ISO/IEC JTC 1/SC 29/WG 11
Coding of moving pictures and audio
Convenorship: UNI (Italy)

Document type: Approved WG 11 document

Title: Manual of Immersive Video Depth Estimation

Status: Approved

Date of document: 2020-05-08

Source: Video

Expected action:

No. of pages: 12

Email of convenor: leonardo@chiariglione.org

Committee URL: mpeg.chiariglione.org

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/N19224
May 2020, Online

Source	**Video**
Status	**Approved**
Title	**Manual of Immersive Video Depth Estimation**
Editor	Dawid Mieloch

1 Introduction

Omnidirectional video formats are currently considered within MPEG in the context of 6DoF/3DoF+ video technology. This document describes the depth estimation technique and software called Immersive Video Depth Estimation (IVDE), which addresses depth estimation from video acquired by multiple omnidirectional cameras, needed to create multi-point 6DoF/3DoF+ scene representation.

2 IVDE

The framework is based mainly on the method described in [1]. 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 **and any number of arbitrarily positioned cameras (both perspective and omnidirectional)** can be used during the estimation.

- In the 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 framework uses a parallelization method that **reduces the processing time of graph-based depth estimation.**

2.1 *Depth estimation*

The estimation of depth in the proposed method is based on a cost function minimization, performed using GraphCut method [2]. 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, $d_s \in \underline{\mathrm{d}}$.

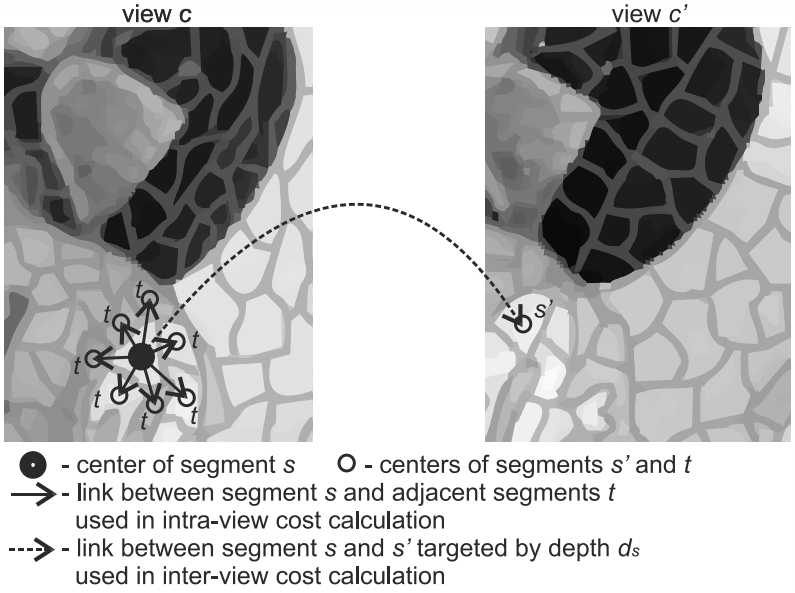

Fig. 1. Inter-view and intra-view costs.

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 $m_{s,s\prime}$, is:

$$m_{s,s\prime}(d_s) = \frac{1}{count(\mathrm{W})} \sum_{w \in \mathrm{W}} \left\| [Y C_b C_r]_{\mu_s + w} - [Y C_b C_r]_{T[\mu_s] + w} \right\|_1 ,$$

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 [2]:

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

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 is 30, as discussed in [1]. The use of both equirectangular and perspective views is included in the 3D transform $T[\cdot]$.

2.2 *Neighboring segments depth analysis*

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

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.

2.3 *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, C_b and C_r 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.

2.4 *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. 3). Only two depth maps can be merged into one by one thread during the merging cycle. Therefore, for n threads, $\lceil log2(n) \rceil$ additional cycles are needed to estimate the final depth map with all depth levels.

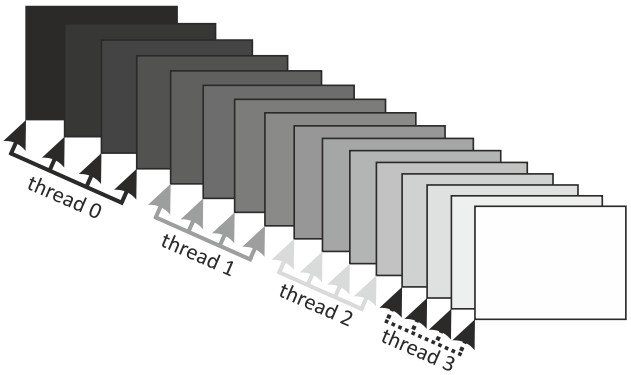

Fig. 2. Depth levels are divided into blocks, each rectangle represents a different level of the depth of a scene.

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.

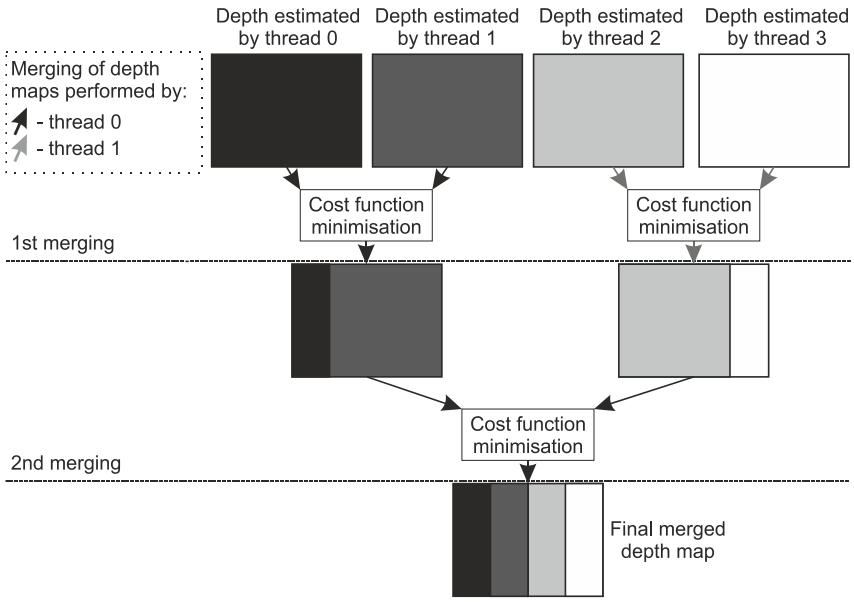

Fig. 3. Depth map merging process for the case of 4-thread parallelization.

2.5 *Segmentation in omnidirectional videos*

The use of omnidirectional cameras is taken into account during the superpixel segmentation of input views. The superpixel segmentation [3] 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.

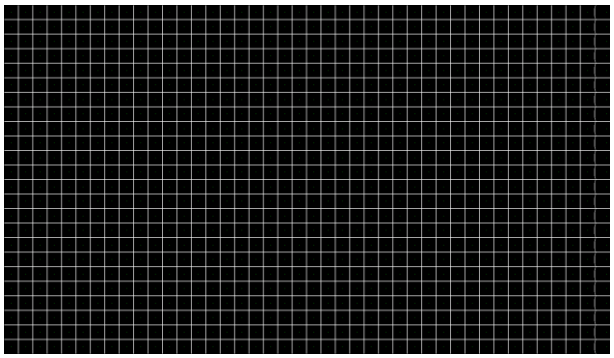

Fig. 4. Initial grid of superpixels used in segmentation.

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.

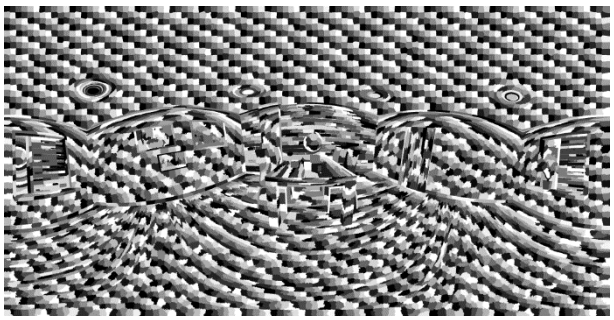

Fig. 5. Result of unmodified superpixel segmentation for an equirectangular image.

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.

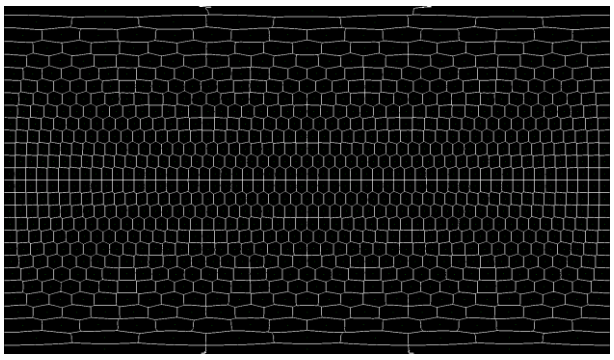

Fig. 6. Proposed initial grid of superpixels used in segmentation of an equirectangular image.

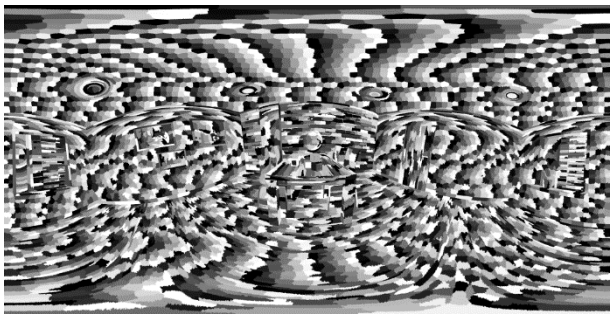

Fig. 7. Result of modified superpixel segmentation for an equirectangular image.

3 Building the project

The IVDE framework does not use any external libraries for image processing operations. The project can be built for Windows and Linux using CMake.

In order to build the project for Visual Studio, open the command line and go to the folder that contains the `CMakeLists.txt` file. If you installed CMake, you can build the project for the x64 architecture using the following command: `cmake .\ -A x64 -B build`. The project will be built to the build folder.

4 Configuration file

This section provides information on configuration of IVDE. Below, the example of configuration file for PoznanFencing test sequence is enclosed. Table 1 includes description of all parameters.

```
#=============== INPUT PARAMETERS ===============
NumOfThreads                        2
StartFrame                          0
TotalNumberOfFrames                 8

NeighboringSegmentsDepthAnalysis    1

FileCameraParameter                 cam_params.txt
ChrominanceFormat                   420
NearestZValue                       3.5
FarthestZValue                      7
NumberOfZSteps                      256
MatchNeighbors                      4
MatchThresh                         30
Matcher                             Block
MatchingBlockSize                   1
SmoothingCoefficient                1
NumberOfCycles                      1

NameOfCamera0                       param_cam0
InputView0                          PoznanFencing_1920x1080_cam0.yuv
ViewType0                           Perspective
ViewWidth0                          1920
ViewHeight0                         1080
ViewNumOfSuperpixels0               150000
OutputDepthMap0                     pf_depth_1920x1080_cf400_16bps_cam0.yuv

NameOfCamera1                       param_cam1
InputView1                          PoznanFencing_1920x1080_cam1.yuv
```

```
ViewType1                           Perspective
ViewWidth1                          1920
ViewHeight1                         1080
ViewNumOfSuperpixels1               150000
OutputDepthMap1                     pf_depth_1920x1080_cf400_16bps_cam1.yuv

NameOfCamera2                       param_cam2
InputView2                          PoznanFencing_1920x1080_cam2.yuv
ViewType2                           Perspective
ViewWidth2                          1920
ViewHeight2                         1080
ViewNumOfSuperpixels2               150000
OutputDepthMap2                     pf_depth_1920x1080_cf400_16bps_cam2.yuv

NameOfCamera3                       param_cam3
InputView3                          PoznanFencing_1920x1080_cam3.yuv
ViewType3                           Perspective
ViewWidth3                          1920
ViewHeight3                         1080
ViewNumOfSuperpixels3               150000
OutputDepthMap3                     pf_depth_1920x1080_cf400_16bps_cam3.yuv

NameOfCamera4                       param_cam4
InputView4                          PoznanFencing_1920x1080_cam4.yuv
ViewType4                           Perspective
ViewWidth4                          1920
ViewHeight4                         1080
ViewNumOfSuperpixels4               150000
OutputDepthMap4                     pf_depth_1920x1080_cf400_16bps_cam4.yuv

#=============== SEGMENTATION ===============

SuperpixelSegmentationType    SNIC
SuperpixelColorCoeff          20

#=============== TEMPORAL ENHANCEMENT ===============

TemporalEnhancement                         1
TemporalEnhancementIFramePeriod             8
TemporalEnhancementThresh                   0.5
NumberOfCyclesInIFrame                      1
```

Table 1. IVDE configuration parameters.

Name of the parameter	Description	Type, value
`NumOfThreads`	Number of CPU threads used by software	Unsigned int
`StartFrame`	The first frame of input views that should be used for estimation	Unsigned int
`TotalNumberOfFrames`	Number of frames	Unsigned int
`NeighboringSegmentsDepth Analysis`	Turning on/off the neighboring segments depth analysis (see section 2.2)	Unsigned int, 0 or 1

FileCameraParameter	Path and name of the file that contains VSRS-style camera parameters (included in the MPEG repository, see section 5)	String
ChrominanceFormat	Chrominance format of input views	Unsigned int, 420 or 444
NearestZValue	The nearest depth plane in the scene	Double
FarthestZValue	The farthest depth plane in the scene	Double
NumberOfZSteps	The number of depth steps between the nearest and farthest depth planes	Unsigned int
MatchNeighbors	Number of neighboring views matched with each view	Unsigned int
MatchThresh	The threshold of the inter-view matching cost	Unsigned int
Matcher	Type of used matcher	String, “Block” or “Pixel”
MatchingBlockSize	Size of block used in inter-view matching cost	Unsigned int
SmoothingCoefficient	Value of β in the intra-view discontinuity cost	Double
NumberOfCycles	Number of GraphCut cycles	Unsigned int
NameOfCamera0	Name of view in camera parameters file	String
InputView0	The filename of view number 0 (number of views is not limited), 8 bps inputs are supported only	String
ViewType0	Type of view number 0, perspective and 360 degree omnidirectional views are supported	String, “Perspective” or “Omnidirectional”
ViewWidth0	Width of view number 0	Unsigned int
ViewHeight0	Height of view number 0	Unsigned int
ViewNumOfSuperpixels0	Number of superpixels used for the estimation of the depth map of view number 0	Unsigned int
OutputDepthMap0	The filename of 16-bit, cf 4:0:0 depth map of view number 0	String
SuperpixelSegmentation Type	Type of used superpixel segmentation	String, “SNIC”
SuperpixelColorCoeff	Coefficient used in superpixel segmentation to influence shapes of superpixels, high coefficient (>20) decreases the influence of color on the shape	Double
TemporalEnhancement	Turning on/off the temporal consistency enhancement	Unsigned int, 0 or 1
TemporalEnhancement IFramePeriod	Number of frames between I-type depth frames +1	Unsigned int
TemporalEnhancement Thresh	The threshold used in the temporal consistency enhancement	Double
NumberOfCyclesInIFrame	Number of GraphCut cycles in I-type depth frame	Unsigned int

5 MPEG Repository

The repository for IVDE is available on MPEG GIT:

http://mpegx.int-evry.fr/software/MPEG/Explorations/6DoF/IVDE

In the repository, two folders can be found:

- `/src`: which contains the source code of IVDE
- `/cfgs`: which contains camera parameters and configuration parameter files per sequence. The current configuration parameter files are those presented in [4].

The `CMakeLists.txt` file for `CMake` is in the root directory.

6 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] 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.

[4] D. Mieloch, A. Dziembowski, J. Stankowski, O. Stankiewicz, M. Domański, G. Lee, J. Yun, “[MPEG-I Visual] Immersive video depth estimation”, ISO/IEC SC29/WG11 MPEG2020/M53407, Online, April 2020