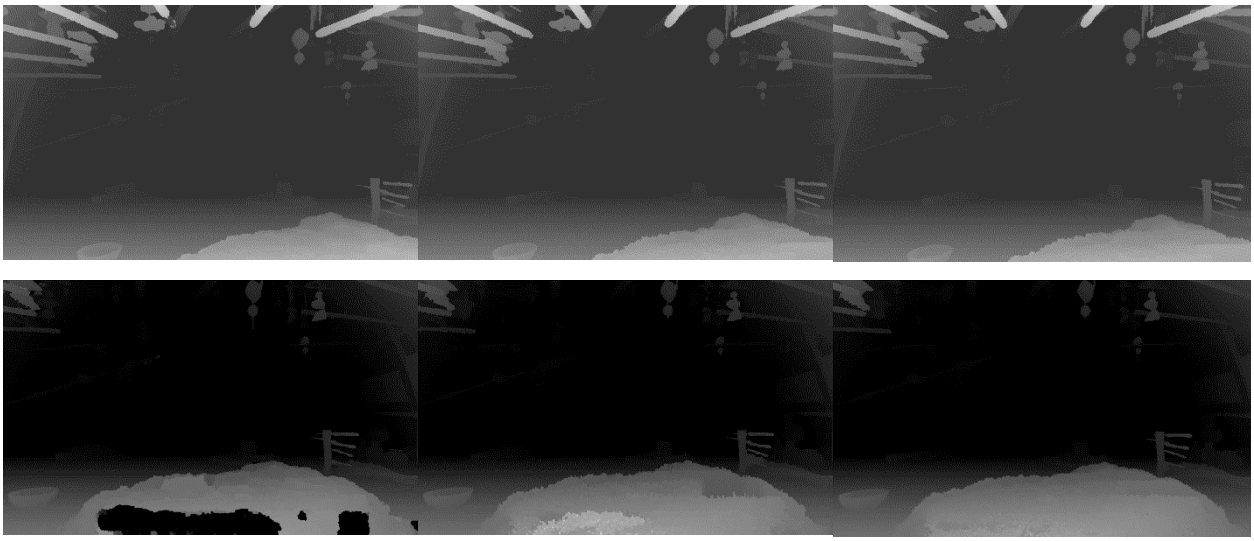
Intel Frog

Orange Dancing

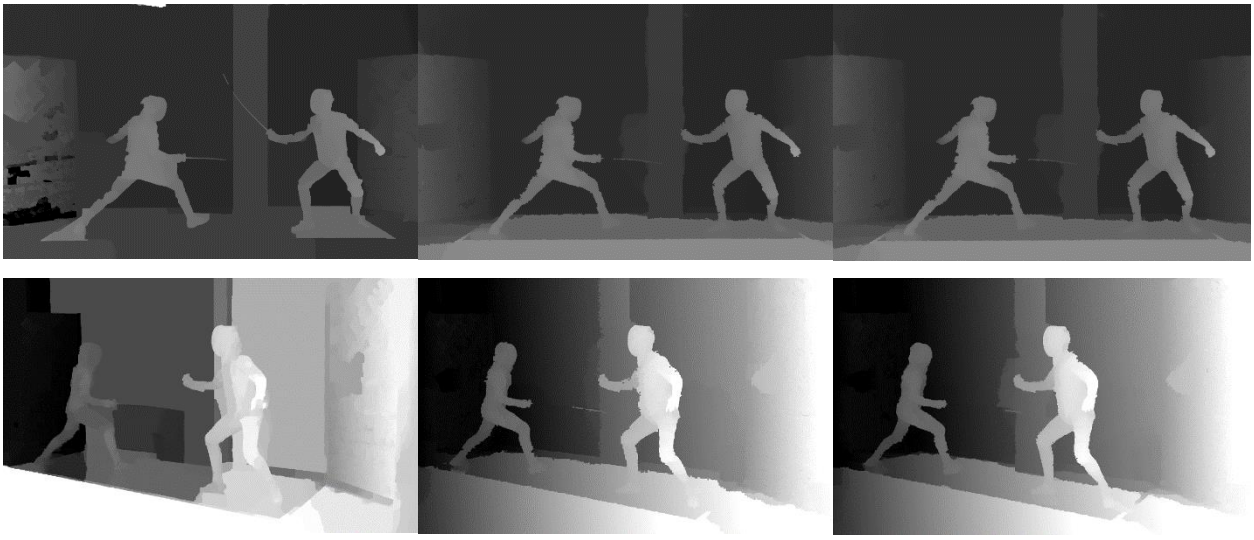


Orange Kitchen

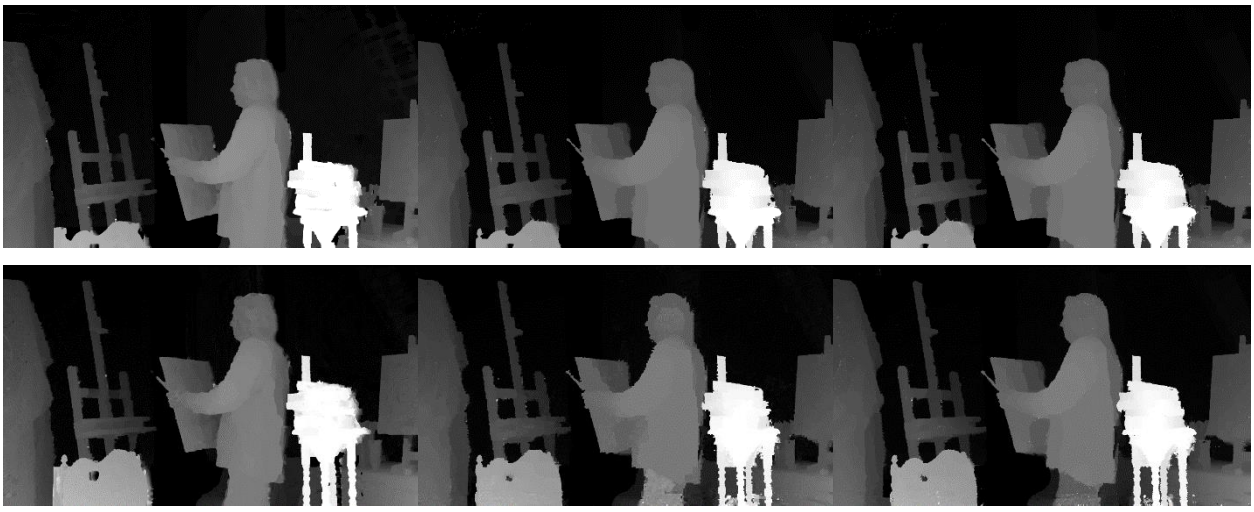
Orange Shaman



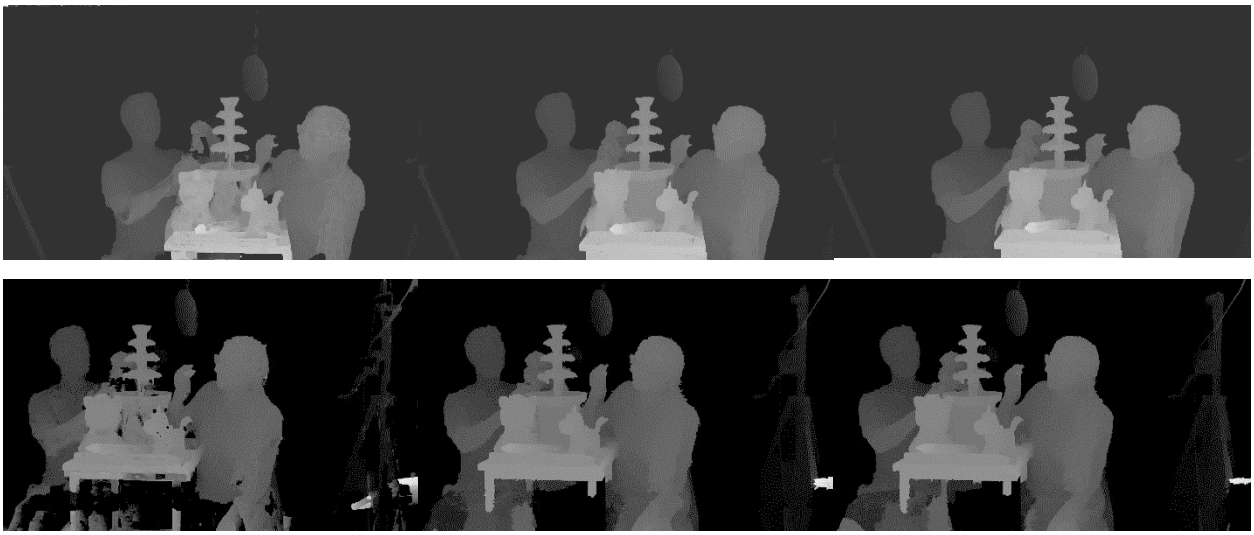
Poznan Fencing



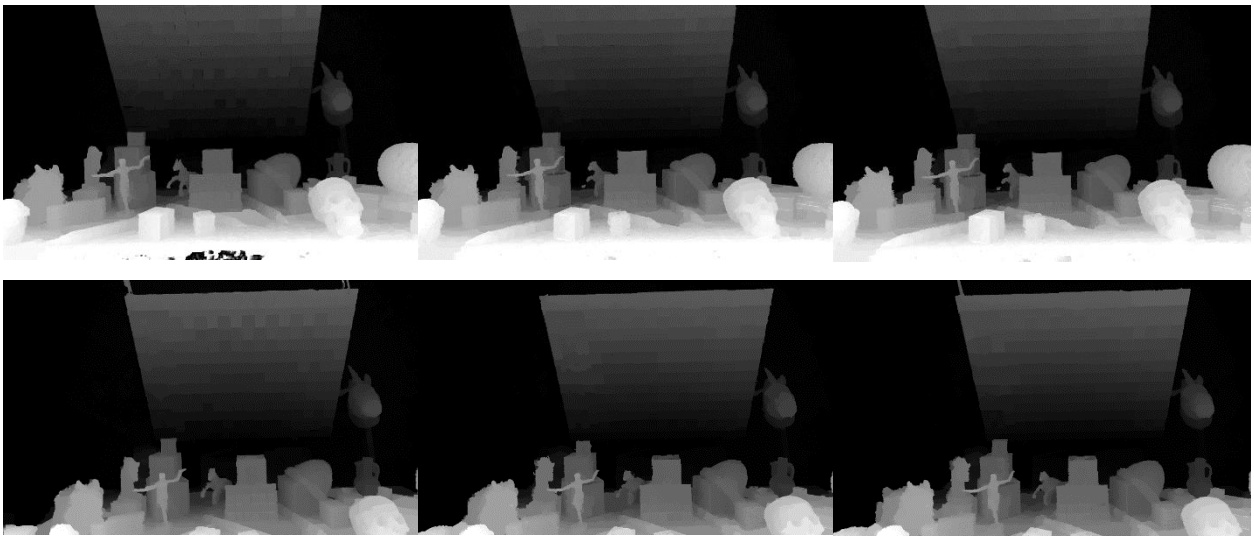
Technicolor Painter



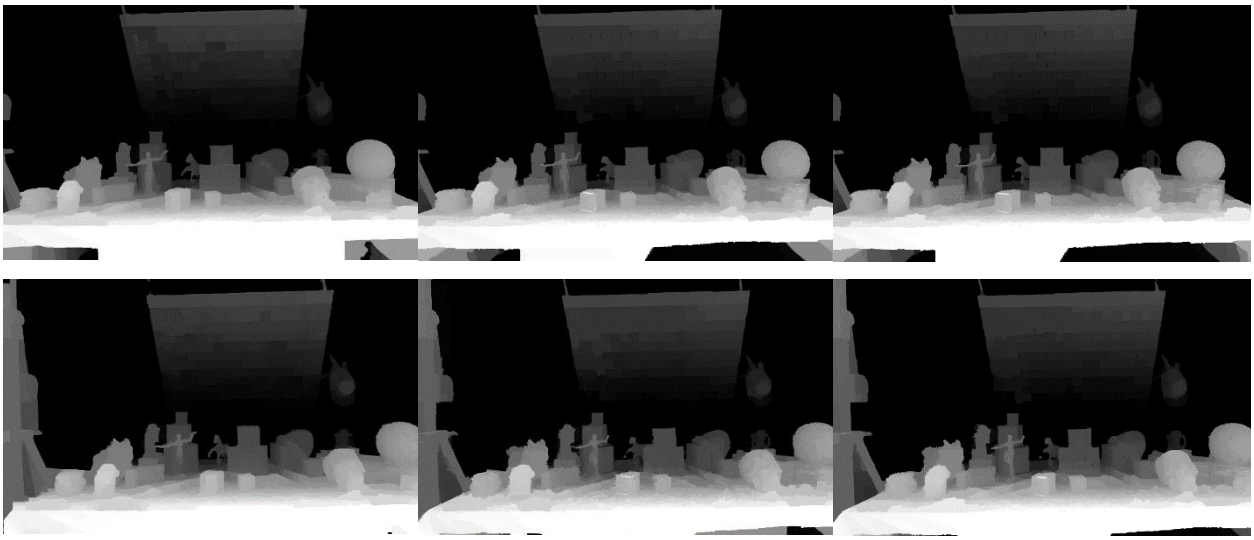
ULB Baby Unicorn



ULB Unicorn A



ULB Unicorn B



4 Overview of software

The software is written in C++, uses OpenMP parallelization, and does not require any additional libraries. Depth estimation for perspective and omnidirectional videos (full 360 only at this moment) is possible. Possible input textures are 8-bit, cf 420 and cf 444. The package includes configuration files for all tested sequences.

5 Conclusion

The document provides a brief description of the technique and software IVDE for depth estimation from perspective and omnidirectional video, in particular for 3DoF+ and beyond. Full source code of the method is provided for the MPEG-I activities.

The proposal shows very good performance in terms of quality of depth maps and computational complexity. Estimated depth maps are also inter-view consistent, what significantly increases the subjective quality of virtual navigation. Moreover, the method achieves high quality of depth maps for unified configuration files (with only one parameter changed).

6 Recommendations

We propose:

- to make this software a new/additional reference software for depth estimation,
- to change the z_{near} value for ULBBabyUnicorn.

Acknowledgement

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

References

[1] D. Mieloch, O. Stankiewicz and M. Domański, "Depth Map Estimation for Free-Viewpoint Television and Virtual Navigation," IEEE Access, vol. 8, pp. 5760-5776, 2020.

[2] R. Achanta and S. Süsstrunk, “Superpixels and Polygons using simple non-iterative clustering,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, pp. 4895–4904.

[3] Common Test Conditions for Immersive Video, ISO/IEC JTC1/SC29/WG11 MPEG2019/N18997, Brussels, January 2020.

[4] Exploration Experiments for MPEG-I Visual: 6DoF, ISO/IEC JTC1/SC29/WG11 MPEG2019/N18998, Brussels, January 2020.

[5] Jaime Sancho, Takanori Senoh, Ségolène Rogge, Daniele Bonatto, Rubén Salvador, Eduardo Juarez, Adrian Munteanu, Gauthier Lafruit, “[MPEG-I Visual] RDE fine-tuning to achieve DERS 8.0 performance”, ISO/IEC SC29/WG11 MPEG2020/M52135, Brussels, January 2020.
[6] V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 2, pp. 147–159, Feb. 2004.
[7] D. Mieloch, “[MPEG-I Visual] Estimated depth maps for ClassroomVideo sequence”, ISO/IEC SC29/WG11 MPEG2020/M53567, Online, April 2020.