INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC 1/SC 29/WG 04 MPEG VIDEO CODING

ISO/IEC JTC 1/SC 29/WG 04 m64164

July 2023, Geneva, Switzerland

Title	**[MIV] Versatile multicamera system calibration framework for immersive video applications**
Source	**PUT, ETRI**
Authors	**Dawid Mieloch, Błażej Szydełko, Adrian Dziembowski, Dominika Klóska, Jun Young Jeong, Gwangsoon Lee**

Abstract

Accurate extrinsic parameters calibration is crucial for various applications, particularly in immersive video, where camera calibration plays a significant role, as its quality is essential for accurate reconstruction and efficient compression of three-dimensional scene information. While methods for intrinsic parameters calibration, color correction, and depth estimation are publicly available, there is a lack of versatile techniques for estimating extrinsics in the context of immersive video. The proposed software addresses these limitations by proposing a versatile extrinsic parameters estimation method and a framework for testing the quality of extrinsics. The software is compatible with MPEG immersive video framework, allowing for seamless integration and evaluation. It is recommended to include this framework as a reference tool for immersive video activities.

1 Introduction

Accurate camera calibration is fundamental for various applications, including precise measurements, accurate 3D reconstructions, and reliable scene understanding [1]. It also plays a vital role in camera networks and setup systems [2]. While intrinsic parameters calibration (i.e., estimating the optical characteristics of cameras) for machine vision applications poses no issues in terms of availability of open source implementations, comparative accuracy evaluation [3], [4], the extrinsic parameters calibration (estimating their relative positioning), particularly in the context of immersive video applications, can be still very challenging.

Estimating extrinsic parameters is essential for achieving accurate 3D reconstruction from multiple views [5]. By determining the relative positions and orientations of multiple cameras, it becomes possible to triangulate corresponding image points and reconstruct the 3D structure of the scene. Therefore, calibration accuracy is crucial for depth estimation, as it relies on precise knowledge of the position and orientation of each camera within the system [5] and, as a result, on an efficient compression and transmission of scene information [6]. Therefore, without proper calibration, depth estimation and rendering of virtual views in immersive video would be compromised. Immersive video requires calibration methods that are versatile enough to accommodate a wide range of camera configurations and placements.

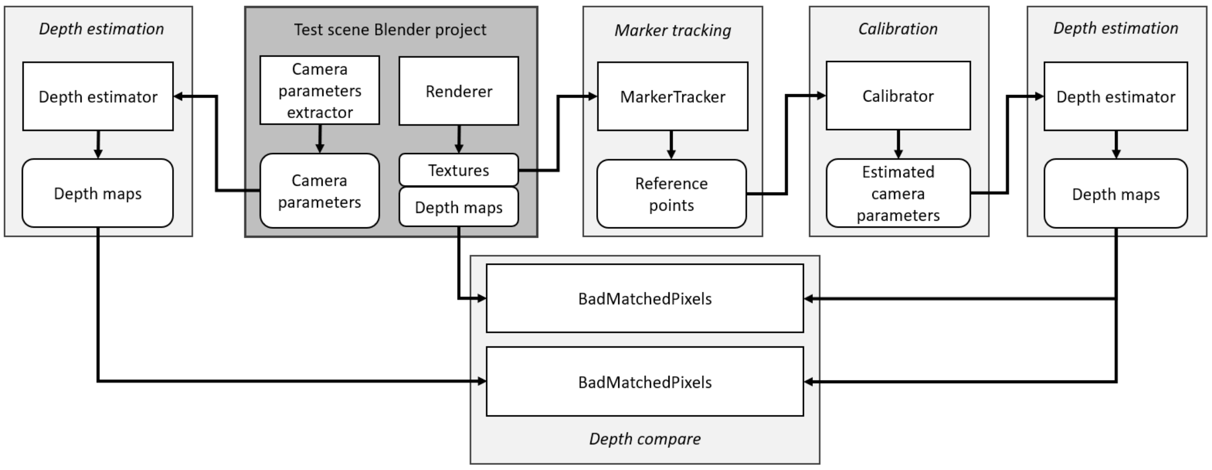
While methods for intrinsic parameters calibration, color correction [7], and depth estimation [8], [9] have already been developed and made publicly available, as well as well-established frameworks for their comparison (e.g., MIV CTC, Middlebury database [10]), the evaluation of extrinsic parameters quality lacks a standardized approach.

To address these challenges, this research proposes two main contributions. Firstly, a versatile method for estimating extrinsic parameters was developed, enabling the extrinsic parameters estimation in various camera configurations and placements. Secondly, a comprehensive framework is introduced to evaluate different methods of extrinsic parameters estimation. This framework involves simulating a virtual walk in a Blender-rendered multiview sequence, estimating parameters from the virtual sequence, comparing the estimated parameters against ground truth, evaluating the quality of estimated depth maps using both parameter sets, and synthesizing virtual views based on the estimated depth maps. The provided software is compatible with the MPEG immersive video framework, supporting JSON-based data format, omnidirectional media format (OMAF) coordinate system [11], and depth maps with normalized disparity [12], therefore, can be easily used to evaluate the extrinsic parameters in this application.

2 Software description

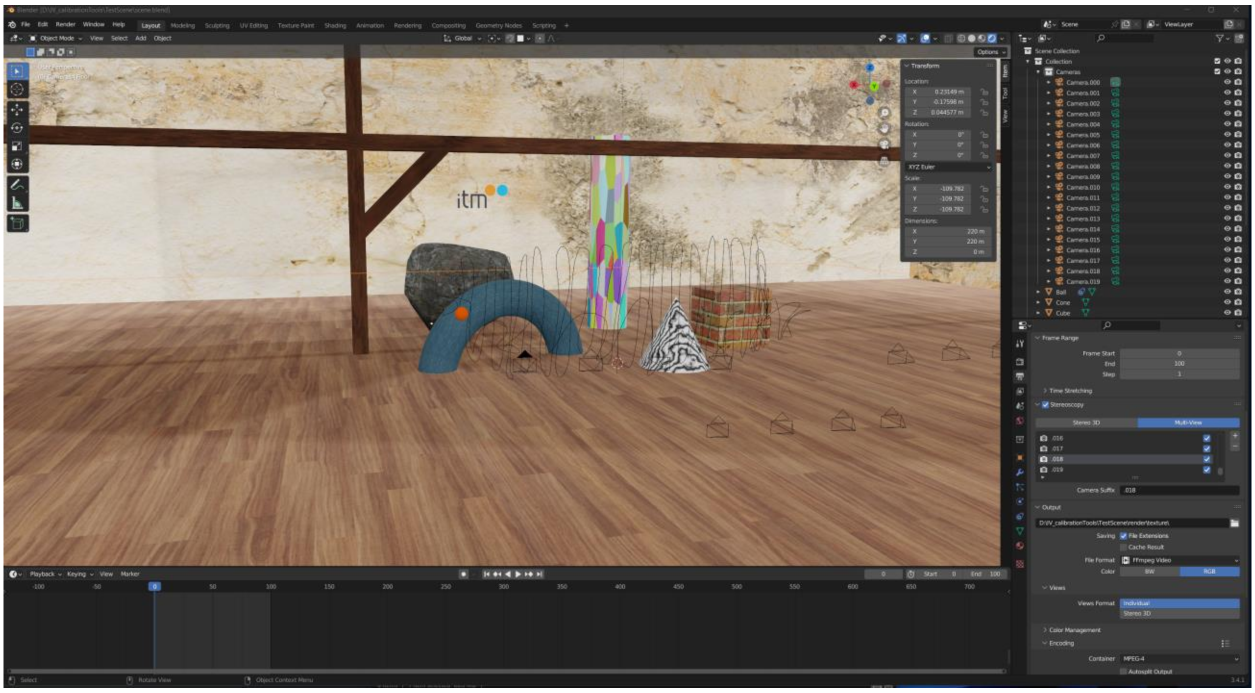
The repository with the framework: https://github.com/bszydelko/IV_calibrationTools

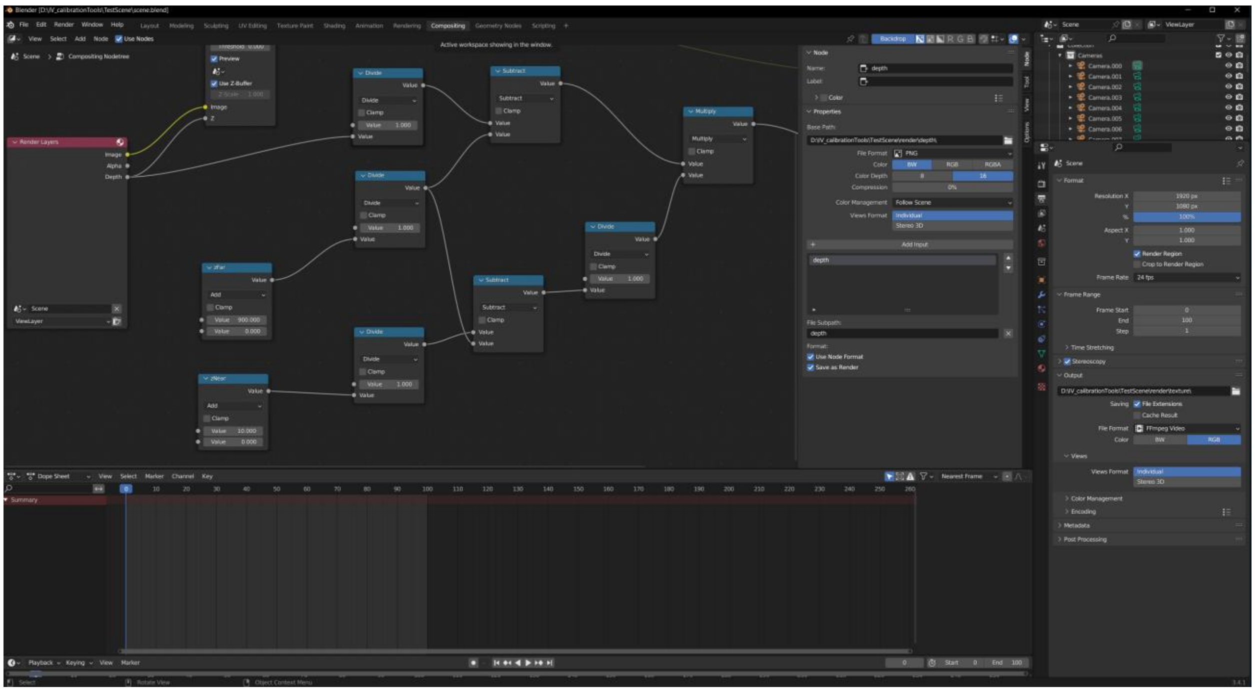
Overview



Blender project for generating of multiview calibration sequence

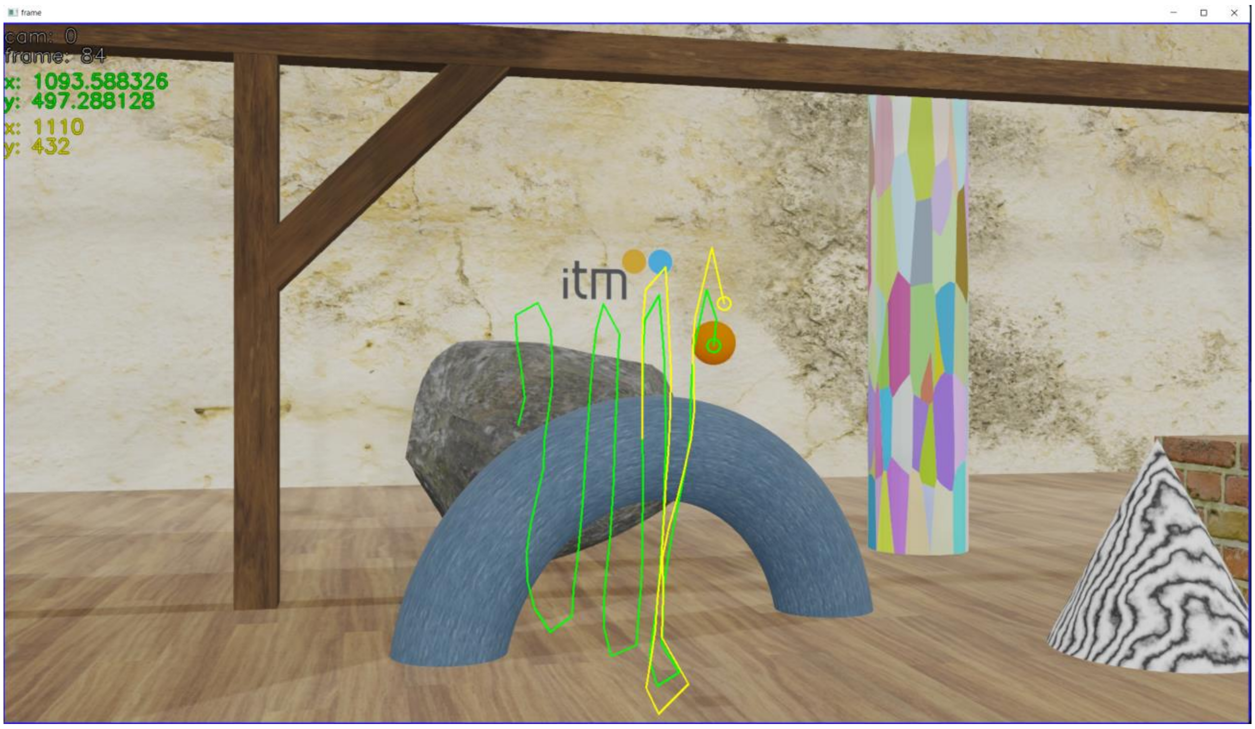
The calibration sequence used with this tool can contain any objects the user wants. Every object should be stationary except for a marker. It is possible to test this kit with any camera configuration. The default calibration sequence attached to this toolkit contains a few stationary objects and one moving object - an orange ball - which is the marker that will be tracked for the calibration. The marker should sweep the scene to cover with its positions most of the visible area possibly. The camera configuration used for this sequence is an arch containing twenty cameras.





Marker Tracker

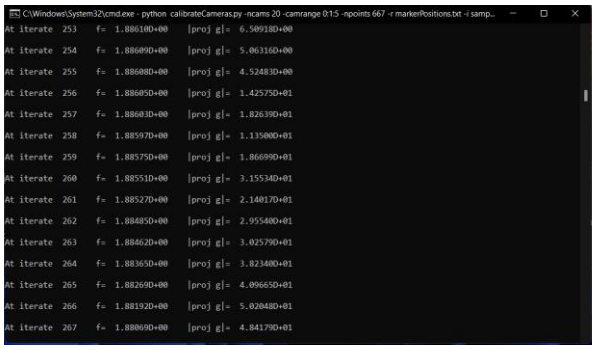
This tool generates calibration points based on the calibration sequence. The calibration sequence looks similar to the one you will process with this tool kit. The difference is - the only moving object in the calibration sequence is a marker. In the default scene, the marker is an orange ball moving throughout the scene. The marker tracker tracks the ball movement for each view and each frame and writes it into the .txt file. For a single frame from any view, there are three pieces of information that Marker Tracker writes into the file: position X in the view, position Y in the view, and a number indicating if the marker was visible in the frame (1) or not visible in the frame (0).



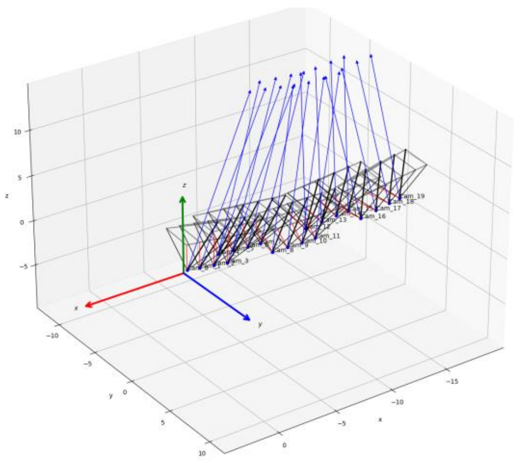
```
frame
cam: 0
frame: 84
x: 1093.588326
y: 497.288128
x: 1110
y: 432
```

Calibrator

With the calibration points generated by Marker Tracker, Calibrator tool generates .json file with sequence parameters, which will be used for the depth estimation process. The calibrator generates an epipolar line between two views and calculates how far the points are generated with a marker tracker from this line. Next, to minimize the distance between the points and the epipolar line Calibrator tool manipulates camera parameters (e.g., rotation). This process is repeated for every view pair in the sequence. Output .json file contains camera parameters for which the distance between MarkerTracker points and epipolar lines was minimal.



```
C:\Windows\System32\cmd.exe - python  calibrateCameras.py -ncams 20 -camrange 0:1:5 -npoints 667 -r markerPositions.txt -i samp...
At iterate  253    f=  1.88610D+00    |proj g|=  6.50918D+00

At iterate  254    f=  1.88609D+00    |proj g|=  5.06316D+00

At iterate  255    f=  1.88608D+00    |proj g|=  4.52483D+00

At iterate  256    f=  1.88605D+00    |proj g|=  1.42575D+01

At iterate  257    f=  1.88603D+00    |proj g|=  1.82639D+01

At iterate  258    f=  1.88597D+00    |proj g|=  1.13500D+01

At iterate  259    f=  1.88575D+00    |proj g|=  1.86699D+01

At iterate  260    f=  1.88551D+00    |proj g|=  3.15534D+01

At iterate  261    f=  1.88527D+00    |proj g|=  2.14017D+01

At iterate  262    f=  1.88485D+00    |proj g|=  2.95540D+01

At iterate  263    f=  1.88462D+00    |proj g|=  3.02579D+01

At iterate  264    f=  1.88365D+00    |proj g|=  3.82340D+01

At iterate  265    f=  1.88269D+00    |proj g|=  4.09665D+01

At iterate  266    f=  1.88192D+00    |proj g|=  5.02048D+01

At iterate  267    f=  1.88069D+00    |proj g|=  4.84179D+01
```



3 How to run the software

This calibration software pack contains a few separate tools that should be used in a specific order: *Rendering scene with Blender -> Extracting parameters -> Marker Tracking -> Calibration -> Depth Estimation -> Comparing estimated depth.*

Before you start, please ensure that you have OpenCV for C++ (https://github.com/opencv/opencv and vcpkg) and Blender installed on your computer. Also, please add the Blender installation folder to your environment variables as follows:

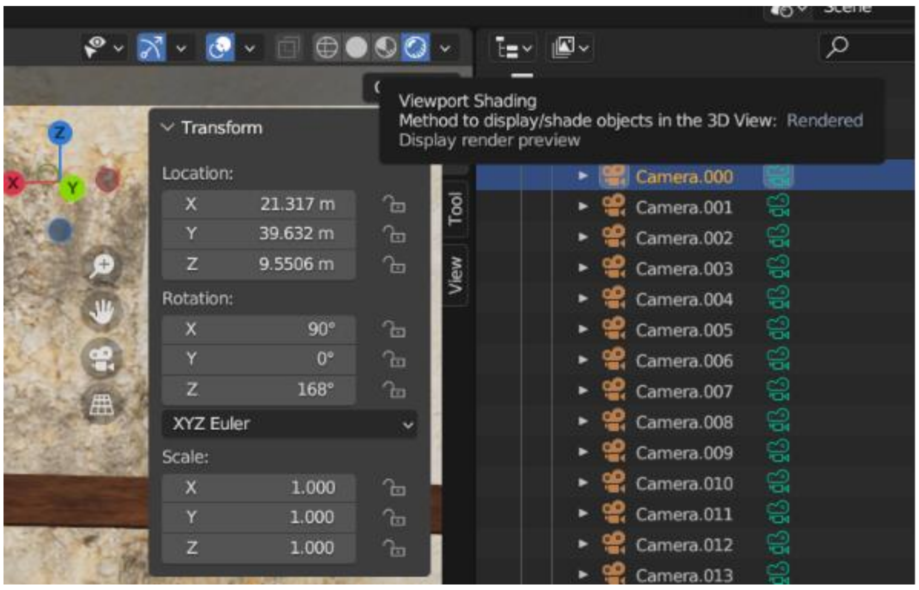
- Copy the path to the folder you have installed Blender in, e.g.: *C:\Program Files\Blender Foundation\Blender 3.4*
- Go to the *system properties -> advanced -> environment variables*
- In the Environment Variables window, search for the System Variables section and find the **Path** variable. Select this variable and click the edit button:
- Click the New button and paste the path to the Blender installation folder. Finish by clicking the Ok button.

The first step is to render textures and depth maps using Blender to use the MarkerTracker tool. You can either create your own scene, use the default scene attached with the tool or add some modifications to the scene attached with the tool.

In this scenario, we will use the default scene.

a) Go to: *CalibrationTools -> TestScene* and open *scene.blend*,

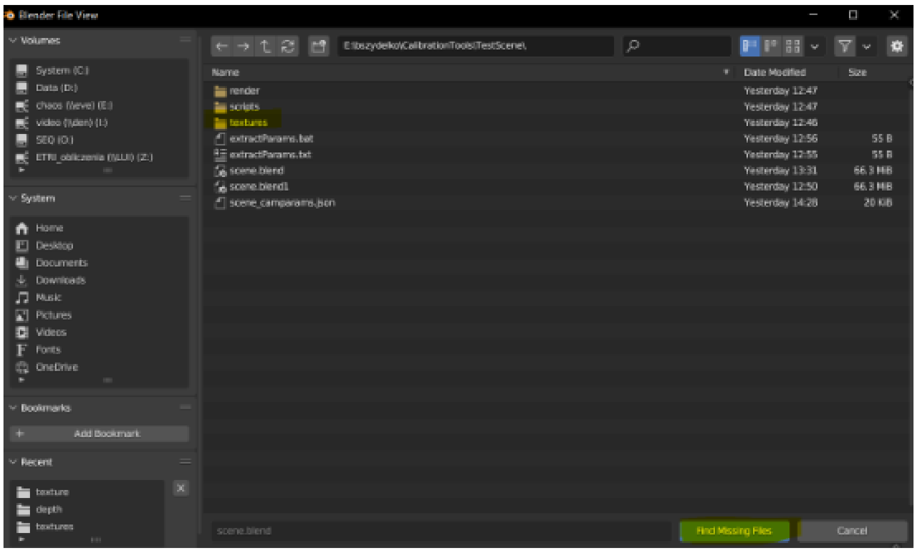
b) Switch to the ViewportShading view:



Viewport Shading
Method to display/shade objects in the 3D View: Rendered
Display render preview

Transform
Location:
X 21.317 m
Y 39.632 m
Z 9.5506 m
Rotation:
X 90°
Y 0°
Z 168°
XYZ Euler
Scale:
X 1.000
Y 1.000
Z 1.000

c) If you see a lack of textures in the scene, you should manually set the path to the folder with textures. In the Blender menu: *File -> External Data -> Find Missing Files,* choose the *textures* folder and press the *Find Missing Files* button:



E:\bszydelko\CalibrationTools\TestScene\

Name	Date Modified	Size
render	Yesterday 12:47	
scripts	Yesterday 12:47	
textures	Yesterday 12:46	
extractParams.bat	Yesterday 12:56	55 B
extractParams.txt	Yesterday 12:55	55 B
scene.blend	Yesterday 13:31	66.3 MiB
scene.blend1	Yesterday 12:50	66.3 MiB
scene_camparams.json	Yesterday 14:28	20 KiB

scene.blend

The next step is scene rendering which will result in textures and depth of a scene:

a) First, render only one frame of the calibration sequence - this will be the mask used for marker tracking (later step):

- From the top menu, select *Render -> Render Animation*,
- The files will be saved in *IV_calibrationTools -> TestScene -> render -> texture*,
- You can delete the rendered depth because there is no need for a depth mask.

b) Now render the whole sequence:

- Set the number of frames to render (in this example, the number of frames was set to 300),
- Make sure that the mask files have different names than the files that you will render now,
- From the top menu, select *Render -> Render Animation*,
- Texture files are being saved in *IV_calibrationTools -> TestScene -> render -> texture* and depth files are being saved to: *IV_calibrationTools -> TestScene -> render -> depth*,
- Use the .bat scripts located in texture and depth folders (convert2yuv.bat).

c) Extract camera parameters from Blender:

- Run the exctractParams.bat located in *IV_calibrationTools -> TestScene*.

Using MarkerTracker tool

a) Go to *IV_calibrationTools -> MarkerTracking* and run *build.bat*,

b) Go to the *build* folder and run *MarkerTracker.sln*,

c) Change Solution Configuration from Debug to Release, and set MarkerTracker as a startup project,

d) Build project,

e) Go to *CalibrationTools -> MarkerTracker -> samples* and run *runMarkerTracker.bat* from cmd,

f) The script will start opening the calibration sequence. You will see how the software is tracking the marker object,

g) The output of the MarkerTracker tool is a markerPositions.txt file that will be used as input for the next Calibration tool.

Using Calibrator tool

a) Go to *CalibrationTools -> Calibration* and run *runCalibrateCameras.bat*. Below you can find a description of the parameters that should be given in order to run this tool:

```
- -ncams (the number of cameras),
- -camrange *start\_cam:step:end\_cam*,
- -npoints (number of marker positions which can be found by looking at the number of lines in the *CalibrationTools->Calibrator->markerPositions.txt* file),
- -r (path to the ground truth marker positions file, which is shown in the previous point),
- -i (initial camera positions file name),
- -o (output file name).
```

b) This tool outputs a few.json files containing camera parameters after calibration. E.g., calibParams_it5.json contains calibrated camera parameters after the 5th iteration. One of these files should be used as an input file for depth estimation. It is advised to choose the file with the smallest reprojection error (it is visible after each iteration in cmd when running *runCalibrateCameras.bat*).

Depth estimation

a) Go to *IV_calibrationTools -> DepthEstimation*,

b) The estimator we attached to this toolkit is MPEG reference software called Immersive Video Depth Estimation. Instructions on how to use it can be found here: *IV_calibrationTools -> DepthEstimation -> ivde -> README.md*,

c) Go to *groundTruthParams* and run *runDepthEstimator.bat* to estimate depth from ground truth parameters that you have extracted after rendering the calibration sequence,

d) Next go to *calibratorParams* folder,

e) Edit the *runDepthEstimator.bat* by pasting the name of the file with the smallest reprojection error as the second input file,

f) Run the abovementioned .bat script to estimate depth from calibrated parameters.

Depth comparison

a) When the depth estimation process is finished, go to *IV_calibrationTools -> DepthCompare* and run *calcBadMatchedPixels.py* script,

b) The script will output the *results.xlsx* file comparing depth estimated from ground truth parameters with depth estimated from calibrated parameters.

The example of the results is shown below in the table. The comparison shows what is the percentage of pixels that differ more than set threshold when comparing ground truth depth maps with ones estimated using parameters from proposed calibrator. Bit depth of depth maps is 16 bits.

Threshold / View	512	1024	2048
v0	23.13908	11.97405	6.228829
v1	14.96726	7.251013	3.627074
v2	9.691117	2.939574	1.148582
v3	7.599875	3.163966	1.883343
v4	16.22242	4.836034	3.631125
v5	11.83044	1.749421	1.175878
v6	10.22304	1.786989	1.016975
v7	12.41512	3.515095	2.208478
v8	5.13479	3.422984	2.236015
v9	2.420525	1.278742	1.013937
v10	2.550974	1.048852	0.808642
v11	4.261767	2.080584	1.056086
v12	10.91739	3.888744	3.162809
v13	8.312114	1.17988	0.91088
v14	8.160639	1.155816	0.8519
v15	11.83449	4.163146	3.438561
v16	8.615403	4.649981	2.457562
v17	11.26138	3.370467	1.989149
v18	16.0719	6.624518	2.658083
v19	21.53328	10.33893	4.12066

The results below compare depth maps estimated using parameters from proposed calibrator with depth maps depth maps estimated using ground truth parameters.

Threshold / View	512	1024	2048
v0	22.54596	10.73838	4.254292
v1	18.53265	6.774595	2.806761
v2	11.04282	2.643374	0.891831
v3	9.80869	2.864487	1.306279
v4	21.07687	4.339603	2.930218
v5	10.91001	1.158275	0.607591
v6	8.626929	0.906298	0.485725
v7	9.362799	1.768374	1.002508
v8	7.528115	1.498409	0.62201
v9	5.385561	0.535204	0.2799
v10	7.751061	0.496238	0.219618
v11	7.959925	1.073302	0.292631
v12	5.477045	2.106578	0.849151

v13	3.330343	0.627267	0.403983
v14	4.63614	0.650511	0.44811
v15	10.32292	3.139371	2.303096
v16	8.087867	2.749228	0.91821
v17	10.94184	2.620129	1.097367
v18	15.25251	5.39294	1.837047
v19	17.9797	8.217737	3.055122

4 Recommendations

It is recommended to include this framework as a reference tool for immersive video activities.

5 References

[1] A. S. Olagoke, H. Ibrahim and S. S. Teoh, "Literature Survey on Multi-Camera System and Its Application," in IEEE Access, vol. 8, pp. 172892-172922, 2020.
[2] Xiaogang Wang, "Intelligent multi-camera video surveillance: A review," Pattern Recognition Letters, Volume 34, Issue 1, 2013, Pages 3-19.
[3] Joaquim Salvi, Xavier Armangué, Joan Batlle, "A comparative review of camera calibrating methods with accuracy evaluation," Pattern Recognition, Volume 35, Issue 7, 2002, Pages 1617-1635.
[4] T. Tsoy, R. Safin, E. A. Martinez-Garcia, S. Dutta Roy, S. K. Saha, and E. Magid, "Exhaustive Simulation Approach for a Virtual Camera Calibration Evaluation in Gazebo," 2022 8th International Conference on Automation, Robotics and Applications (ICARA), Prague, Czech Republic, 2022, pp. 233-238
[5] Kukolj D., Bolecek L., Polak L., Kratochvil T., Zach O., Kufa J., Slanina M., Grajek T., Samelak J., Domański M., Milovanovic D.A., "3D Content Acquisition and Coding," (2019) Signals and Communication Technology, pp. 41 - 95.
[6] D. Mieloch, A. Dziembowski and M. Domański, "Depth Map Refinement for Immersive Video," in IEEE Access, vol. 9, pp. 10778-10788, 2021.
[7] A. Dziembowski, D. Mieloch, S. Różek and M. Domański, "Color Correction for Immersive Video Applications," in IEEE Access, vol. 9, pp. 75626-75640, 2021.
[8] D. Mieloch et al., "Overview and Efficiency of Decoder-Side Depth Estimation in MPEG Immersive Video," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 9, pp. 6360-6374, Sept. 2022.
[9] S. Rogge et al., "MPEG-I Depth Estimation Reference Software," 2019 International Conference on 3D Immersion (IC3D), Brussels, Belgium, 2019.
[11] D. Scharstein and R. Szeliski, ''High-accuracy stereo depth maps using structured light,'' in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Jun. 2003, pp. 195–202.
[12] M. M. Hannuksela and Y. -K. Wang, "An Overview of Omnidirectional MediA Format (OMAF)," in Proceedings of the IEEE, vol. 109, no. 9, pp. 1590-1606, Sept. 2021.
[13] "Depth Map Formats Used Within MPEG 3D Technologies," document ISO/IEC JTC1/SC29/WG11 MPEG2017/N16730, Geneva, Switzerland, Jan. 2017.

6 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).