Funded by the European Union

Holographic Vision for Immersive Tele-Robotic OperatioN

HoviTron

Call identifier: H2020-ICT-2019-3

Grant Agreement: 951989

D2.1 – Test conditions and refactorization

Project funded by the European Commission within the H2020 Programme (2014-2020)			
Start date/ Duration	01 June 2020 / 24 months		
Deliverable leader	Eduardo Juárez		
Deliverable contributor(s)	Jaime Sancho, Guillermo Vazquez, Miguel Chavarrías, César Sanz, Eduardo Juárez		
Due date (DoA)	31/12/2020	**Submission date**	24/12/2020

Type		
R	Document, report excluding the periodic and final reports	x
DEC	Websites, patents filing, press & media actions, videos, etc.	
ETHICS	Ethics requirement	
ORDP	Open Research Data Pilot	
Dissemination level		
PU	PUBLIC, fully open, no embargo e.g. web	x
CO	CONFIDENTIAL, only for members of the consortium (including the Commission Services)	

1 Introduction

This deliverable addresses the first step in the generation of real-time high-quality depth maps using RGB conventional cameras. To do so, MPEG-I (Immersive) Depth Estimation Reference Software (DERS) [DERS] and MPEG-I Immersive Video Depth Estimation (IVDE) [IVDE] are analyzed and refactorized, as they are going to be the basis of the HoviTron software. These applications aim at producing high-quality depth maps, with processing times far from the real-time constraints required in HoviTron. Consequently, their quality results and processing times will be considered, initially, as higher bounds. The analysis and refactorization, hence, will be employed to find and isolate the different stages included in the applications. In the future work of HoviTron, these stages will be accelerated according to a trade-off research in terms of quality and processing time.

Along with DERS and IVDE, MPEG-I specifies common test conditions that ensures the correct evaluation of new tool proposals. They include test material, i.e., multi-view RGB sequences, as well as a common procedure to assess the quality and the processing time. In this deliverable, MPEG common test conditions have been used to perform the analysis of the tools, however, during the project, and beginning with this document, a new set of HoviTron test conditions will be developed. In this context, and addressed in this deliverable, one of the first decisions to make is the number of cameras and the shape of the HoviTron multiview camera array.

2 State of the Art Software

HoviTron real-time software will find its foundation in two MPEG-I applications: DERS and IVDE. In this section, they are analyzed and divided into interconnected stages. This study is needed for the implementation of a software that includes accelerated stages of DERS, IVDE or both.

2.1 Depth Estimation Reference Software (DERS)

DERS [DERS] has been the reference software for depth estimation in MPEG-I explorative experiments until October 2020, with contributions of many developers across the years. Due to continuous updates, which are still happening at this moment, DERS is in its nineth version, which can demonstrate that this software is one of the current state of the art tools.

DERS is an application that uses as input two or more cameras along with its camera parameters, intrinsic and extrinsic, to produce a depth map in the position of one of these cameras, which is called reference camera. To do so, it also needs a configuration file, where not only the camera files are provided, but it also contains several user-configurable parameters that can vary the output result in different ways. The adjustment of these parameters requires a knowledge of the inner stages of the software, hence, DERS also includes a user manual [DERS-Manual].

After the analysis and refactorization of DERS, seven main stages were found. They can be seen in Figure 1 and divided into three groups: initialization, in green, auxiliary processes, in gray, and core, in blue. In addition, four of these stages, in dashed lines, are optional.

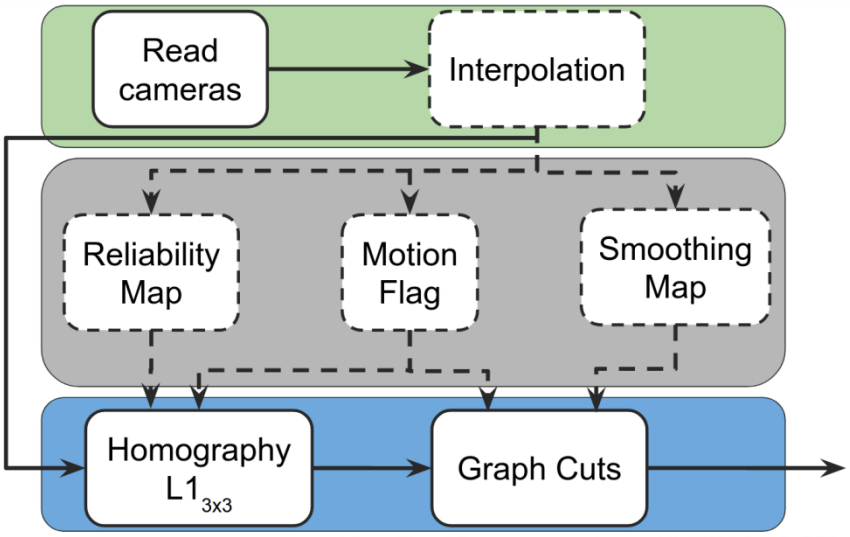

Figure 1. DERS block diagram.

The explanation for every stage is the following:

1) **Read Cameras:** parses the configuration file, reads camera textures and camera parameters and performs some initialization operations.
2) **Interpolation:** (optional) interpolates the images read in the previous stage by a user-defined factor.
3) **Reliability Map:** (optional) using the reference camera as input, this stage produces a vertical and horizontal reliability map that will be used in the homography stage. In these maps, each pixel is

the average of the horizontal or vertical absolute differences in a neighbor area of 3×3 pixels and scaled by a user-defined parameter. These maps provide an idea of the texture level in the surroundings of every pixel, since in areas with low level of texture, the depth estimation is more difficult.

4) **Motion Flag:** (optional) in this stage, the current frame of the reference camera is compared with the previous one, to obtain a binary mask. This mask will be used both in the homography and graph cuts stages to update only the pixels that have changed since the previous frame.

5) **Smoothing Map:** (optional) this stage is similar to the reliability map mentioned before. It produces vertical and horizontal maps, where every pixel indicates the level of continuity in a neighbor window of 3×3 pixels. As well as the reliability map, this value is scaled by a user-defined parameter. These maps are used in the graph cuts stage.

6) **Homography L1:** this stage uses the geometrical information found in the intrinsic parameters (*K* matrix) and extrinsic parameters (*R*|*t* matrix) as well as their corresponding images to produce an initial cost cube. To do so, it relies on the sweeping plane algorithm, depicted in Figure 2. Firstly, the 3D space is bounded between a Z_{near} and a Z_{far} plane, in the *z* axis, and divided into $N_{candidates}$ planes (in black). Then, a ray between the optical axis of the reference camera and a *pixel (I,j)* is created (in red). The intersection of this ray and the intermediate planes leads to a set of 3D points denominated $Z_{candidates}$, which will be all the possible depth values for the *pixel(I,j)* (in blue). Finally, these points are projected in another camera, producing the pixel candidates for the reference *pixel(I,j)* (in green). This process is implemented using the homography matrix in Equation 1. Once the candidates are calculated, they are compared with the reference camera using a Sum of Absolut Differences (SAD) in a 3×3 spatial window. The result is a cost for every pixel and candidate, which is represented as a cost cube, where *x* and *y* are the spatial dimensions of the reference camera, and *z* is a depth candidate. In addition, if the reliability map was generated and depending on the dominant direction of the camera pair, the vertical or horizontal reliability map is used to scale the initial cost cube. This process is repeated for the surrounding cameras, creating a new pair reference-secondary camera for every one of them, and keeping in the cost cube the minimum cost values. In this way, the minimum cost for every pixel and depth in the surrounding cameras is retained in the cost cube.

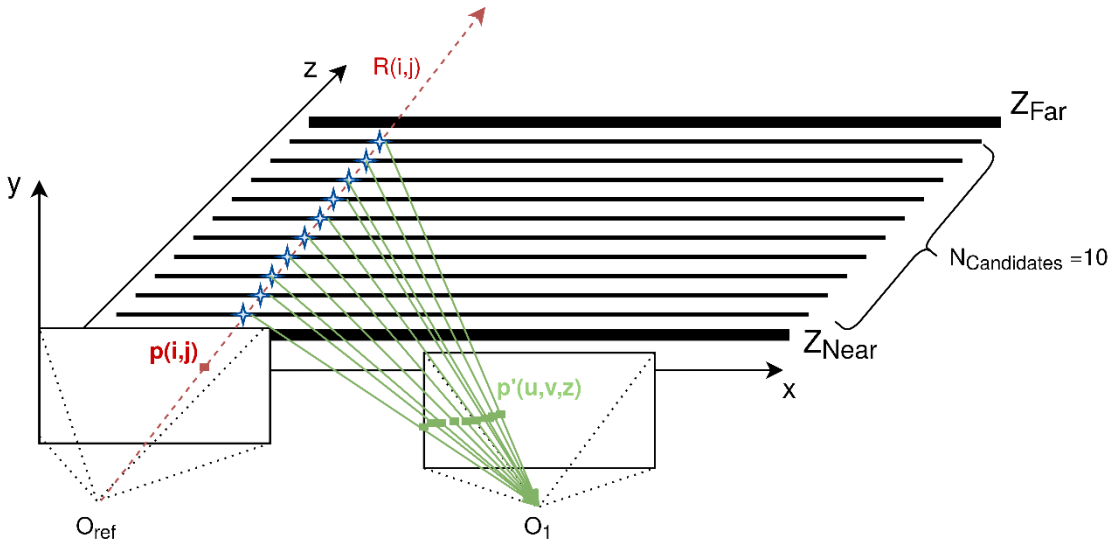

Figure 2. Sweeping plane algorithm.

$$\frac{1}{s}\begin{bmatrix} u \\ v \\ s \\ 1 \end{bmatrix} = \begin{pmatrix} K_{cam}[R_{cam}|t_{cam}] \left(\begin{matrix} K_{ref}[R_{ref}|t_{ref}] \\ 0 \quad 0 \quad 0 \quad 1 \end{matrix}\right)^{-1} \\ 0 \quad 0 \quad 0 \quad 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ 1 \\ 1/z \end{pmatrix}$$

Equation 1. Homography matrix.

7) **Graph Cuts:** starting from the cost cube generated in the homography stage, Graph Cuts aims at obtaining the optimal depth map, given that (i) a pixel is likely to have the depth assigned to the smallest candidate cost and (ii) that similar color regions normally have akin depth values (without large depth differences). These constraints are formulated in Equation 2 as the energy function to minimize.

$$E(f) = E_{data}(f) + E_{smooth}(f) = \sum_{p \in \mathcal{P}} C(p, f_p) + \sum_{(p,q) \in \mathcal{N}} S(p,q) V_{p,q}(f_p, f_q)$$

Equation 2. Graph cuts energy function.

Where *f* is a labelling, i.e., a depth map, that produces a specific energy value, *P* are the image pixels and *N* is a neighborhood of pixels. The first term, the data term, measures the energy of assigning one pixel to a certain depth using the information from the initial cost cube. The second term, the smoothing term, measures the energy of assigning one pixel to a certain depth given its similarity to its neighborhood, by means of the smoothing map and the labelling of the neighbor pixels. Graph cuts uses a graph structure to solve this energy minimization problem, and it is implemented through Kolmogorov et al. library [Graph-Cuts].

It is important to remark that DERS is a view-output oriented application, which means that one instance of the program will produce only one output view (the reference camera one). This entails that, if a depth map for every camera in the dataset is necessary (which is common), the number of DERS instances needed will be the same as the number of cameras. Consequently, as these processes are independent, they can be executed in parallel.

2.2 Immersive Video Depth Estimation (IVDE)

IVDE [IVDE] has become the reference software for depth estimation in MPEG-I explorative experiments in the last months, after competing during a period with DERS. As MPEG-I experts considered this tool to be more suitable for depth experiments, HoviTron consortium decided to evaluate it, as it may be interesting for the project.

IVDE's analysis and refactorization has resulted in eight main stages represented in Figure 3. They are grouped in two main blocks: initialization, represented in green, and the core of the depth estimation, represented in blue.

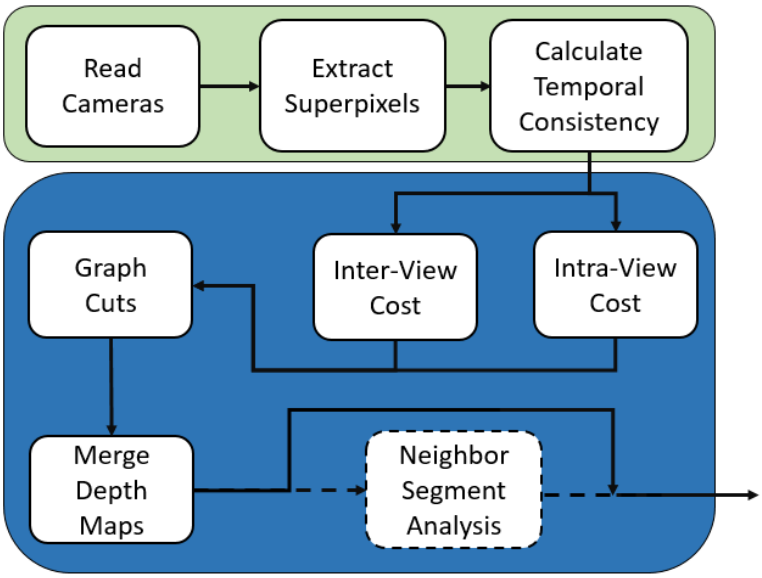

Figure 3. IVDE block diagram.

1) **Read Cameras:** loads the configuration parameters and carries out initialization tasks.
2) **Extract Superpixels:** performs the segmentation of the image in superpixels. Initially, the image is divided into a grid of a fixed number of superpixels. These segments are shaped according to the color and spatial distance between the image pixels and the center of their neighbor superpixels.
3) **Calculate Temporal Consistency**: compares segments between consecutive frames to mark unchanged ones. For the first frame (I-type), depth is estimated for all the segments. In the following frames (P-type), the depth is calculated only for those segments marked as changed.
4) **Intra-View Cost:** calculates the cost between neighbor segments in the same view according to the following equation:

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

Where d_s and d_t are the currently considered depth of the segment s and its neighbor t. The smoothing coefficient β is previously calculated as the L1 distance between the average YCbCr components of a segment and its neighbor according to the expression:

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

Where β_0 is a user-defined parameter.

5) **Inter-View Cost:** calculates the cost between a segment and its corresponding segment from a neighbor view as the average of the L1 norm inside a given window. The calculation is performed according to:

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

Where μ_s is the vector of coordinates with centre in segment s, $T[\mu_s]$ is the transformed segment according to the cameras parameters and w is the vector of coordinates of a point inside the given window W. The final inter-view cost is calculated through:

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

Where $d_{s'}$ is the currently considered depth for the respective transformed in the neighbor view of the segment s.

6) **Graph Cuts:** performs the minimization of the energy function that results from the combination of the intra-view ($V_{s,t}(d_s, d_t)$) and inter-view ($M_{s,s'}(d_s)$) costs:

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

Graph cuts in IVDE is also implemented through Kolmogorov et al. library [Graph-Cuts], however, to it is parallelized using CPU threads. The basis of the parallelization is that the aforementioned library is used iteratively for each depth plane defined to obtain the final result. In IVDE, this loop is divided into different threads, in such a way that every thread calculates several of the total depth planes. Although this method improves the performance for parallel architectures, it also introduces a certain error and the need of the next stage.

7) **Merge Depth Maps:** combines the depth estimated by each thread in its range of depth levels by repeating the calculation of intra and inter-view costs. This stage will produce the final output for Graph Cuts.
8) **Neighbor Segment Analysis:** (optional) for each segment in the estimated depth map, neighbor segments are taken as candidates (i) if they reduce the inter-cost of the processed segment or (ii) if the corresponding segment in a neighbor view has the same value of depth.

As commented in DERS, it is important to remark that IVDE is a dataset-output oriented application, which means that one instance of the program will produce a depth map for every camera in the dataset.

3 MPEG-I 6DoF Common Test Conditions

In order to test the new tool proposals (or updates) fairly in MPEG-I, there exists a set of common test conditions. These test conditions for depth estimation and visual synthesis are further explained in this section; including the multiview sequences used as input, the validation chain and the metrics involved in the evaluation.

3.1 Test material

The MPEG-I group counts with several multiview test sequences collected from its members, from businesses and universities. They have different properties, which are summarized in Table 1.

Table 1. Test material.

Sequence	Type	Configuration	# Test frames	Resolution
Orange Shaman	Synthetic	Square 5×5	17	1920×1080
Orange Dancing	Synthetic	Arc 14×3	17	1920×1080
Orange Kitchen	Synthetic	Square 5×5	17	1920×1080
Technicolor Painter	Natural	Square 4×4	17	2048×1088
Intel Frog	Natural	Line 1×14	17	1920×1080
ULB Toys	Natural	Square 5×5	1	1920×1080
ULB Chocolat	Natural	Rectangle 3×5	17	1856×1032
Poznan Fencing	Natural	Line 1×10	17	1920×1080

3.2 Validation chain

Most of these sequences do not count with depth maps to compare with, hindering their objective evaluation. As a result, MPEG-I group developed a validation chain inspired by the final goal of the depth maps: its introduction in immersive systems. The chain, as depicted in Figure 4, shows that first, for every camera in the dataset, a depth map is generated using a depth generation tool, namely DERS or IVDE. Then, using a view synthesis tool such as Versatile Video Synthesizer (VVS) [w18172] or Reference View Synthesizer (RVS) [w17759], a synthesized view is generated in the position of the real cameras, taking into consideration the information only of its neighbor cameras. In Figure 4, camera 4 is highlighted to remark that for generating a synthesized view in its position, both RGB images and depth maps of the neighbor cameras are used (but not the information of camera 4).

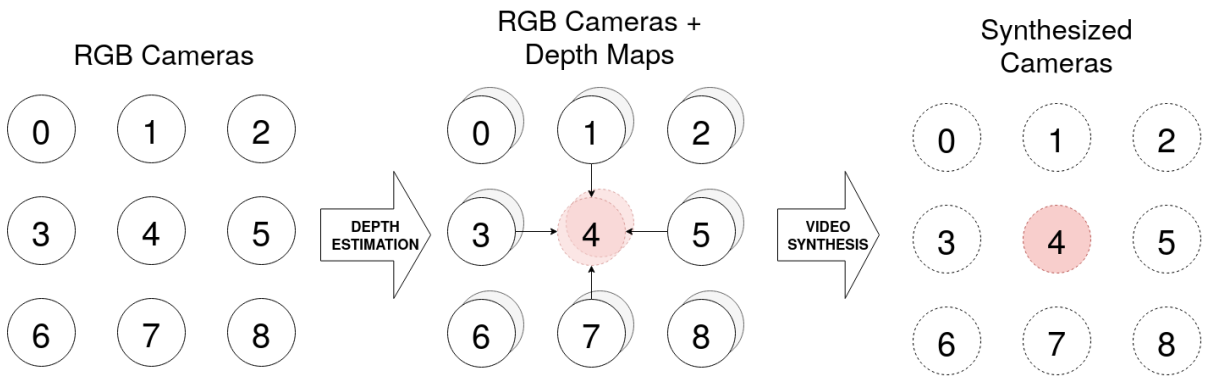

Figure 4. MPEG Validation Chain.

In this way, every camera has a synthesized view that can be compared with the image from the actual camera, which can be converted to an objective quality measurement. Considering that the synthesized views are highly dependent on the depth maps, this strategy can be used to evaluate depth estimation tools.

3.3 Metrics

Once every camera in the dataset has a corresponding synthesized view, there is a need to use a metric to find out the objective quality of the synthesized images. After obtaining the objective quality in a frame and view, all these views and frames are averaged to obtain a single value per dataset. In MPEG-I, the objective quality metrics are the following [w19221]:

1. **Weighted Spherical PSNR (WS-PSNR) [w18069]**: a PSNR where every pixel (I,j) has associated a weight based on the spherical area covered in that position, depending on its projection plane. It is calculated as:

$$WMSE = \frac{1}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} w(i,j)} \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} \big(y(i,j) - y(i,j)\big)^2 \times w(i,j),$$

$$WS - PSNR = 10\log\frac{MAX_I^2}{WMSE},$$

For calculating WS-PSNR in equirectangular projections (ERP), the weights are the following:

$$w(i,j)_{ERP} = cos\frac{(j + 0.5 - N/2)\pi}{N},$$

2. **Immersive Video PSNR (IV-PSNR) [w18709]**: a variation of PSNR for immersive video applications. Its two main differences are (i) Corresponding Pixel Shift and (ii) Global Color Difference. Corresponding Pixel Shift eliminates the influence of a slight shift of objects' edges caused by reprojection errors. Global Color Difference reduces the influence of different color characteristics of different input views. It is calculated for YUV images as:

$$IVPSNR_{YUV} = \frac{\sum_{c=0}^{2} IVPSNR(c) \cdot CCW(c)}{\sum_{c=0}^{2} CCW(c)},$$

where CCW(c) is the Color Component Weight for each color component c and IVPSNR(c) is:

$$IVPSNR(c) = 10\log\left(\frac{MAX^2}{IVMSE(c)}\right),$$

$$\mathrm{IVMSE}(c) = \frac{1}{W \cdot H}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}\min_{\substack{x_R\in[x-\mathrm{CPS},x+\mathrm{CPS}]\\ y_R\in[y-\mathrm{CPS},y+\mathrm{CPS}]}}(c_T(x,y,c) - c_R(x_R,y_R,c) + \mathrm{GCD(c)})^2,$$

where W and H are width and height of the image, c_T (x,y,c) and c_R (x,y,c) are values of color component c in the position (x,y) in the test image and the reference image, respectively, CPS is the maximum Corresponding Pixel Shift between reference and test image, and GCD is the Global Color Difference for component c:

$$\mathrm{GCD}(c) = \max\left(\frac{1}{W \cdot H}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}\left(c_R(x,y,c) - c_T(x,y,c)\right)\ , \mathrm{MUD}(c)\right),$$

where MUDI is the Maximum Unnoticeable Difference for color component c.

CCWI, MUDI and CPS values are predefined. Typical values are:

- CCWI:
  - CCW(0)=1 (luma component),
  - CCW(1)=0.25 (1st chroma component),
  - CCW(2)=0.25 (2nd chroma component),
- MUDI = 1% for all the color components,
- CPS = 2.

3. **Video Multimethod Assessment Fusion (VMAF) [VMAF]:** an objective metric which combines three elementary quality metrics (Visual Information Fidelity, Detail Loss Metric and Motion). A Support Vector Machine (SVM) regression algorithm is employed to weight each elementary metric.

In addition, subjective quality results are also required to complement the evaluation. These results consist of the generation of synthesized video sequences that simulate the movement of a camera across the data-set. These sequences are then evaluated subjectively by the experts in the group.

Another important aspect to measure is the processing time of the depth estimation tool. In this regard, in MPEG-I, only the average time per view is considered.

4 Anchor Results

In order to test DERS and IVDE (described in Section 2), the common test conditions for MPEG-I explorative experiments in depth estimation (reported in Section 3) were followed to obtain the results in Table 2, using VVS 2.0 as synthesis tool. These results were computed in *Centro de Supercomputación y Visualización de Madrid* (CeSViMa), a super-computer with 68 nodes, each one with two Intel Xeon Gold 6230 20 cores @ 2.10 GHz, 192 GB RAM and 480 SSD disk. This platform is chosen given the number of data to process, taking into account that every sequence counts with between 10 and 42 views, with 17 frames (except ULB Unicorn A) every one of them. Consequently, and due to its type of output, DERS uses a number of independent cores equal to the number of views in the test. In the case of IVDE, and given its functioning, every sequence uses 10 cores to process all the views in the dataset. The configuration files used to obtain these results are found in Annex A.

This study is two-fold, (i) it is a comparison between DERS and IVDE, which also provides a higher bound limit for objective quality and processing times (for MPEG-I sequences), and (ii) it is the reference camera setup to compare with, in Section 6.

Quality and processing time results are further explained in the following subsections.

Table 2. Anchor results.

Sequence	Tool	WS-PSNR Y (db)	WS-PSNR U (dB)	WS-PSNR V (dB)	IV-PSNR (dB)	VMAF	Av. Time (s)	Global Delay (s)
Orange Shaman	DERS	39.12	48.83	46.93	44.13	82.80	14266	18302
	IVDE	37.93	48.30	46.31	44.55	79.77	1373	34318
Orange Dancing	DERS	32.84	49.86	51.45	42.05	78.65	6987	8135
	IVDE	31.05	48.77	50.81	40.81	74.55	1011	42474
Orange Kitchen	DERS	31.94	46.65	49.59	40.13	81.72	8429	10310
	IVDE	31.37	46.92	49.74	40.11	80.33	1182	29548
Technicolor Painter	DERS	35.30	46.80	46.92	43.43	85.81	15353	21176
	IVDE	34.71	46.58	46.67	42.86	84.57	2597	41556
Intel Frog	DERS	28.03	42.11	40.99	37.98	73.40	21382	26300
	IVDE	27.22	41.61	39.89	37.26	71.89	2850	37046
ULB ToysTable	DERS	27.34	43.43	43.50	37.62	79.38	1221	19796
	IVDE	25.21	42.19	42.53	35.34	72.43	189	44263
ULB Chocolat	DERS	27.82	38.66	37.79	35.20	66.86	15446	19796
	IVDE	27.84	38.64	37.72	35.01	68.21	2950	44263
Poznan Fencing	DERS	28.20	45.08	41.39	36.64	59.56	24450	28825
	IVDE	25.66	43.64	40.66	36.12	49.06	2596	25962

4.1 Objective quality results

The objective quality results can be better visualized in Figure 5, Figure 6 and Figure 7, with a representation of WS-PSNR Y, IV-PSNR and VMAF, respectively, for every sequence and for both tools.

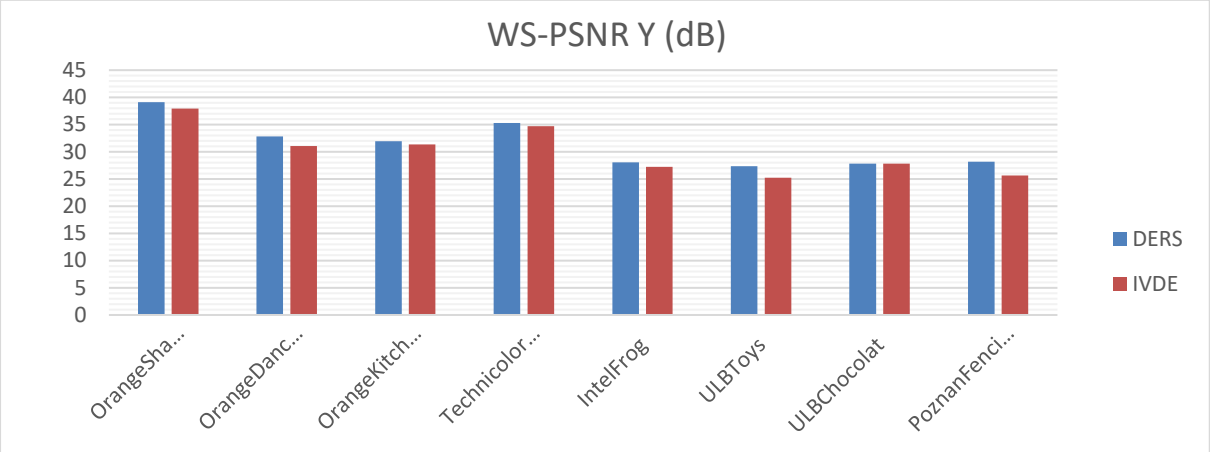

Figure 5. WS-PSNR Y Anchor Results.

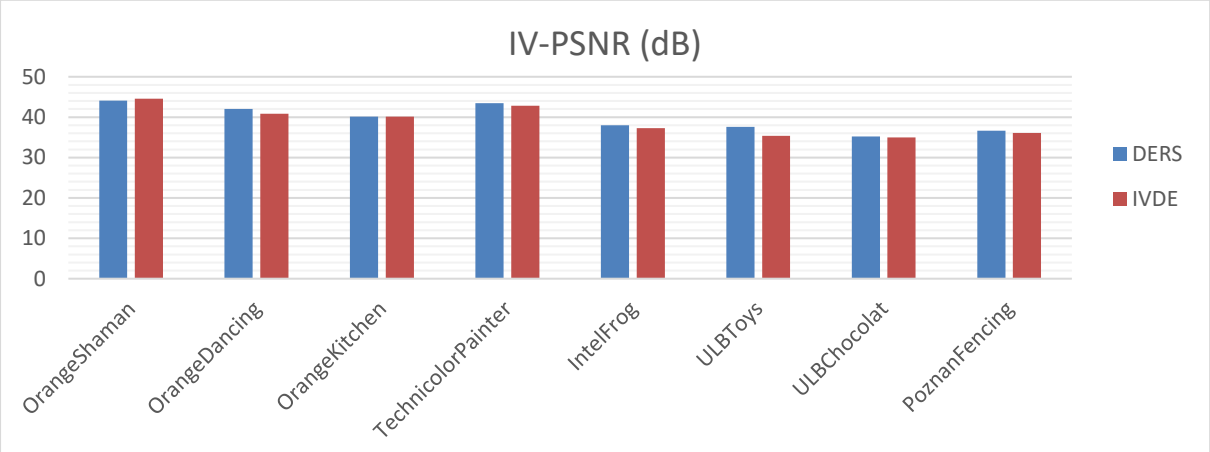

Figure 6. IV-PSNR Anchor Results.

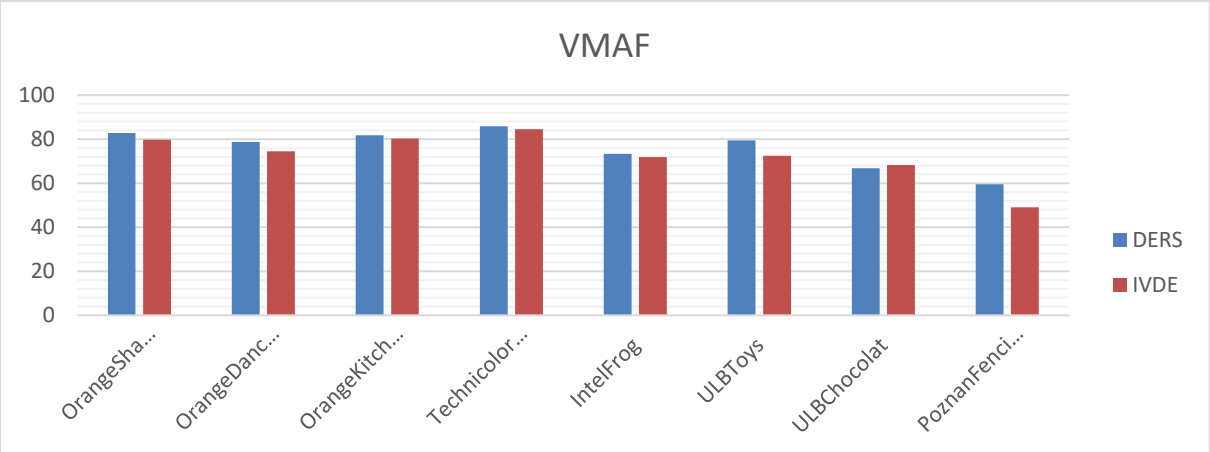

Figure 7. VMAF anchor results.

The objective quality results show that, in general, DERS obtains slightly better results than IVDE. The average results can be shown in Table 3, where the same tendency is observed.

Table 3. Average anchor results.

Average	WS-PSNR Y	IV-PSNR	VMAF
DERS	31.32 dB	39.64 dB	76.02
IVDE	30.12 dB	39.00 dB	72.60

4.2 Subjective quality results

In order to show the subjective differences between both applications, the detail of two sequences (Orange Kitchen and ULB Toys Table) are included. They can be seen in Figure 8 and Figure 9, respectively. First row is DERS and second row is IVDE. The first column is the original sequence, the second column the depth map, and the third column the synthesized view

Figure 8. Orange Kitchen detail. The first row is DERS and the second row is IVDE. The first column is the original sequence, the second column the depth map, and the third column the synthesized view.

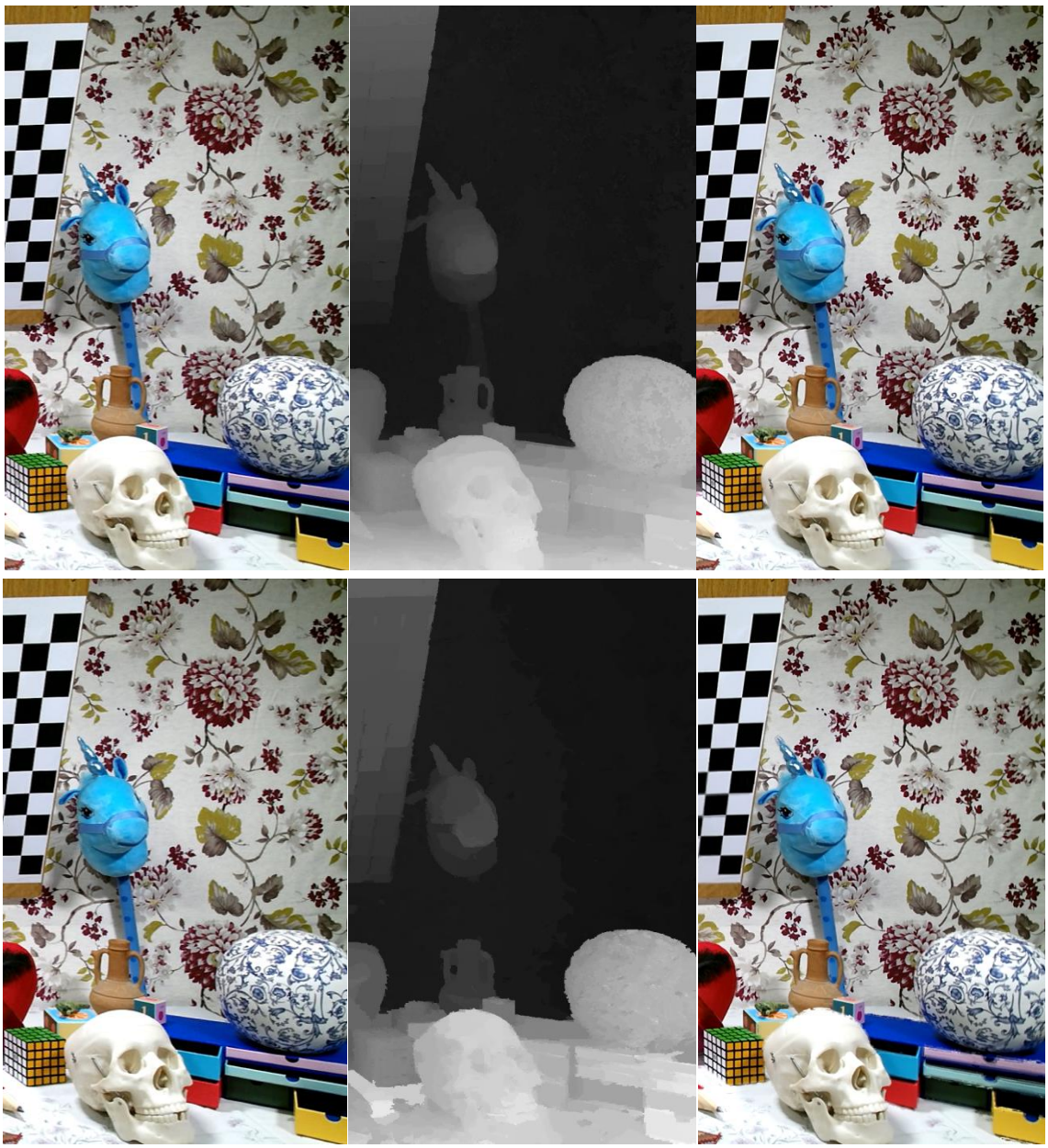

Figure 9. ULB Toys Table detail. The first row is DERS and the second row is IVDE. The first column is the original sequence, the second column the depth map, and the third column the synthesized view.

4.3 Processing time results

Processing time results can be observed in Figure 10 and Figure 11. While Figure 10 presents the average time per view, that is an MPEG-I metric, Figure 11 depicts a new metric, called global delay. This metric measures the wall time since all the view computations begin, and until the last one of them finishes. In this way, the real time for obtaining all the depth maps in a dataset is obtained, considering the possible parallelism at view level. It is important to notice that these times are always within a view and for all its frames; this means that times for every sequence except ULB Unicorn A, which is not video, comprises 17 frames.

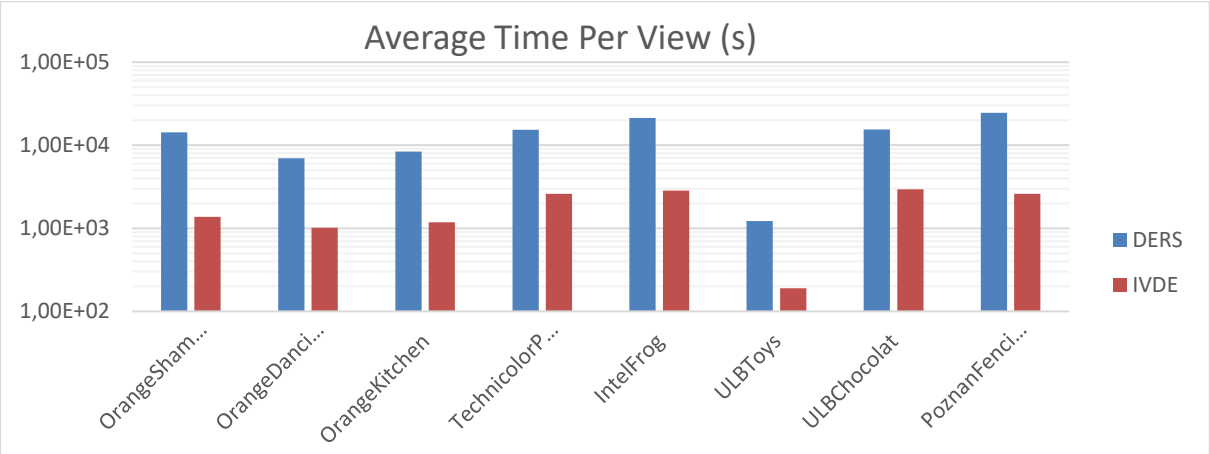

Figure 10. Average time per view anchor results.

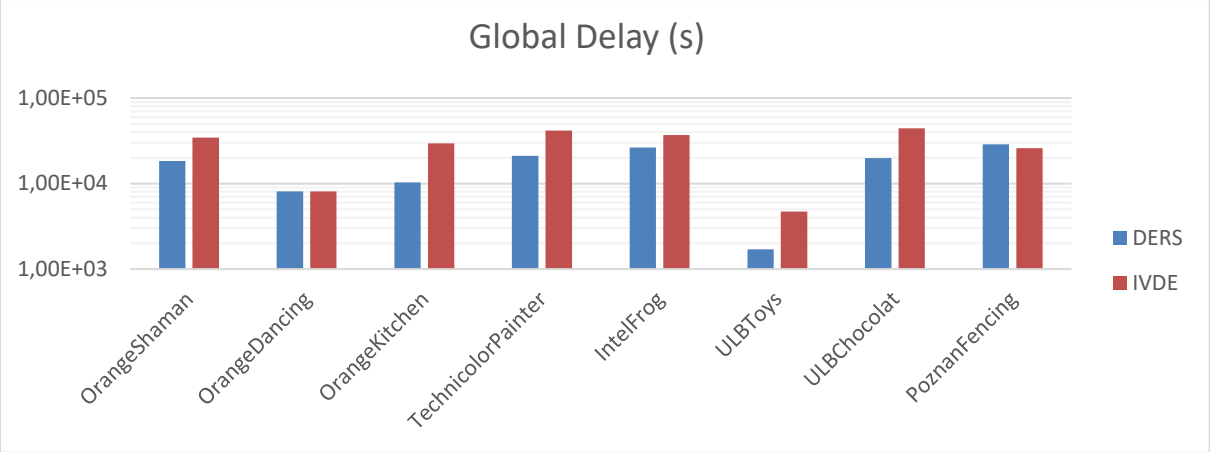

Figure 11. Global delay anchor results.

As can be seen, the average time per view penalizes DERS, which obtains in general one order of magnitude larger times. This is due to the fact that DERS is a view-output oriented application, hence the application parallelism is not explicitly expressed. In contrast, IVDE is a dataset-output oriented application, that is internally parallelized. For this reason, the average time per view presented in IVDE is the result of dividing the whole application time by the number of cameras in the dataset.

In the global delay case, this large difference is not observed, even obtaining better results in DERS. This is the consequence of using a large parallelism in DERS, which was masked in the previous metric. For obtaining the results, the same number of views in the dataset was calculated in parallel using DERS, meaning that the global delay is the maximum value obtained for all the views. For IVDE, all the datasets counted with 10 cores to parallelize the application, and the global delay is the processing time of it.

These results show that, the processing time needed in a scenario where every view can be processed by an independent core, is in general lower for DERS. In contrast, when not all the views are assigned to independent cores, IVDE clearly obtains better results.

4.4 Profiling

In addition to the processing time used by the whole application, some tests were performed to determine which is the percentage of time consumed in every stage. Although it is difficult to accurately profile the application due to the differences depending on the content and the huge number of sequences, views and frames, a brief profiling of DERS is included in Table 4. It includes two views for three different sequences.

Table 4. DERS Stage profiling.

Sequence	View	Read Cameras & Interpolation	Reliability & Smoothing Map	Motion Flag	Homography	Graph Cuts
OKitchen	v0	8.57%	0.12%	0.19%	52.85%	38.26%
OKitchen	v10	6.7%	0.1%	0.2%	50.9%	42.1%
TPainter	v0	3.8%	0.1%	0.2%	44.1%	51.9%
TPainter	v10	4.9%	0.1%	0.1%	58.8%	36.1%
IFrog	v0	2.6%	0.1%	0.2%	49.7%	47.4%
IFrog	V10	3.7%	0.0%	0.1%	63.9%	32.3%

This profiling shows that the Homography and Graph Cuts stages are the bottlenecks of DERS. In addition, it is important to remark that Graph Cuts is an algorithm that depends on the content, hence producing the variation of percentages for each view in the table above.

For IVDE, a similar profiling is included in Table 5. In this case, the difficulties of profiling arise when the application spawns new threads. For this reason, inter-intra cost and graph cuts are grouped in the same stage. As expected, this is almost the 100% percent of the time in the process, as it happens in DERS for the Homography and Graph Cuts stages joint.

Table 5. IVDE Stage profiling.

Sequence	Read Cameras & SuperPixels & Temporal Consistency	Inter/Intra & Graph Cuts	Merge Depth Maps	Neighbors Segment Analysis
OKitchen	1.0%	93.5%	4.9%	0.7%
TPainter	0.8%	93.8%	4.7%	0.7%
IFrog	2.0%	90.6%	5.8%	1.6%

5 Camera Setup Experiment

5.1 Experiment definition

As stated in the introduction, this document intends to start with the specification of the HoviTron test conditions, beginning with the determination of the setup employed. To do so, this experiment shows the behaviour in terms of quality and time for different simulated setups. They are the result of reducing the number of cameras for four of the MPEG-I sequences (Orange Shaman, Orange Kitchen, Technicolor Painter and ULB Chocolat), going from a 2x2 setup, to the original setup.

5.2 Results

The results for every camera in DERS and IVDE are presented in the following charts. First, the PSNR-Y obtained in the different simulated setups (and the full setup), for every sequence, and then, the equivalent results for processing time. All the charts show surfaces, where *x* and *y* are the position of the camera in the real setup and the color represents the value of quality or time. The color scale is fixed for every setup in a sequence, helping to contrast differences between setups.

5.2.1 Quality in DERS

DERS PSNR-Y charts for Orange Shaman, Orange Kitchen, Technicolor Painter and ULB Chocolat are depicted in Figure 12, Figure 13, Figure 14 and Figure 15, respectively.

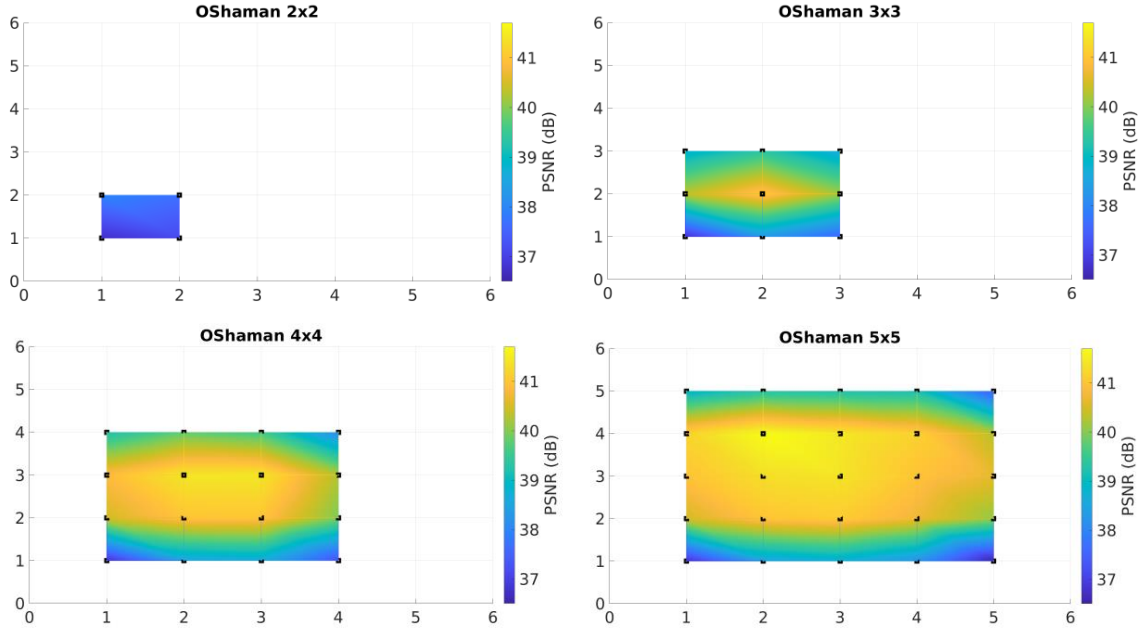

Figure 12. DERS OrangeShaman PSNR-Y for every setup.

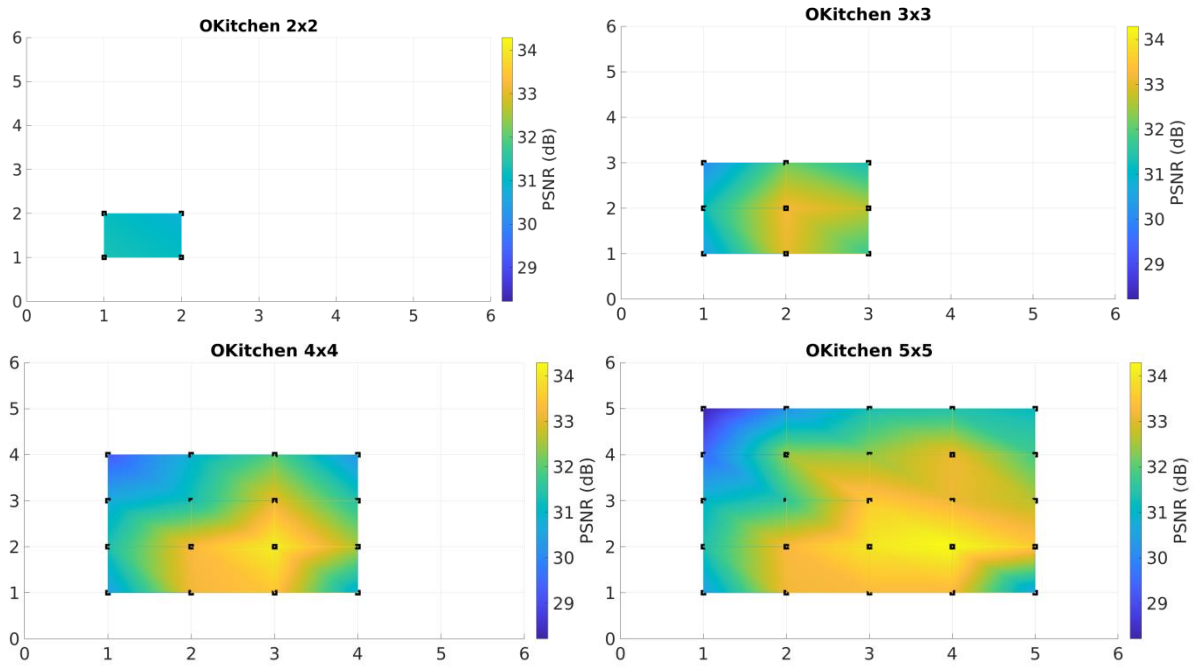

Figure 13. DERS Orange Kitchen PSNR-Y for every setup.

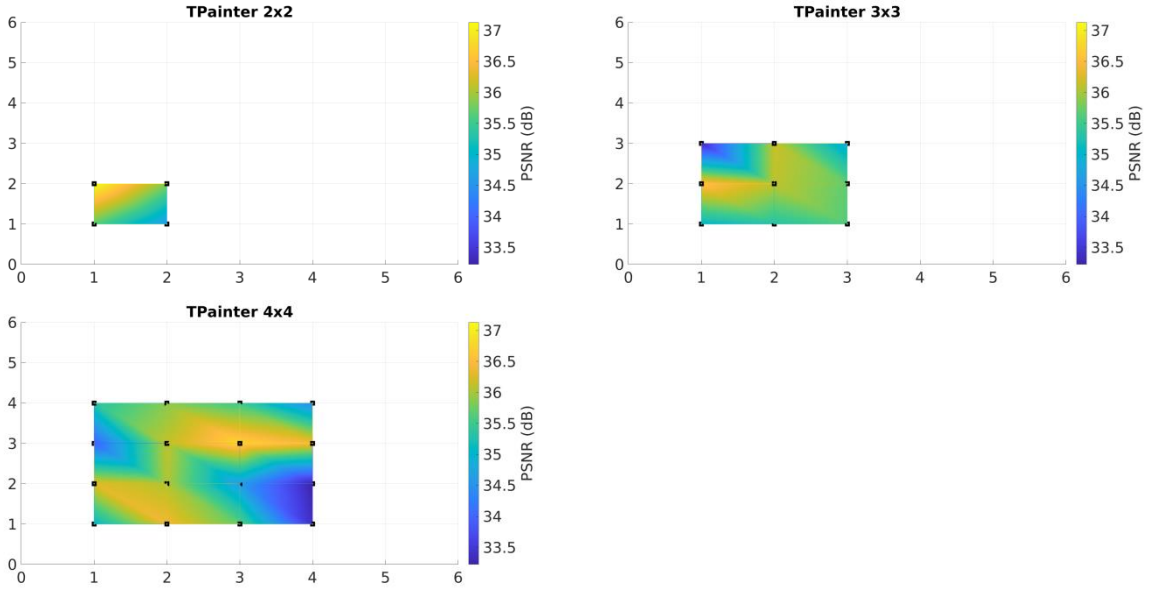

Figure 14. DERS Technicolor Painter PSNR-Y for every setup.

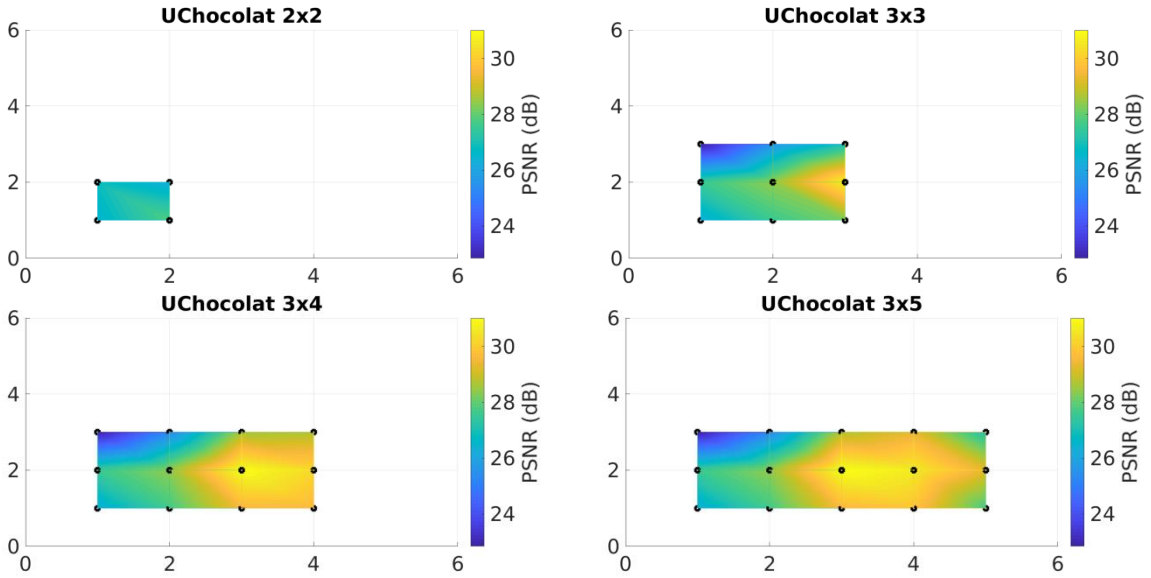

Figure 15. DERS ULB Chocolat PSNR-Y for every setup.

5.2.2 Processing time in DERS

DERS Processing times in the same order are depicted in Figure 16, Figure 17, Figure 18 and Figure 19 respectively.

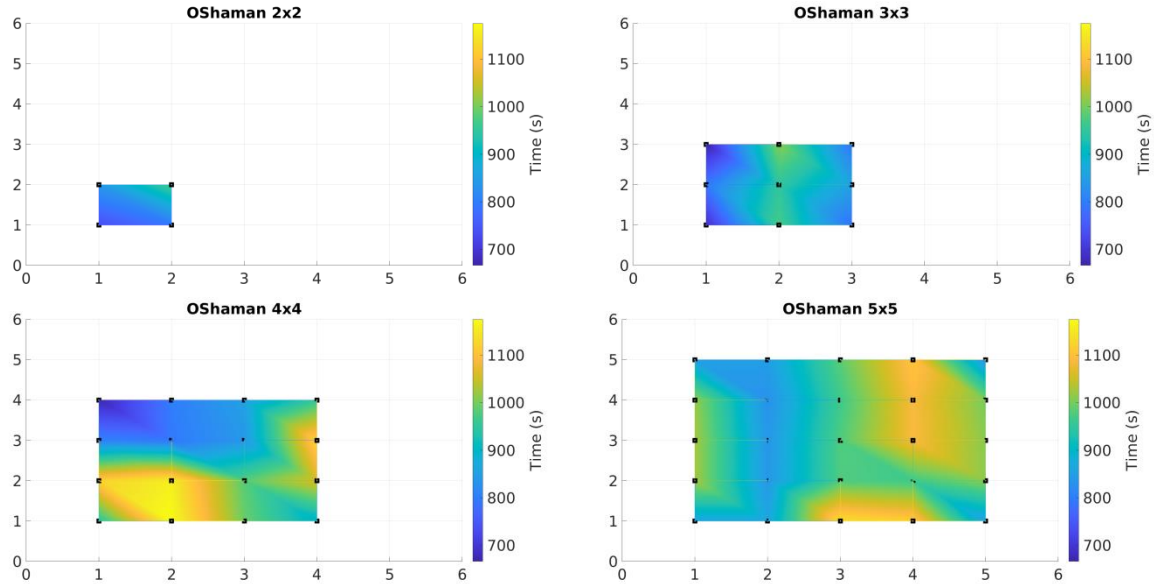

Figure 16. DERS Orange Shaman processing time for every setup.

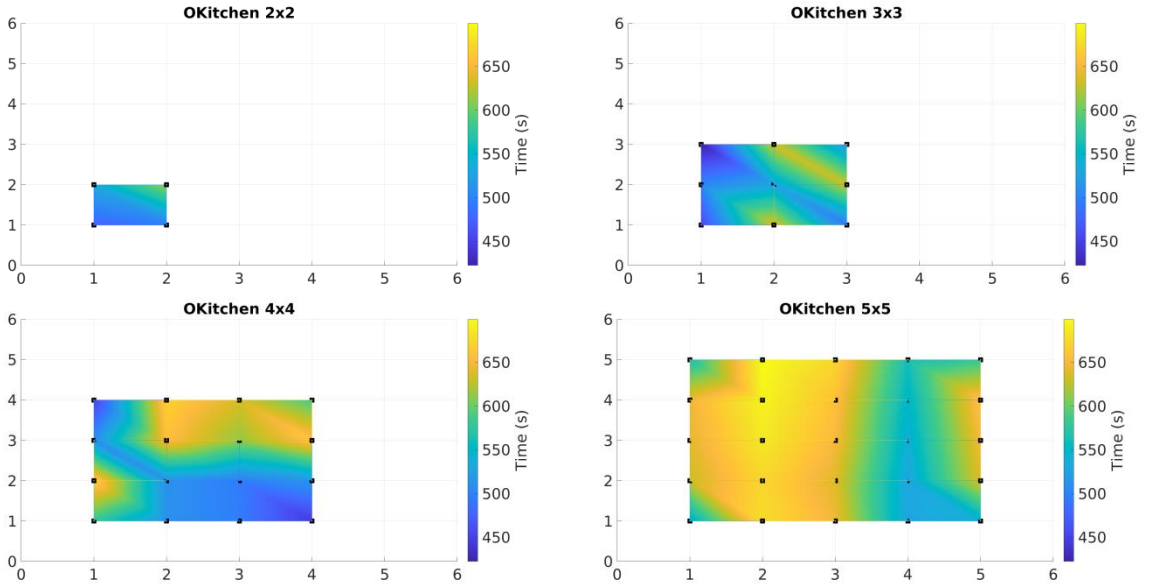

Figure 17. DERS Orange Kitchen processing time for every setup.

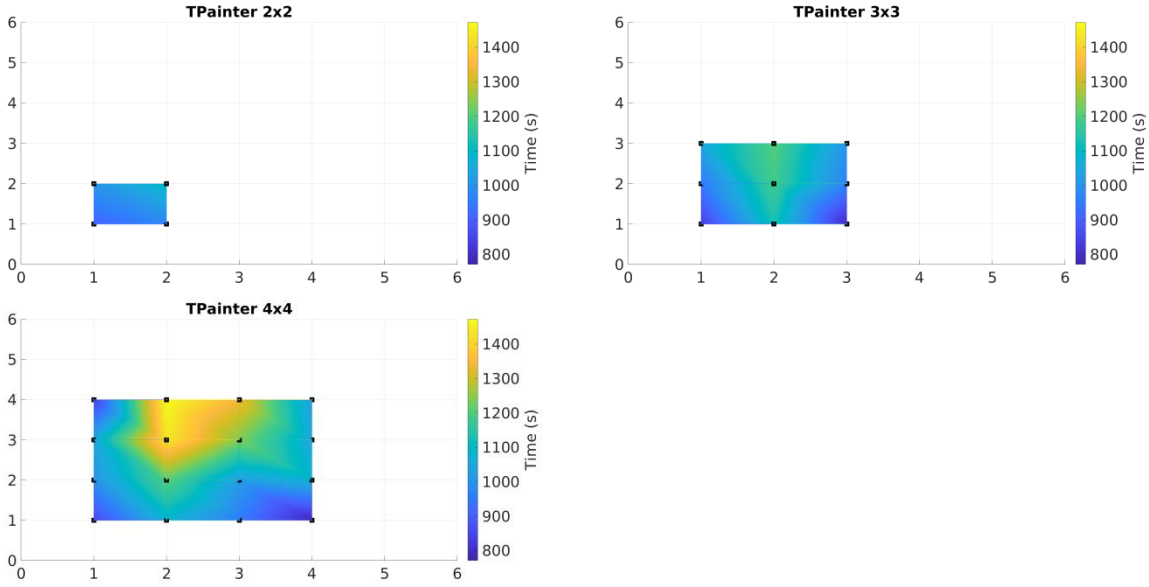

Figure 18. DERS Technicolor Painter processing time for every setup.

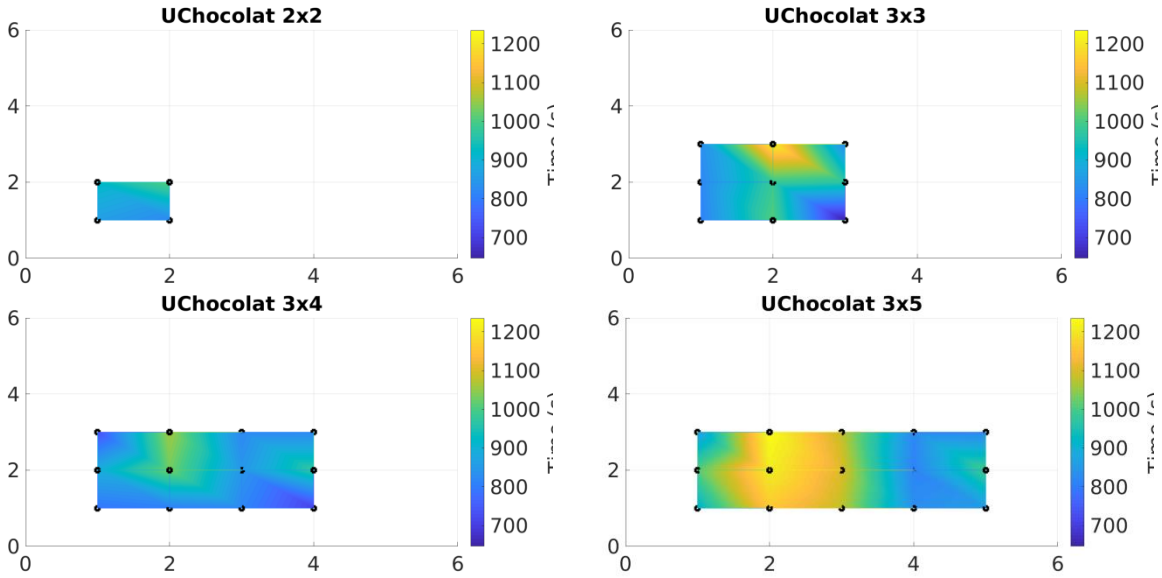

Figure 19. DERS ULB Chocolat processing time for every setup.

5.2.3 Quality in IVDE

The results for IVDE are presented in a similar way, however, only for PSNR-Y values, as it is not possible to measure individual camera processing times.

PSNR-Y charts for Orange Shaman, Orange Kitchen, Technicolor Painter and ULB Chocolat are depicted in Figure 20, Figure 21, Figure 22 and Figure 23, respectively.

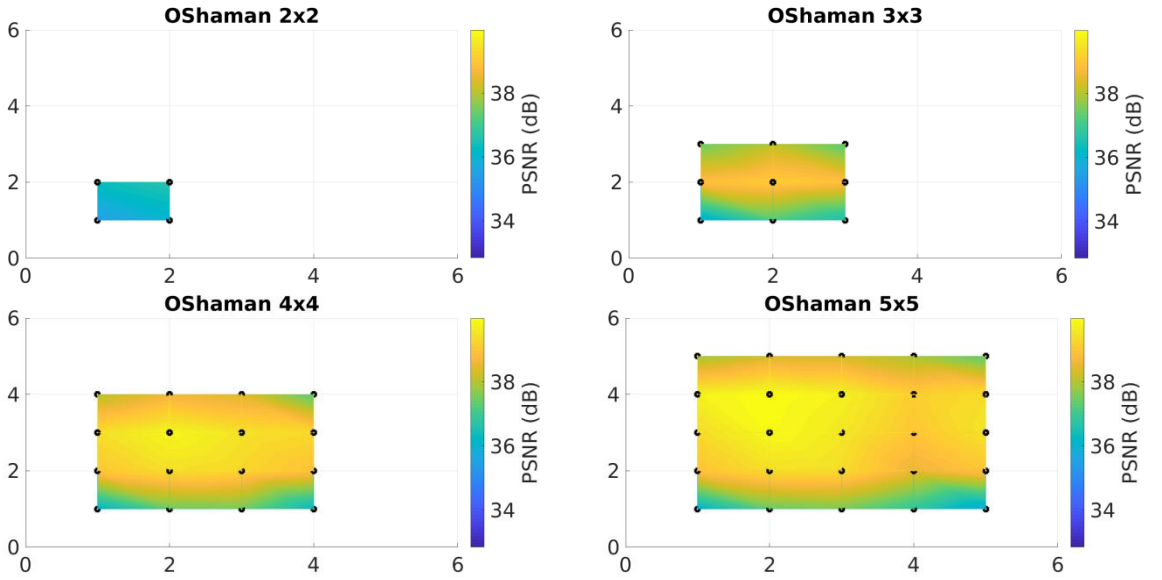

Figure 20. IVDE Orange Shaman PSNR-Y for every setup.

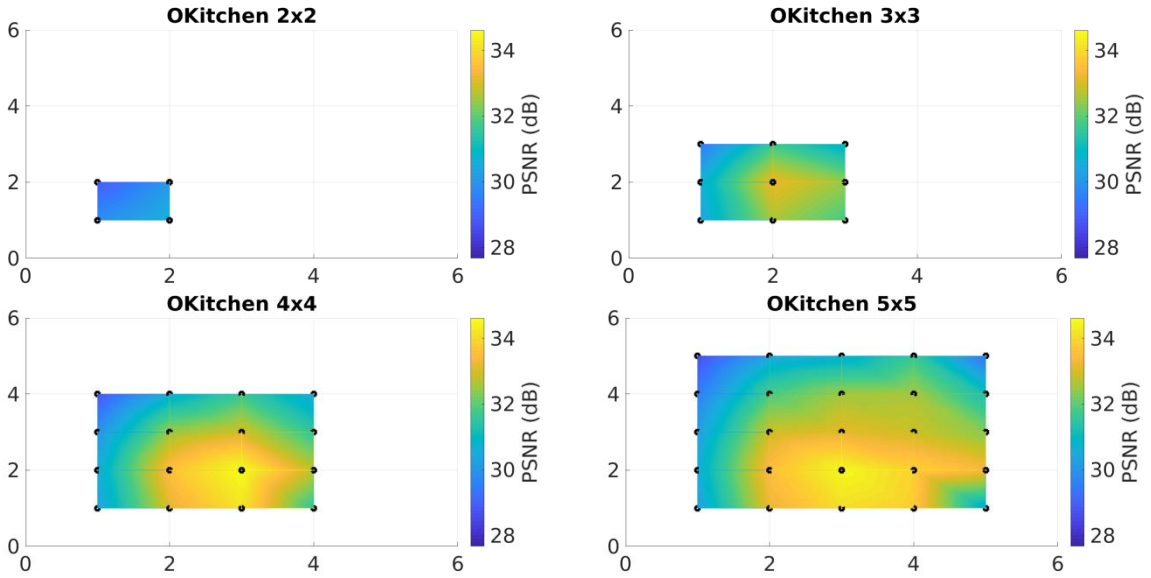

Figure 21. IVDE Orange Kitchen PSNR-Y for every setup.

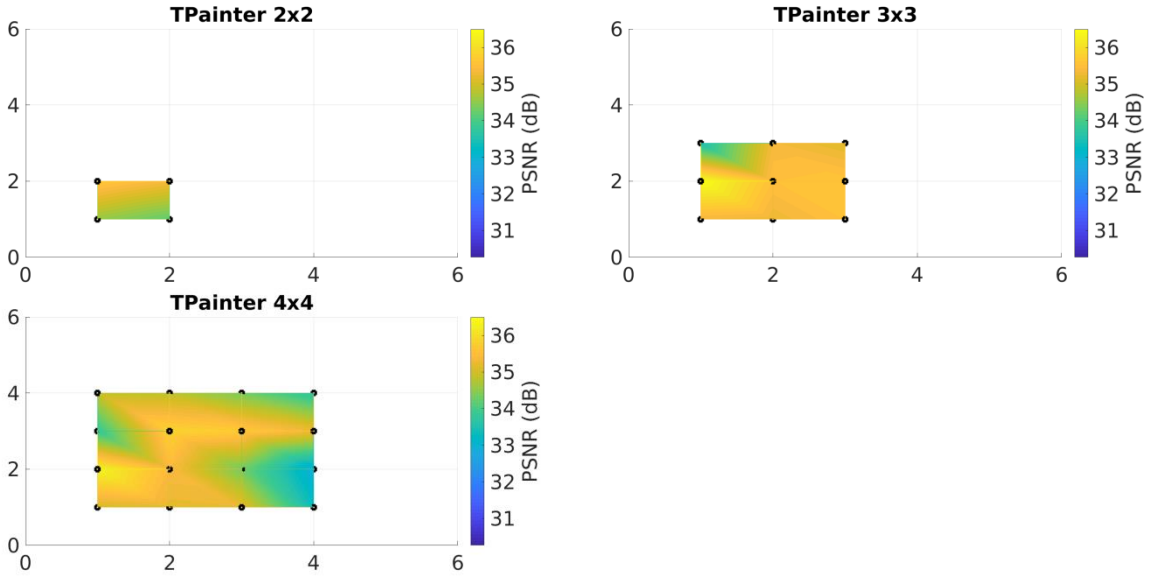

Figure 22. IVDE Technicolor Painter PSNR-Y for every setup.

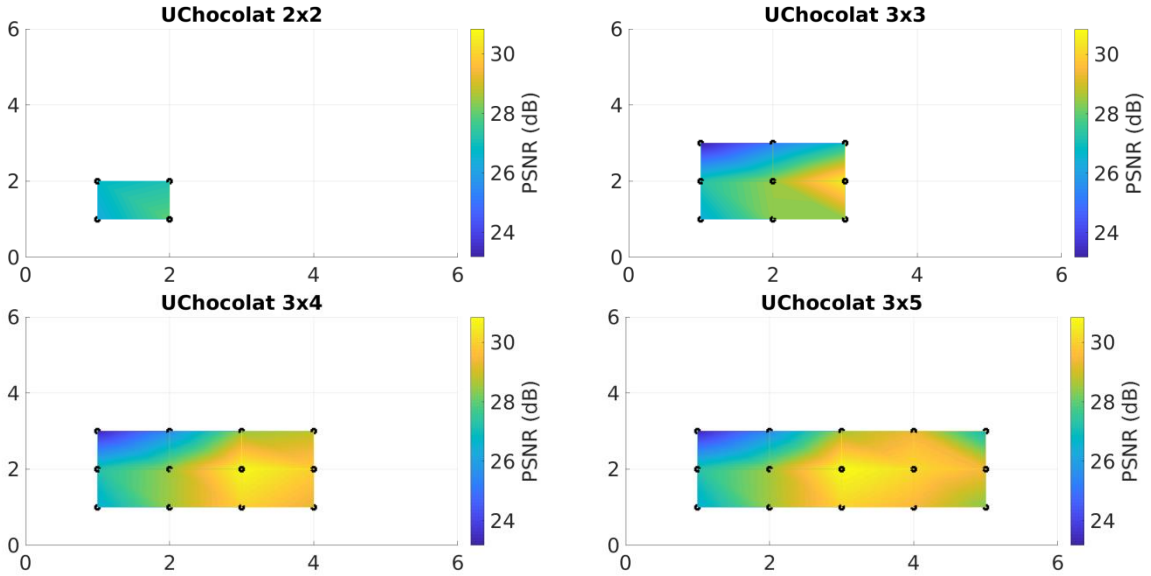

Figure 23. IVDE ULB Chocolat PSNR-Y for every setup.

5.3 Discussion

In the first place, for the PSNR-Y values and using DERS, it is clear that as the setup grows, in general, quality also increases. This is due to the addition of new points of view that help to generate the depth estimation and the synthesis. However, this growing only occurs for views where new near cameras are added; quality on corners and borders of the setup rarely increases.

Another important fact is the maximum PSNR-Y obtained for every setup, which is summarized in Table 6. As it can be seen, the maximum value is almost obtained in a setup of 3x3, with a slight variation of PSNR-Y for larger setups.

Table 6. DERS maximum PSNR-Y for setup.

Sequence	2x2	3x3	4x4[1]	5x5[2]
OShaman	37.9536	41.0369	41.4213	41.7030
OKitchen	31.3258	33.2119	34.1138	34.2954
TPainter	36.3304	36.5759	36.7475	-
UChocolat	27.8012	30.7380	31.0091	31.0130

This is also observed for IVDE, however, the differences between borders and central cameras are less pronounced, due to its inter-view consistency. It can be seen in Table 7.

Table 7. IVDE maximum PSNR-Y for setup.

Sequence	2x2	3x3	4x4[1]	5x5[2]
OShaman	36.6482	39.132	39.8419	39.9869
OKitchen	30.7075	33.2155	34.6067	34.5065
TPainter	35.675	36.4936	36.3297	-
UChocolat	27.999	30.5953	30.7739	30.8434

An additional experiment was conducted due to the results obtained in the tables above. As the processing time depends on the number of cameras, in HoviTron it is expected to use the smallest setup with the better quality. This leads to the 3x3 setup; however, it may be possible that results for 2x2 setups are masked as there is not a camera in the center of the array and hence, its quality cannot be measured. For this reason, a special simulated setup was considered: starting from a 3x3 setup, only the 4 corner cameras are considered to produce the depth estimation and the video synthesis. In this scenario, the central camera is synthesized and compared with the actual central camera of the 3x3 setup, obtaining an objective quality measurement for the central camera of the 2x2 setup.

This is depicted in Figure 24 and Figure 25 for DERS and IVDE, respectively, and using the same scale as in the previous charts (minimum and maximum for all the setups). In this way, it is possible to realize that the quality in the central camera is similar to the ones in the corners, not following the same tendency found in 3x3 setups.

[1] For UChocolat, the setup is 3x4, as the original setup is 3x5.

[2] For UChocolat, the setup is 3x5, as the original setup is 3x5. For TPainter, there is not 5x5 setup as the original setup is 4x4.

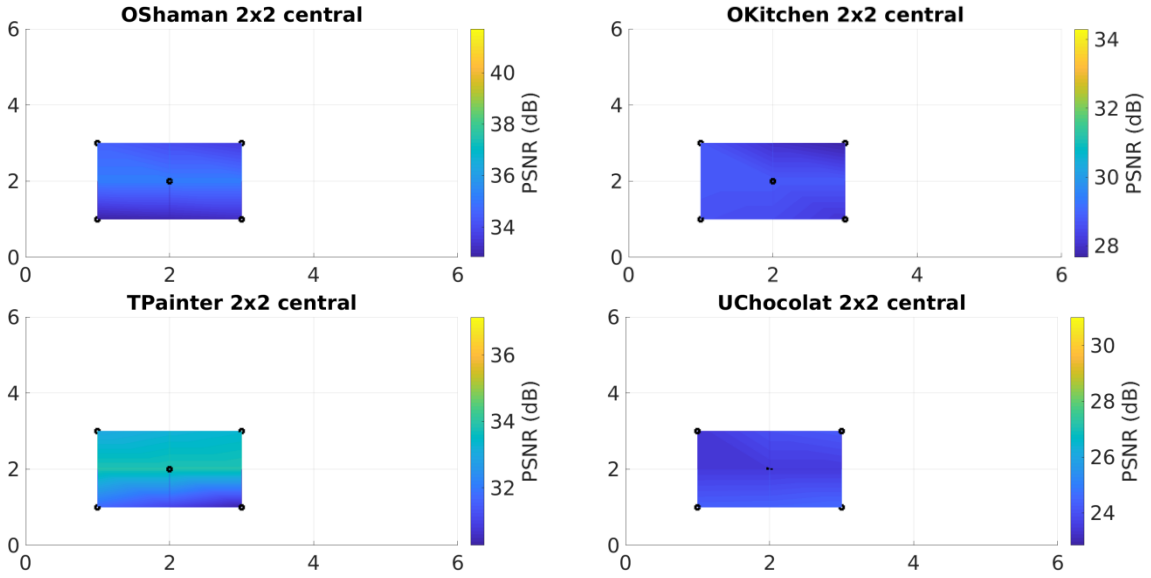

Figure 24. DERS special setup PSNR-Y.

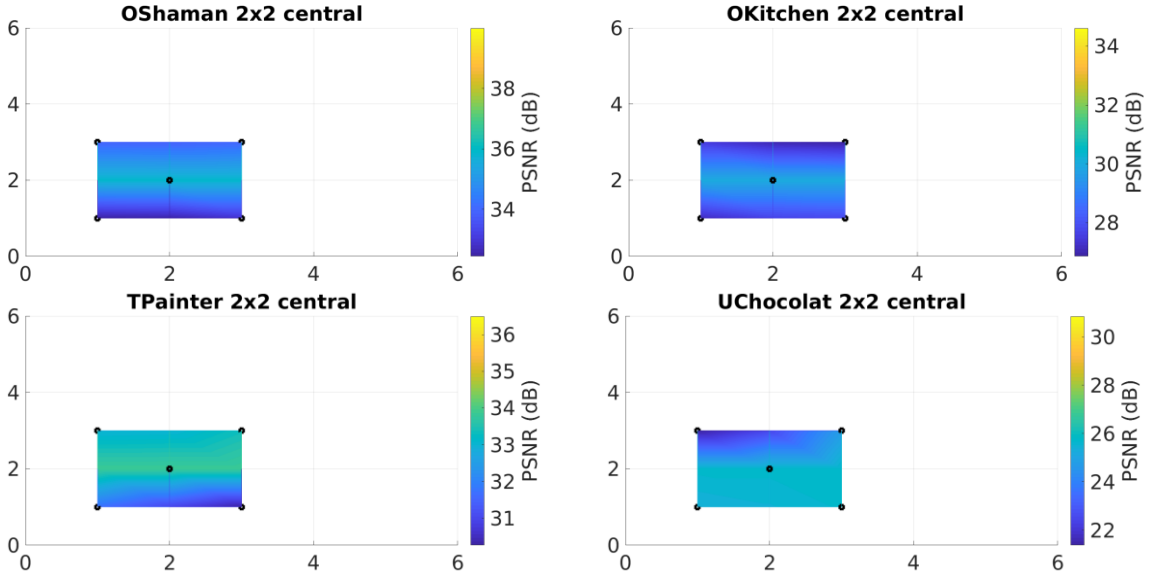

Figure 25. IVDE special setup PSNR-Y.

Regarding processing times, the general rule is that the processing time will follow a linear function with the number of cameras (although it can be masked in a parallel architecture for DERS). However, the time per camera is, at least in DERS, not constant. As depicted in the previous charts, it oscillates around a 40%, and it completely depends on the content of the camera. Although some borders or corners use less points of view, their processing time is not necessarily lower. This is due to Graph Cuts, which is independent to the number of neighbor cameras and it is a convergent process that depends on the content.

6 Acceleration prospects and planning

The previous study allows to plan the acceleration procedure that would achieve the real time constraint set in HoviTron. As explained, HoviTron application will find its basis on DERS, IVDE or a combination of both, therefore these applications will be used as a higher bound of quality and processing time.

As these processes are intended for processing images, the first accelerator architecture to think on is a Graphics Processing Units (GPU). Using this architecture presumably would obtain faster results relying on massive parallelism, which normally addresses independent processing element to different pixels.

In this regard, members of HoviTron have conducted experiments accelerating DERS, generating an application denominated GoRG [GoRG]. This application achieves an acceleration of approximately x25 times at a cost of, in average, 1.6 dB in PSNR-Y quality. This acceleration moves the average processing time of one frame in DERS from around 500 seconds to 20 seconds, considering the platform used in these experiments. However, preliminary tests performed with a newer platform (NVIDIA RTX 3090), revealed that these times can be decreased to the half, approximately.

The aforementioned work relies on a state of the art library to process Graph Cuts, CUDA Cuts [CUDA Cuts], which introduces quality losses and is the bottleneck of GoRG, with more than 95% of processing time. For these reasons, using a better implementation of Graph Cuts, for example JF-Cuts [JF-Cuts], is expected to decrease both the quality losses (completely) and the processing times (in a factor of around x10, according to its work).

Another acceleration strategy comes from one of the ideas in IVDE. If pixels are grouped into superpixels, the number of calculations can be decreased severely, at the cost of some quality. If 150.000 super pixels are considered, which is the parameter used in IVDE for the experiments in this work, the size of a full HD image (1920x1080) can be decreased in a factor of around 14; presumably achieving another magnitude order of speed-up.

Finally, as introduced by Senoh et. al. [m54255], it is possible to divide Graph Cuts into smaller blocks and then fuse these segments with a small loss of quality, processing faster and in a parallel way. This acceleration is also expected to improve the results drastically, as Graph Cuts is the main bottleneck in the GPU accelerated version.

7 Conclusions

This document describes the applications that will be the basis of the HoviTron software: DERS and IVDE, analyzing their stages and providing anchor results for the experiments. In addition, it includes results for simulated setups that will be considered when defining the HoviTron setup. Finally, a brief discussion on the acceleration of the tools is given.

In the first place, when analyzing DERS and IVDE, it is observed that these applications share a common structure, where the images are first compared to other views to obtain an unrefined cost cube, and then, Graph Cuts is used to produce the final depth map. The main functional differences are that, unlike DERS, (i) IVDE does not consider singular pixels, but superpixels and (ii) IVDE performs an adjustment based on the coherency inter-view. In addition, IVDE uses different threads to parallelize Graph Cuts. Results show that objective quality is similar for both tools, although slightly worse in IVDE. In the case of processing time, IVDE achieves better processing times if the number of cores employed to accelerate is lower than the number of views.

In the second place, simulated setup results showed that setups with only 2x2 cameras achieve substantially worse results than larger setups. As the quality increment when moving from 3x3 to a larger setup is not remarkable, it is recommended to employ a 3x3 setup in HoviTron. This setup may have included the plenoptic camera in the central position.

Finally, in terms of acceleration, it is clear that the analyzed tools are far away from real time processing, with times in the order of hundred of seconds. However, its parallelization in a GPU following different strategies seems promising.

8 References

[DERS] Segolene Rogge, Daniele Bonatto, Jaime Sancho, Ruben Salvador, Eduardo Juarez, Adrian Munteanu and Gauthier Lafruit. “MPEG-I Depth Estimation Reference Software,” 2019 International Conference on 3D Immersion (IC3D), Brussels, Belgium, 2019, pp. 1-6, doi: 10.1109/IC3D48390.2019.8975995.

[w18172] P. Boissonade, J. Jung. “Versatile View Synthesizer 2.0 (VVS 2.0) manual [w18172]” ISO/IEC JTC1/SC29/WG11, Jan. 2019.

[w17759] Bart Kroon, Gauthier Lafruit. “Reference View Synthesizer (RVS) 2.0 manual [w17759]” ISO/IEC JTC1/SC29/WG11, July 2018.

[w19221] “Exploration Experiments on Processing for Immersive Video [w19221]” ISO/IEC JTC1/SC29/WG11, May 2020.

[w18069] “WS-PSNR Software Manual [w8069]” ISO/IEC JTC1/SC29/WG11, October 2018.

[w18709] Adrian Dziembowski. “Software manual of IV-PSNR for Immersive Video [w18709]” ISO/IEC JTC1/SC29/WG11, July 2019.

[IVDE] D. Mieloch, O. Stankiewicz and M. Domański, "Depth Map Estimation for Free-Viewpoint Television and Virtual Navigation," in IEEE Access, vol. 8, pp. 5760-5776, 2020, doi: 10.1109/ACCESS.2019.2963487.

[DERS-Manual] Krzysztof Wegner and Olgierd Stankiewicz (2014). “DERS Software Manual”.

[Graph-Cuts]. V. Kolmogorov and R. Zabin, "What energy functions can be minimized via graph cuts?," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 2, pp. 147-159, Feb. 2004, doi: 10.1109/TPAMI.2004.1262177.

[VMAF] Perceptual video quality assessment based on multi-method fusion, https://github.com/Netflix/vmaf

[GoRG] J. Sancho et al., "Towards GPU Accelerated HyperSpectral Depth Estimation in Medical Applications," 2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS), Segovia, Spain, 2020, pp. 1-6, doi: 10.1109/DCIS51330.2020.9268649.

[CUDA-Cuts] V. Vineet and P. J. Narayanan, "CUDA cuts: Fast graph cuts on the GPU," 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Anchorage, AK, 2008, pp. 1-8, doi: 10.1109/CVPRW.2008.4563095.

[JF-Cut] Y. Peng, L. Chen, F. Ou-Yang, W. Chen and J. Yong, "JF-Cut: A Parallel Graph Cut Approach for Large-Scale Image and Video," in IEEE Transactions on Image Processing, vol. 24, no. 2, pp. 655-666, Feb. 2015, doi: 10.1109/TIP.2014.2378060.

[m54255] Takanori Senoh, Nobuji Tetsutani, Jaime Sancho, Eduardo Juárez, Gauthier Lafruit. “Proposal of Automatic Reference Depth Estimation Software (RDE0.96) [m64255]”]” ISO/IEC JTC1/SC29/WG11, June 2020.

9 List of Annexes

[Annex A] Configuration files for DERS and IVDE.

