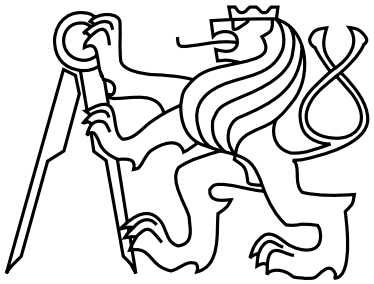
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics



Bachelor's Thesis

Online Calibration of Cameras on Cars

Leonid Tulin

Supervisor: Doc. Ing. Tomáš Pajdla, PhD.

Study Programme: Cybernetics and Robotics, Bachelor

Field of Study: Robotics

May 25, 2018

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Tulin Leonid** Personal ID number: **453550**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Online Calibration of Cameras on Cars

Bachelor's thesis title in Czech:

Průběžná kalibrace kamer na automobilu

Guidelines:

1) Study principles of Structure from Motion and Camera Calibration [1,2].
2) Investigate the quality of camera calibration in COLMAP system [3], evaluate COLMAP camera models on Oxford Robot Car Data [4] and choose and/or adjust the model.
3) Propose an approach to selecting subsets of images providing stable calibration and demonstrate it on Oxford Robot Car Data.

Bibliography / sources:

[1] R. I. Hartley and A. Zisserman, A. Multiple View Geometry in Computer Vision. Cambridge University Press, 2004
[2] C. Mei and P. Rives. Single View Point Omnidirectional Camera Calibration from Planar Grids. ICRA 2007
[3] https://github.com/colmap
[4] http://robotcar-dataset.robots.ox.ac.uk/

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Tomáš Pajdla, Ph.D., Applied Algebra and Geometry, CIIRC

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **12.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

doc. Ing. Tomáš Pajdla, Ph.D. Supervisor's signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Ing. Pavel Ripka, CSc. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Aknowledgements

I would like to thank to doc. Ing. Tomáš Pajdla, Ph.D. for his expert guidance, provided literature and language correction.

Declaration

I declare that the presented work was developed independently and that i have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

V Praze dne 24. 5. 2018 .....

Abstract

Calibration of cameras is the task of determining the orientation and the location of the camera in the space from the images, made by it. For now, this task is usually solved by offline methods of calibration.

In this work we analyze Structure-from-Motion pipelines and their workflow to calibrate the mutual position of cameras in space and their internal parameters. For our tests we chose one of the most stably-worked SfM pipeline Colmap.

Calibrated dataset is provide by Oxford Robotcar. Sequences of photos were made by autonomous car and placed as real-world data for development and testing of algorithms that used in autonomous vehicle research.

As a result we can calibrate internal camera parameters with low deviation.

Abstrakt

Kalibrace kamer je úkolem určení orientace a umístění kamery v prostoru s použitím snímků udělaných pomocí této kamery. Zatím se tento úkol obvykle řeší offline kalibračními metodami.

V této práce analyzujeme strukturu práce Structure-from-Motion softwarů, abychom je mohli použit pro kalibraci vzájemné polohy kamer v prostoru a jejich vnitřních parametrů.

Kalibrovaný dataset je poskytován společností Oxford Robotcar. Sekvence snímků jsou udělané pomocí autonomního auta a umístěny jako skutečná data pro rozvoj a testování algoritmů používaných při výzkumu autonomních vozidel.

V důsledku můžeme kalibrovat interní parametry kamery s nízkou odchylkou.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays, one of the most popular directions in the automotive industry is the development of autopilot and auxiliary programs such as parking assistant, emergency breaking, cruise control, etc. Since the main sensors for these tasks are cameras (or system of cameras), the problem of their calibration becomes very important.

Camera calibration is the task of determination camera internal and external parameters [2](par. 6) from photos or video made by the camera. Camera calibration problem frequently used in computer vision and object recognition systems. Internal parameters are focal length, image sensor format, principal point and distortion parameters. External parameters are used to denote the coordinate system from 3D world coordinates to 3D camera coordinates.

1.2 Problem formulation

Methods that generally used for car cameras calibration are called photogrammetric[4] methods. Calibration is performed by observing a calibrated object whose geometry in 3D space is well known. This means, that with every needed calibration, car should be send to the service station. Another methods are named self-calibration. Software of this category do not use calibration objects - only the movement of cameras in a static scene.

While there is no stable public method for online calibration of cameras, we will try to analyze one of the offline Structure-from-motion 3D reconstruction pipelines as Colmap, Bundler, Theia, OpenMVG ect. [11, 16, 8, 13] to find how they could be used in online calibration. For our tests we chose Colmap because it stably developing open-source software and in addition has a line of prepared methods of matching and camera models.

This Structure-from-motion pipelines use sequences of photos from cameras to make a 3D reconstruction of object captured by cameras. During photo analysis, program define the camera model and try to find internal and external parameters for each image. That work is aimed to analyze several problems: first - is Colmap can reconstruct the path of car,

second - how precisely it will determinate internal camera parameters and make a conclusion if we could use it for online calibration in the future.

Chapter 2

Colmap

COLMAP[11] is a general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline with a graphical and command-line interface. It offers a wide range of features for reconstruction of ordered and unordered image collections. The software is licensed under the GNU General Public License.

3D reconstruction from images traditionally first recovers a sparse representation of the scene and the camera poses of the input images using Structure-from-Motion. This output then serves as the input to Multi-View Stereo to recover a dense representation of the scene.

2.1 Structure-from-Motion

Structure-from-Motion (SfM) is the process of reconstructing 3D structure from its projections into a series of images. The input is a set of overlapping images of the same object, taken from different viewpoints. The output is a 3D reconstruction of the object, and the reconstructed intrinsic and extrinsic camera parameters of all images.

Whole process of SfM reconstruction we can separate in 4 steps:

- Find characteristic points (features) of complete sequence of images.
- Find similarity between these points for consecutive pairs of images (or for all of images pair).
- Filter out false matches.
- Solve the system of equations and find a three-dimensional structure together with the positions of the cameras.

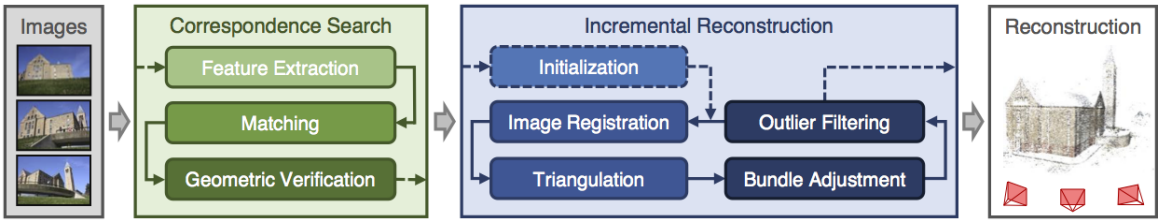


Figure 2.1: COLMAP's incremental Structure-from-Motion pipeline [11]

2.1.1 Points

There are many methods to find characteristic points. The most popular - SIFT[15] (Scale-Invariant Feature Transform). Most of the computer-vision based programs used this method for finding features. SIFT, being a reliable enough method, takes into account scale and orientation of points (which is important for arbitrary camera movement).
For each image I_m, SIFT detects sets $F_i = \{(x_j; f_j) \mid j = 1...N_{F_i}\}$ of local features at location $x_j \in R^2$ represented by an appearance descriptor f_j.

2.1.2 Matching

Each feature has its descriptor. If descriptors of points on different pictures are similar - we can assume that this is the same physical object. The easiest descriptor is the small point area. Practical approach tests every image pair: it searches for feature correspondence in image I_a for every feature in image I_b,using a similarity metric comparing the appearance f_j of the features.

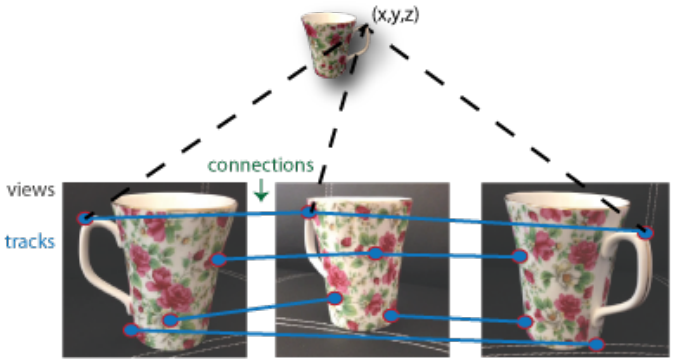


Figure 2.2: Define 3d location of point [x,y,z] using motion tracks[10]

2.1.3 False matching filter (Geometric Verification)

Regardless of descriptor quality there are many mismatches. In other words: it is not guaranteed that corresponding features actually map to the same scene point.
There we can point out 2 steps:

- Geometrically independent filtering.
- Epipolar geometry filtering.

Geometrically independent filter means a simple filter, for example it is the best descriptor match from first image to the second should be equal to the best descriptor match from the second image to the first.
The next filter uses epipolar geometry[3]. This geometry says that each point(feature) in one image and corresponding point on another(match) should be on the same line. This line does not depend on the true three-dimensional coordinates of the point. Mathematically, this property is expressed by the equation:

$$m^T F m' = 0 \tag{2.1}$$

$m = (u, v, 1)^T$ - homogeneous coordinates;
F - The affine fundamental matrix.

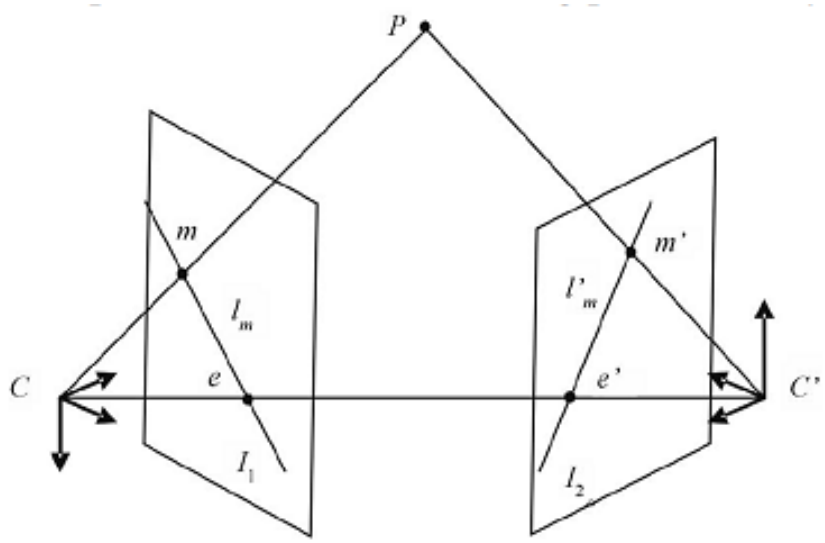


Figure 2.3: Epipolar geometry. C and C' - optical camera centers. If point P on the first image is projected in **m** then on another image its projection should be searched on the line l'_m.

Since we do not know the fundamental matrix, for verification we use next method :

1. Select random points and calculate the fundamental matrix from there correspondences (method described in [2] paragraph 14.3.2).
2. Calculate , how many points satisfy the condition 2.1 with needed accuracy.
3. End cycle when we have sufficient number of features.

The output of these stage is a scene graph: images, nodes and verified matches.

2.1.4 Finding three-dimensional structure

2.1.4.1 Start and image adding

SfM starts the model with a selection of two-view reconstruction. The reconstruction may never recover from a bad initialization , so choosing a suitable initial pair is critical. Method described in [1] .New images can be added to the current model by solving the (PnP) Perspective-n-Point problem (the problem of determining the pose of a calibrated camera from n correspondences between 3D reference points and their 2D projections) using matches to triangulated points in already registered images (2D-3D correspondences). This algorithm includes estimating the pose P_c and, for uncalibrated cameras, its intrinsic parameters.

2.1.4.2 Triangulation

A newly added image must observe existing scene points increase scene coverage by extending the set of points X through triangulation. New scene must be triangulated and added to X after adding at least one image. This step is critical for SfM as it increases the stability of the existing model through redundancy and it enables registration of new images by providing additional 2D-3D correspondences.

2.1.4.3 Bundle Adjustment

Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. BA is the joint non-linear refinement of camera parameters P_c and point parameters X_k that minimizes the reprojection error using a function π that projects scene points to image space and a loss function p_j to potentially down-weight outliers.

$$E = \sum_j p_j(||\pi(P_c, X_k) - x_j||_2^2) \tag{2.2}$$

2.1.5 Multi-View Stereo (MVS)

While Structure-from-Motion is used to structure images (estimates photo location, its orientation, camera parameters), Multi-view-stereo takes this location and orientation etc information from SFM and make a 3D dense point cloud. Colmap's principle of MVS described in [5].

2.2 Used camera models

2.2.1 OpenCv

OpenCV camera model is based on on the pinhole camera model. In this model, a scene view is formed by projecting 3D points into the image plane using a perspective transformation.

$$s \begin{bmatrix} u \\ v \\ l \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$s \cdot m' = A \cdot [R|t] \cdot M'$$

Where:

- $[X; Y; Z]$ - are the coordinates of a 3D point in the world coordinate space
- $[u; v]$ - are the coordinates of the projection point in pixels
- Matrix A represents camera matrix where $c_{x,y}$ is a principal point that is usually at the image center and $f_{x,y}$ are the focal lengths expressed in pixel units.

- $[R|t]$ - matrix of extrinsic parameters.

$[R|t]$ matrix used to describe the camera motion around a static scene, rigid motion of an object in front of a still camera. So this matrix translate coordinates from point (X,Y,Z) to the coordinate system, fixed with respect to the camera.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z \ ; \ y' = y/z \ (z \neq 0).$$

$$u = f_x \cdot x' + c_x; \ v = f_y \cdot y' + c_y.$$

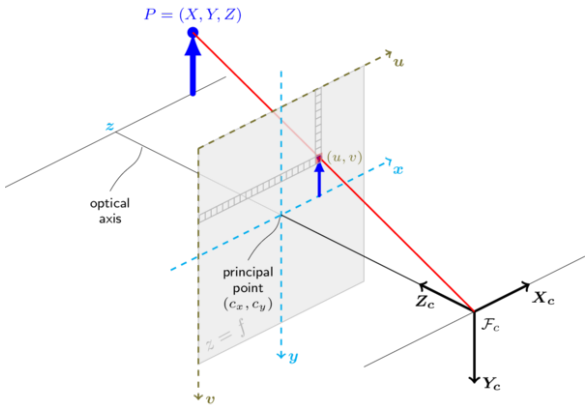


Figure 2.4: Pinhole camera model[9]

This is a linear model of the imaging process, thus the internal parameters are collinear and world lines are showed as lines and so on. But for not-pinhole lenses this hypothesis will not hold. So the most important deviation is a radial distortion. That is why we need to cure this distortion by $L(r)$ - distortion factor. In pixels coordinates the correction is written

$$\hat{x} = x_c + L(r)(x - x_c);$$

$$\hat{y} = y_c + L(r)(y - y_c).$$

where (x, y) pixels coordinates , $(\hat{x}, \hat{y})$ are corrected coordinates, (x_c, y_c) is the center of radial distortion (usual defined as principal point) and $r^2 = (x - x_c)^2 + (y - y_c)^2$. The function $L(r)$ is only defined for positive r and $L(0) = 1$. The function is given by Taylor expansion

$$L(r) = 1 + k_1 r + k_2 r + ... + k_n r$$

where $\{k_1, k_2, ..., k_n\}$ are the radial distortion coefficients.

General approach to determine $L(r)$ is the requirement that images of straight scene lines should be straight. We can define a cost function on the image lines after the correcting

mapping by $L(r)$ and this cost is iteratively minimized over the parameters k_i.[2]($p191$).

For OpenCV list of parameters is expected in the following order:

$[CAMERA\_ID]$, $[MODEL]$, $[WID]$, $[HEIG]$, $[f_x]$, $[f_y]$, $[c_x]$, $[c_y]$, $[k1]$, $[k2]$, $[k3]$, $[k4]$.

```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
# Number of cameras: 4
1 OPENCV 1280 960 1061.6 1035.29 640 480 -0.333951 0.121122 0.0025596 0.00419425
2 OPENCV_FISHEYE 1024 1024 402.294 437.448 512 512 -0.0814756 0.121485 -0.0880346 0.0181435
3 OPENCV_FISHEYE 1024 1024 428.331 444.79 512 512 0.00974985 0.0337692 -0.0801194 0.0217839
4 OPENCV_FISHEYE 1024 1024 419.591 418.346 512 512 -0.0748247 0.00877295 -0.00344216 0.000329053
```

Figure 2.5: Example of OpenCv output format

2.2.2 Simple_Radial

Simple camera model with one focal length and one radial distortion parameter. This model is similar to the model , that used another SfM - VisualSfM[17], with the difference that the distortion here is applied to the projections and not to the measurements.

$$\begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} - A\ (camera\ matrix)$$

Function of distortion factor defined as $r^2 = (x_c^2 + y_c^2)$ and $L(r) = 1 + kr^2$.
Simple_Radial_Fisheye - is equivalent to OpenCv_fisheye but have only 1 redial distortion coefficient($k_1 = k$).

List of parameters is expected in the following order:

$$[CAMERA\_ID],\ [MODEL],\ [WID],\ [HEIG],\ [f],\ [c_x],\ [c_y],\ [k].$$

```
# Camera list with one line of data per camera:
#   CAMERA_ID, MODEL, WIDTH, HEIGHT, PARAMS[]
# Number of cameras: 4
1 SIMPLE_RADIAL 1280 960 1011.91 640 480 -0.241569
2 SIMPLE_RADIAL_FISHEYE 1024 1024 427.761 512 512 -0.0768701
3 SIMPLE_RADIAL_FISHEYE 1024 1024 419.312 512 512 -0.0677735
4 SIMPLE_RADIAL_FISHEYE 1024 1024 429.323 512 512 -0.0755503
```

Figure 2.6: Example of Simple_radial and fisheye models output format

Notice that coefficient k in fisheye model ten times less then in regular simple model.

Chapter 3

Stability tests

3.1 Tested Dataset

Our research critically depends on real-world data. For tests we choose "Oxford RobotCar dataset". The analyzed sequential will be the dataset from 2014/05/06 (13:14:58 GMT)[14]. We choose that dataset because of suitable loop(3.2) and nice weather. In total there are 2046 photos from each camera. Car passed little loop taking photos with 4 cameras :

- 1 x Point Grey Bumblebee XB3 (BBX3-13S2C-38) trinocular stereo camera, 1280x960x3, 16Hz, 1/3" Sony ICX445 CCD, global shutter, 3.8mm lens, 66^o HFoV, 12/24cm baseline.
- 3 x Point Grey Grasshopper2 (GS2-FW-14S5C-C) monocular camera, 1024x1024, 11.1Hz, 2/3" Sony ICX285 CCD, global shutter, 2.67mm fisheye lens (Sunex DSL315B-650-F2.3), 180^o HFoV.

Oxford dataset also has its own calibration results made by OCamCalib [12] toolbox. This toolbox used for offline calibration omnidirectional cameras using the calibration pattern and has its own camera model similar to the model described in [6]. This model used for other purposes in computer vision and is different from the models we have prepared in Colmap.

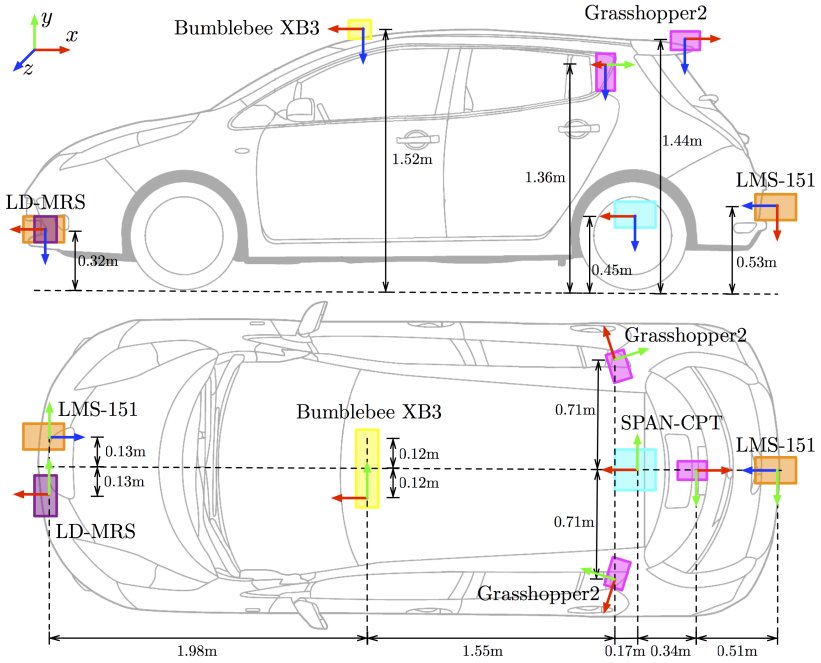


Figure 3.1: Camera positions on vehicle[14]

Figure 3.2: Oxford robotcar path

Input will be sequences of photos , taken by each camera. We will test different matching methods and different camera models witch are already prepared in Colmap. List of matching methods:

- Exhaustive Matching: every image is matched against every other image, so number of image should be low (up to several hundreds) then it will be fast enough .
- Sequential Matching: the mode is useful if the images are acquired in sequential order (image0001.jpg , image0002.jpg ...).
- Vocabulary Tree Matching: every image is matched against its visual nearest neighbors using a vocabulary tree with spatial re-ranking.

- Spatial Matching: every image against its spatial nearest neighbors.
- Transitive Matching: the mode uses the transitive relations of already existing feature matches to produce a more complete matching graph. If an image A matches to an image B and B matches to C, then this matcher attempts to match A to C directly.

For every reconstructed model colmap exports three files (as txt): cameras.txt, images.txt, points3D.txt . From this files we use only "cameras.txt" as 2.5,2.6. This file contains calibrated parameters for each camera used in the reconstruction.

3.2 Experiments

3.2.1 Pre-tests

First step was to check, how precisely colmap could restore the passed by car path. We used different matching methods, camera models and also modified sequences. For example, 3.3 shows a bad choice of a sequence.

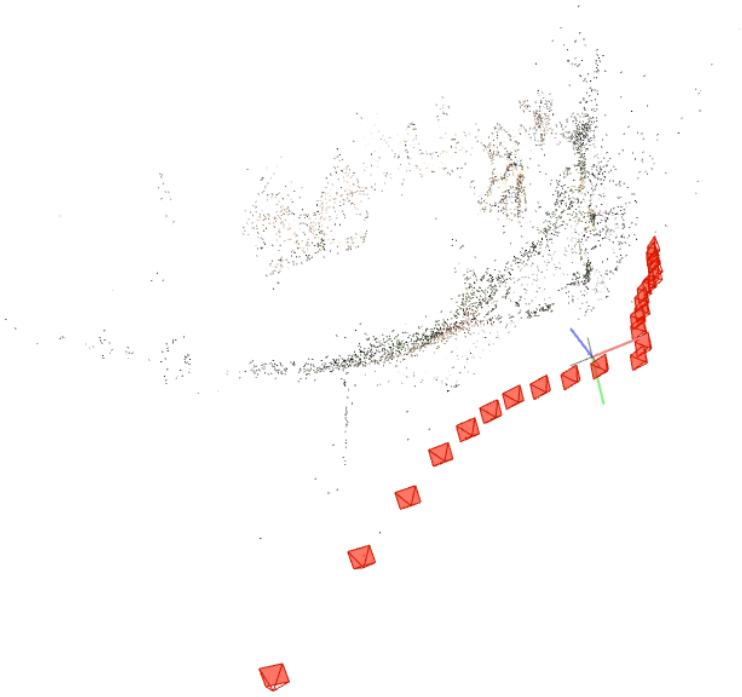
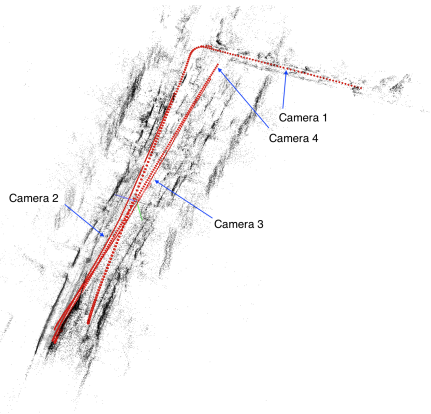


Figure 3.3: Unsuccessful reconstruction of camera position. Bad choice of a sequence

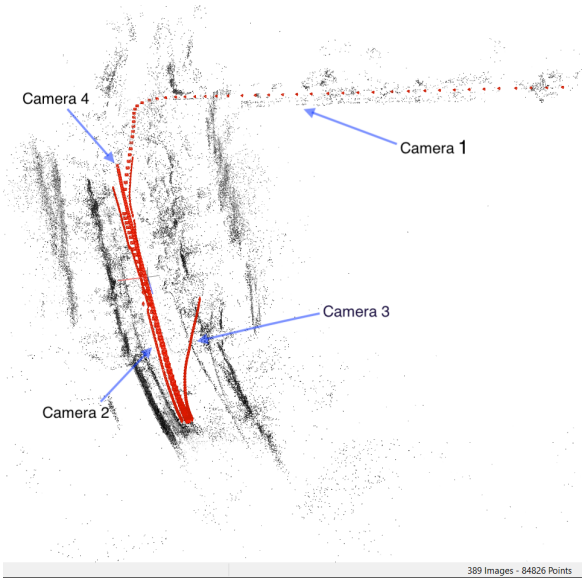
Matching methods Using exhaustive matching for a sequence of 100 photos from each camera, it was possible to reconstruct a scene that was captured by cameras during car movement. But, as was said, this method is useful for small sequences of photos, so in tests

with longer sequence (300-500 photos from each camera) the program could no longer find correct positions of cameras relative to each other. However the main problem of this method is a long time of matching.

Since Sequential Matching is used for a certain order of photographs, the data has been resorted into groups of 4 photos. Each group contains images taken from one place from all 4 cameras. As the result was a good positions of cameras relative to each other, but incorrect position of the whole system of cameras. The same result was obtained with the use of the Vocabulary Tree Matching method. Figures 3.4a and 3.4b shows example of reconstruction using different matching methods.



(a) Reconstruction using Exhaustive matching



(b) Reconstruction using Sequential matching (with re-group)

Figure 3.4: Results of reconstruction

Conclusion from these is that the most stable method of matching is Exhaustive Matching. We use it for sequences with 100 images from each camera.

3.2.2 Tested dataset

For main research we chose 3 datasets: 2 with OpenCv camera model and 1 with simple_radial. Both of models also have fish_eye model which is advantageous in our case.

- OpenCv model - 100 photos from each camera. In feature extraction we used 4 cameras: 1xOpenCv , 3xOpenCv_fisheye.
- OpenCv model - 100 photos from each camera. There we defined 3 fish_eye as one camera model. Idea is that these three cameras should have the same internal parameters, so we used only 2 camera models: 1xOpenCv , 1xOpenCv_fiheye
- Simple_Radial model - 100 photos from each camera. 4 used cameras as 1xSimple_radian and 3xSimple_radian_fisheye.

3.3 Results

As was said, the main test worked on 3 different camera models.

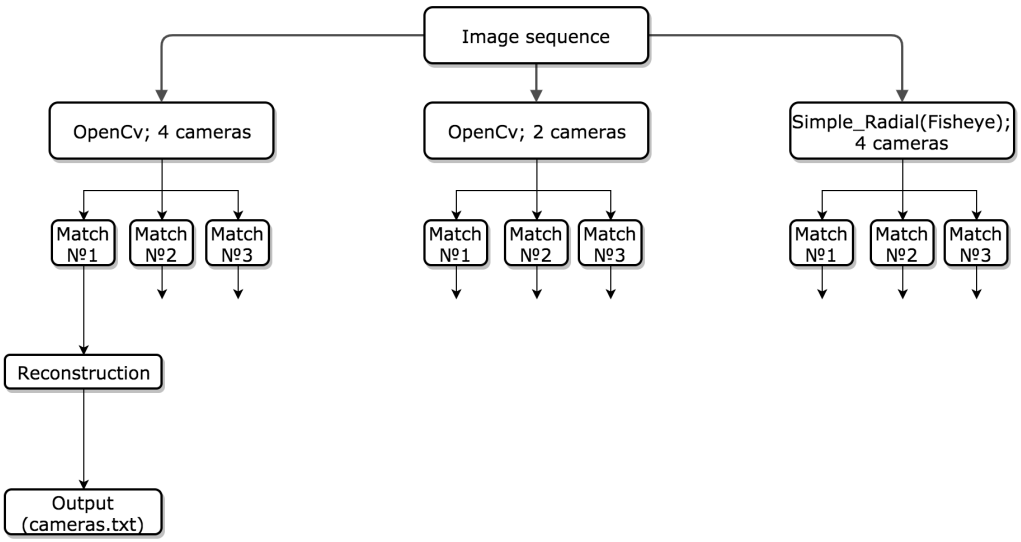


Figure 3.5: General block scheme of main test

We compare results from each reconstruction between results of reconstruction with the same matching and also with results of another matching. The idea is to check how much the parameters vary between different matches. In tables 3.1, 3.2 you can see comparing of calibrated parameters from cameras. Also for each matching there are graphs with parameters changing (3.7,3.6). Whole set of these graphs you can find in addition.

Camera 1	OprnCv model 2. match							
Reconst.	fx	fy	cx	cy	k1	k2	k3	k4
1	1104.37	1029.92	640	480	-0.364639	0.143539	0.001229	0.00258186
2	1098.07	1031.41	640	480	-0.359058	0.148217	0.001335	0.00193668
3	1013.00	1041.81	640	480	-0.34398	0.119602	-0.002218	-0.00528645
Aver	1071.80	1034.41	640	480	-0.3559	0.1371	0.001216	-0.002536
Std	51.0312	6.4776	-	-	0.0107	0.0153	0.0020	0.0044

Table 3.1: Table of camera parameters with OpenCv model from second matching of 3.8

Camera 3	SIMPLE_RADIAL_FISHEYE	
Reconst.	f	k
1	421.001	-0.0709
2	411.932	-0.0615
3	419.312	-0.0678
4	430.513	-0.0729
5	440.073	-0.0814
6	428.747	-0.0756
Aver	425.412	-0.0717
Std	9.9028	0.0068

Table 3.2: Table of camera parameters with Simple_Radial model from first matching 3.20

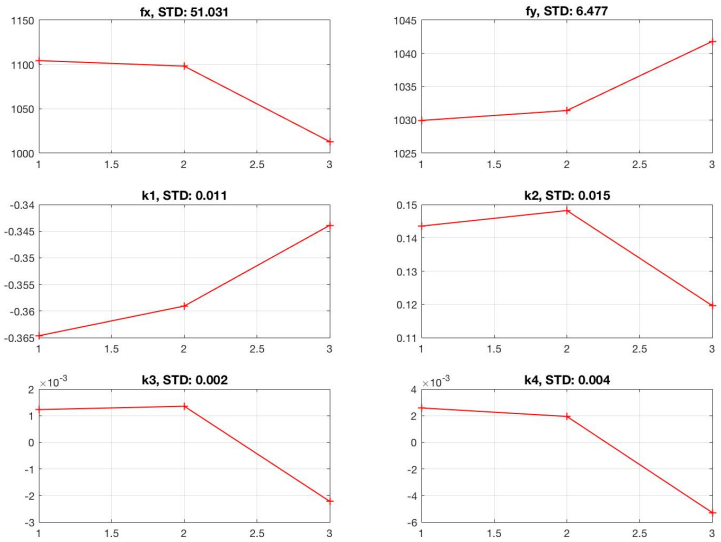


Figure 3.6: Camera 1 , match 2. X axis - number of reconstruction , Y - value of parameter

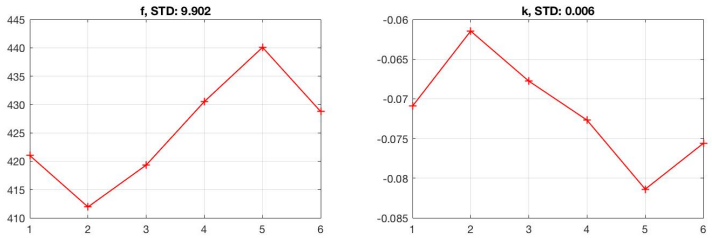
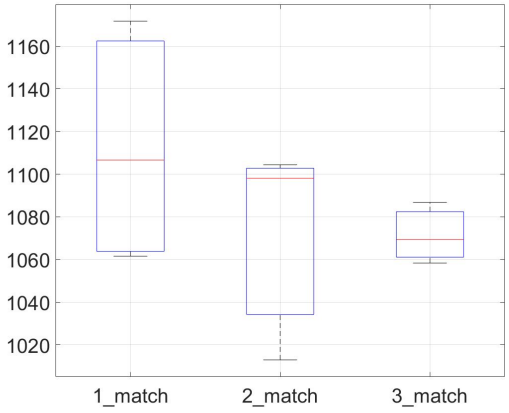


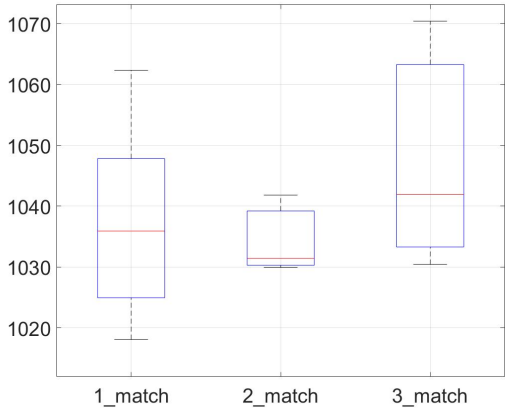
Figure 3.7: Camera 3 , match 1. X axis - number of reconstruction , Y - value of parameter

For better visibility all results shown as "boxplot" graph. Box plots provide a visualization of summary statistics for sample data and contain the following features.[7]

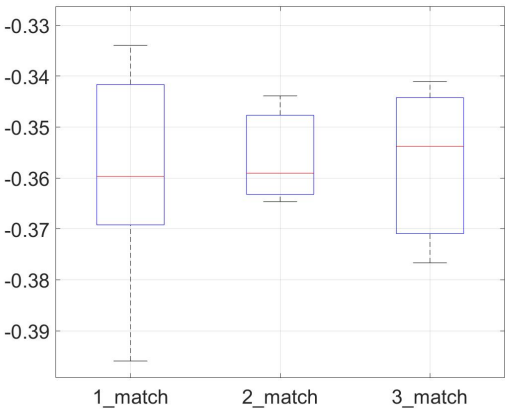
3.3.1 OpenCV , 4 cameras used



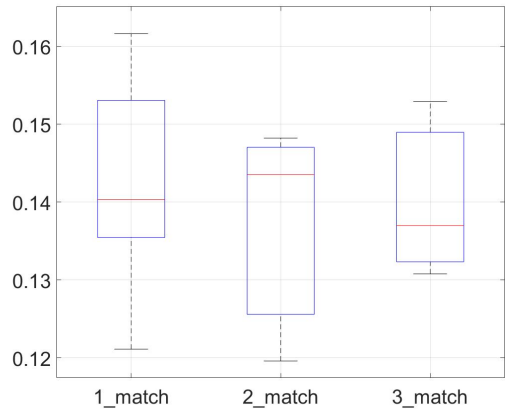
(a) fx



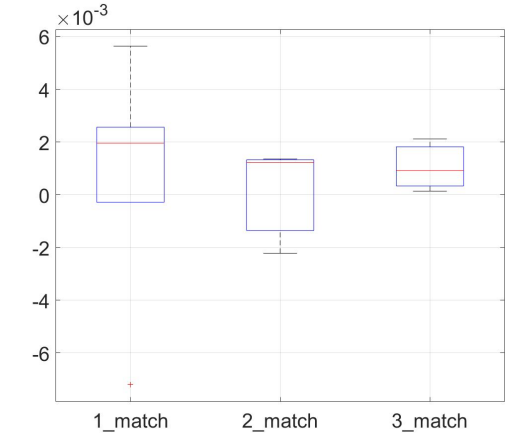
(b) fy



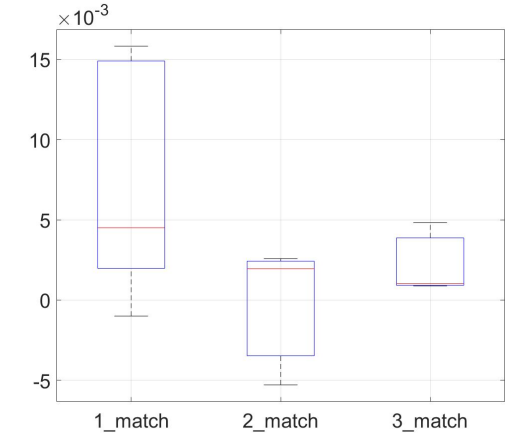
(c) k1



(d) k2

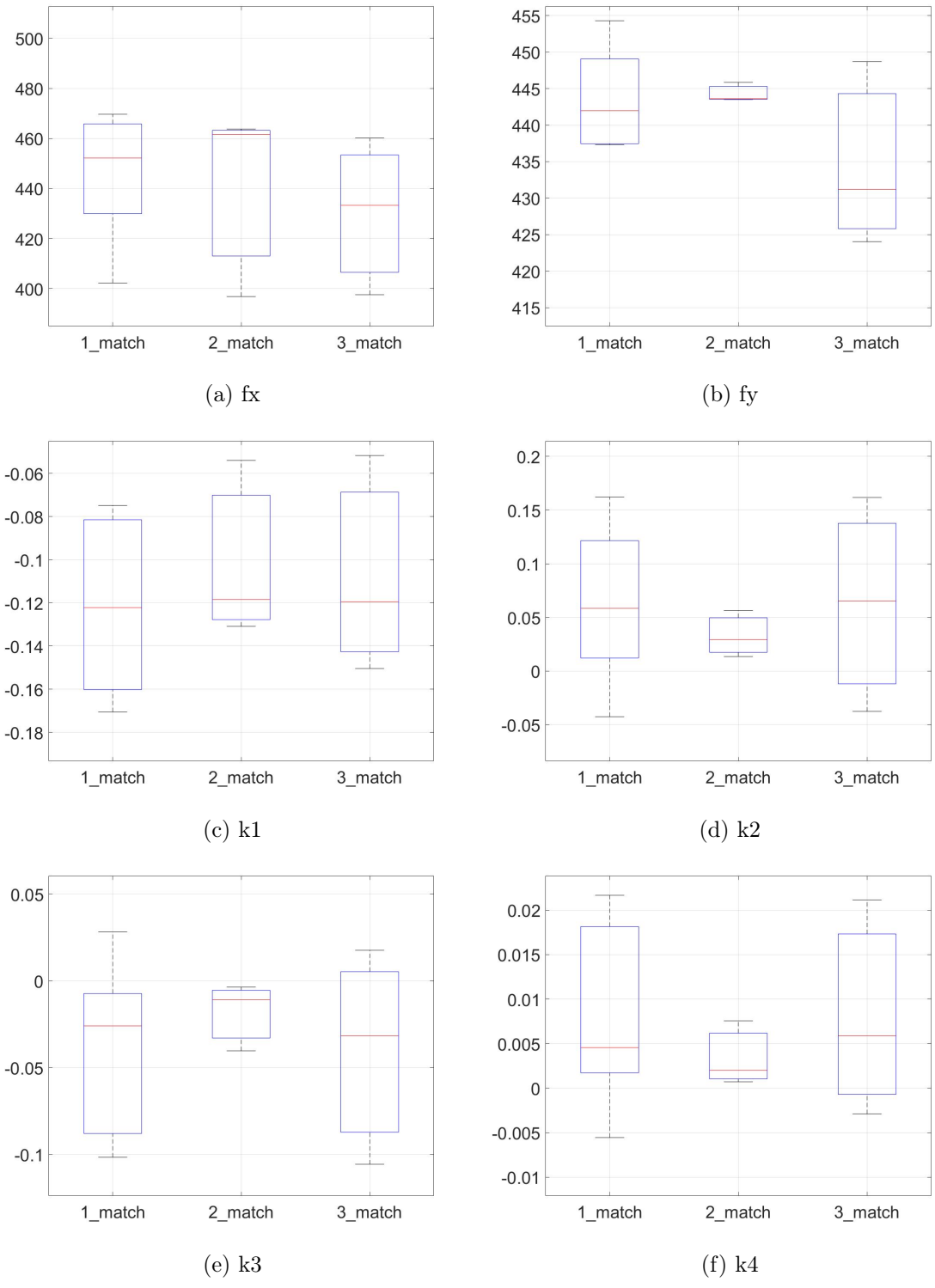


(e) k3



(f) k4

Figure 3.8: Camera 1 constants; OpenCV camera model



(a) fx

(b) fy

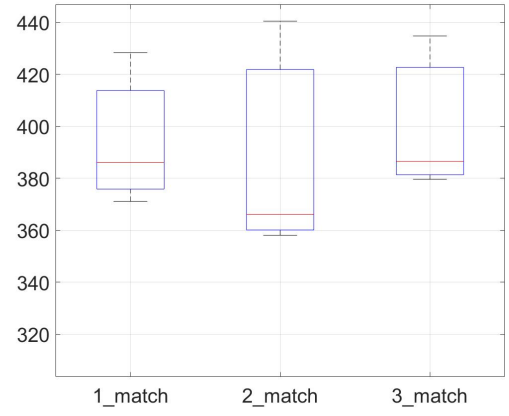
(c) k1

(d) k2

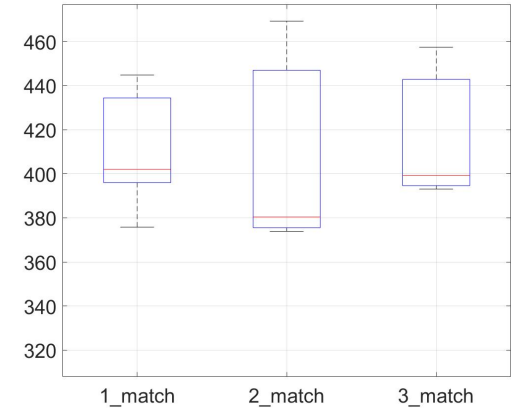
(e) k3

(f) k4

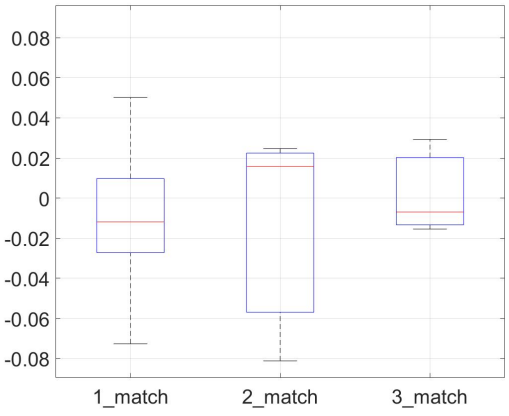
Figure 3.9: Camera 2 constants; OpenCV_Fisheye camera model



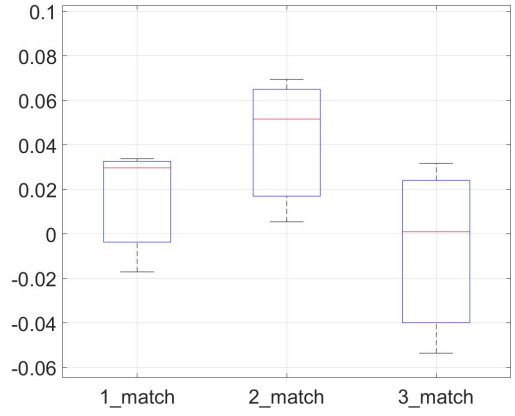
(a) fx



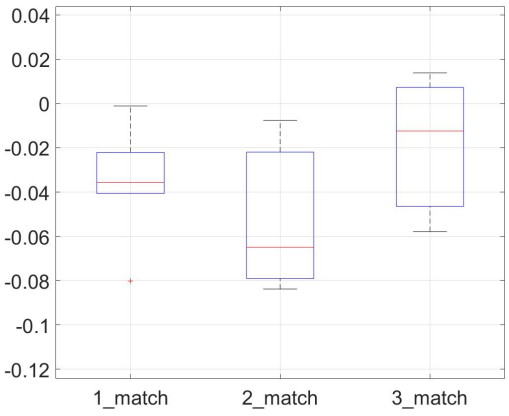
(b) fy



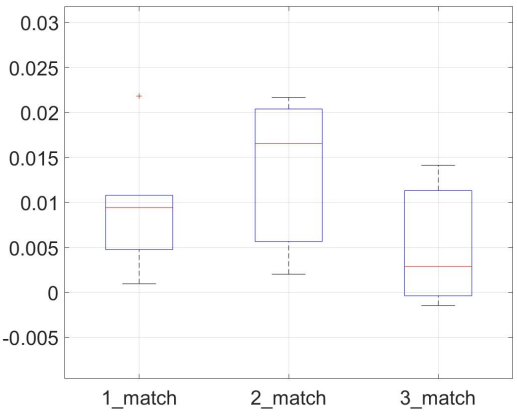
(c) k1



(d) k2

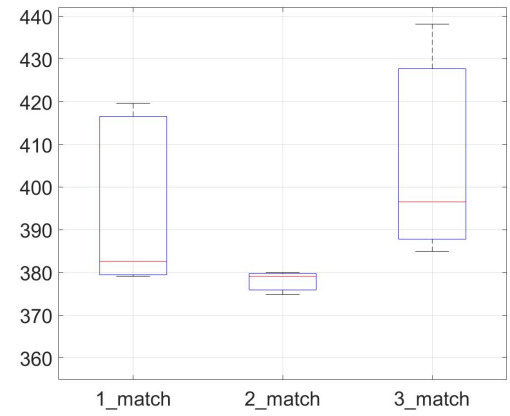


(e) k3

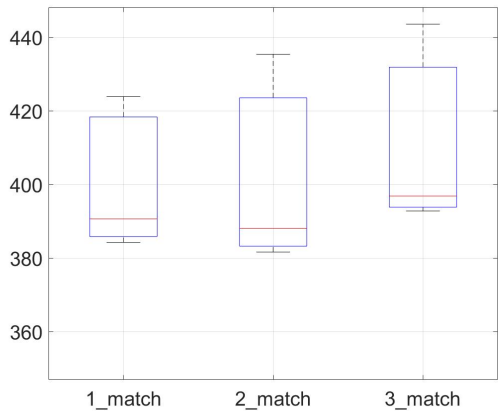


(f) k4

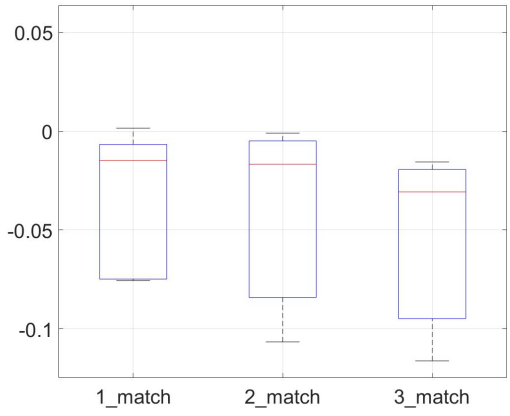
Figure 3.10: Camera 3 constants; OpenCV_Fisheye camera model



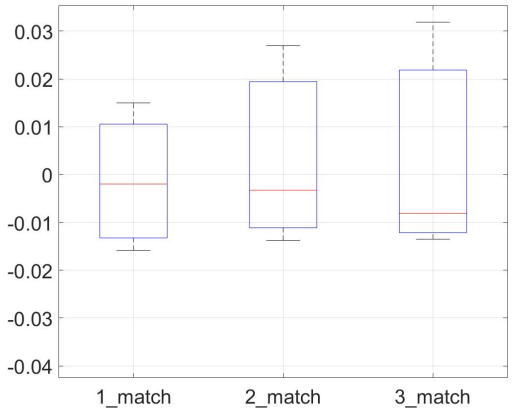
(a) fx



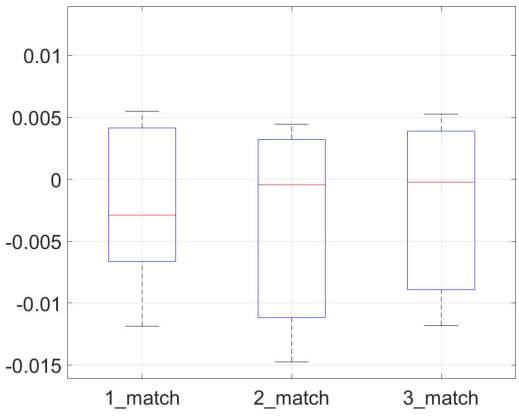
(b) fy



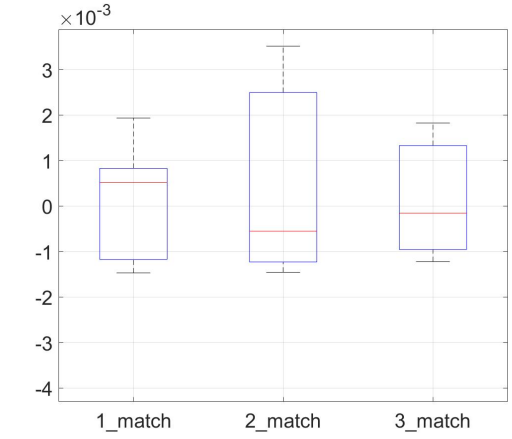
(c) k1



(d) k2



(e) k3



(f) k4

Figure 3.11: Camera 4 constants; OpenCV_Fisheye camera model

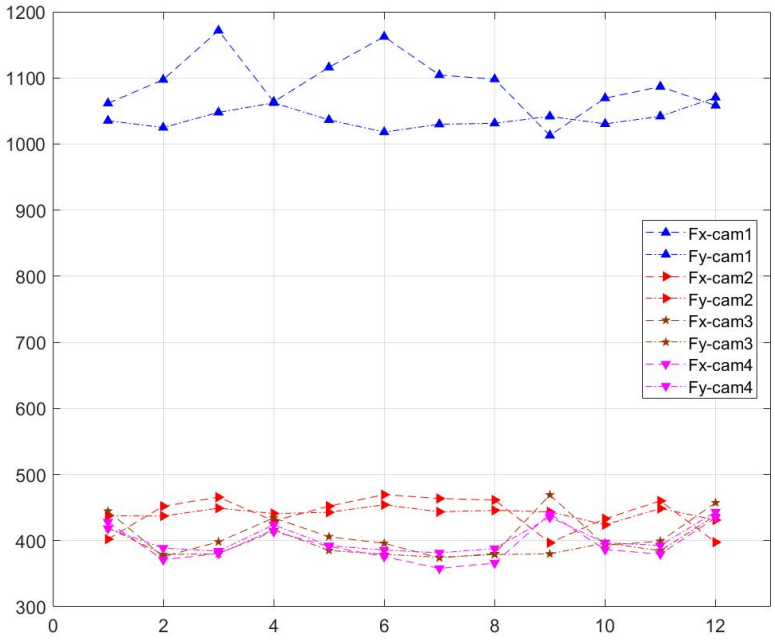


Figure 3.12: OpenCv model with 4 cameras. Coefficients fx, fy from all cameras. X axis - number of reconstruction

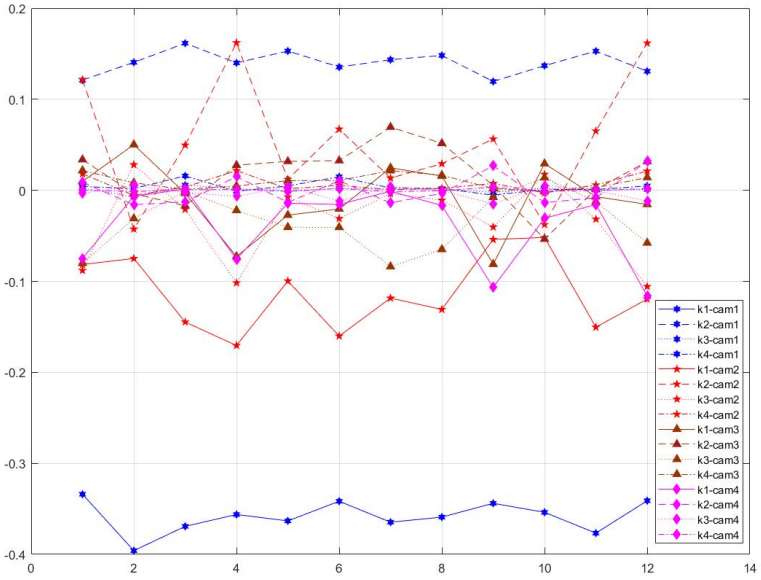
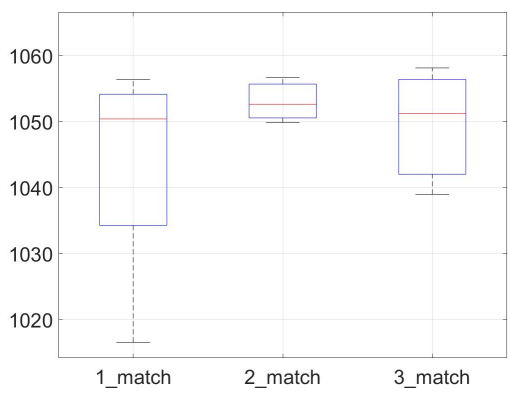
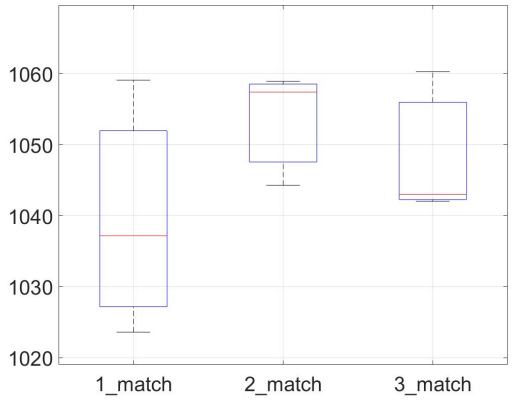


Figure 3.13: OpenCv model with 4 cameras. Coefficients k1-k4 from all cameras. X axis - number of reconstruction

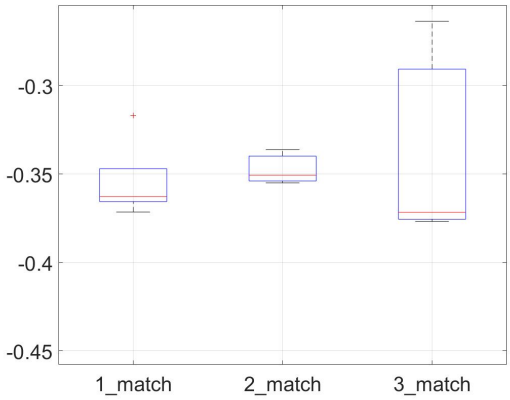
3.3.2 OpenCV , 2 cameras used



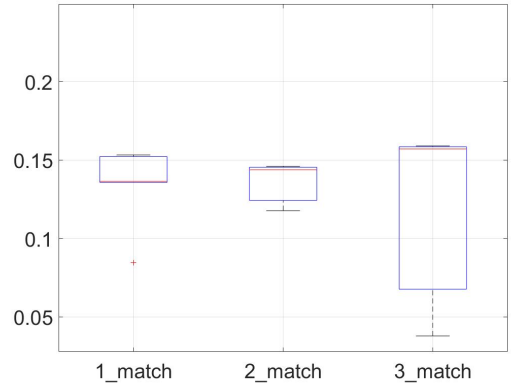
(a) fx



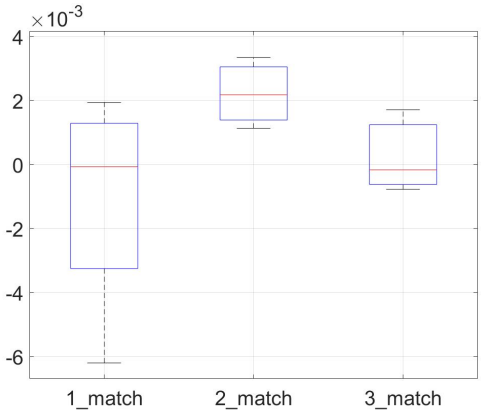
(b) fy



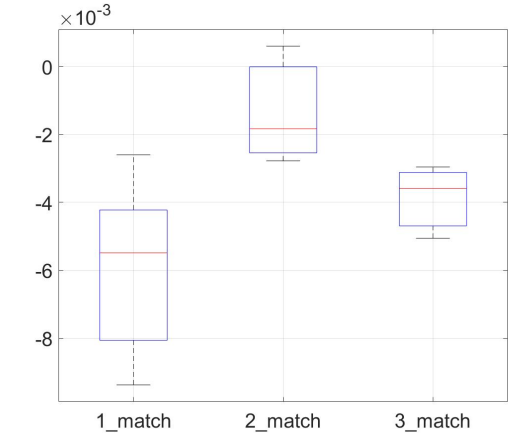
(c) k1



(d) k2

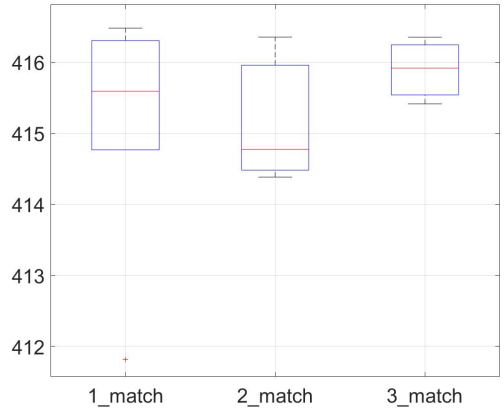


(e) k3

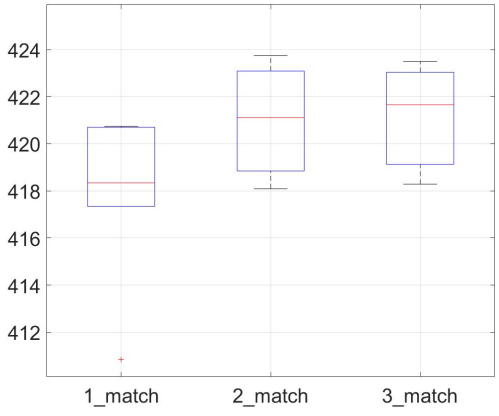


(f) k4

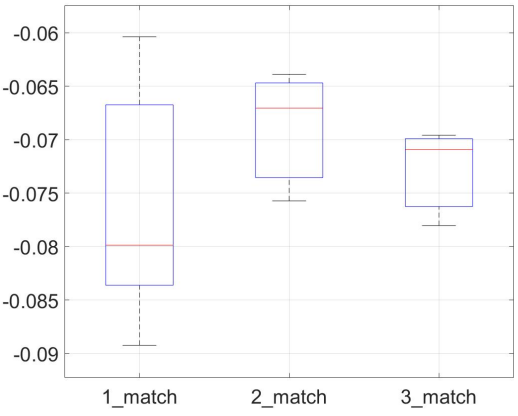
Figure 3.14: Camera 1 constants; OpenCV camera model, 2 cameras used



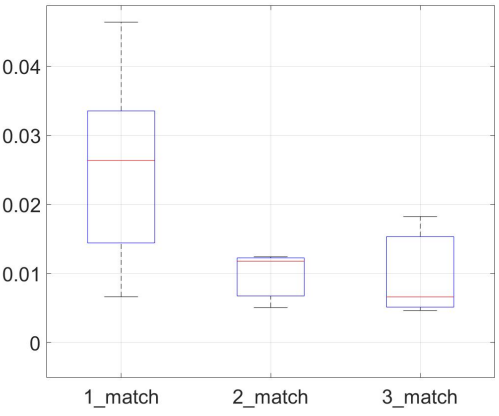
(a) fy



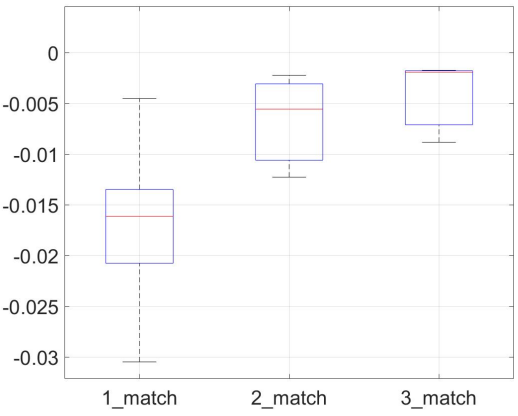
(b) fy



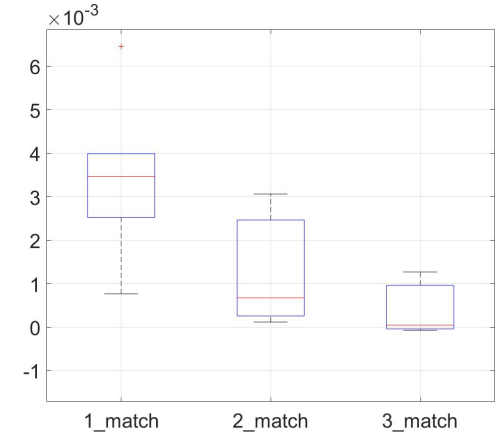
(c) k1



(d) k2



(e) k3



(f) k4

Figure 3.15: Camera 2 constants; OpenCV_Fisheye camera model, 2 cameras used

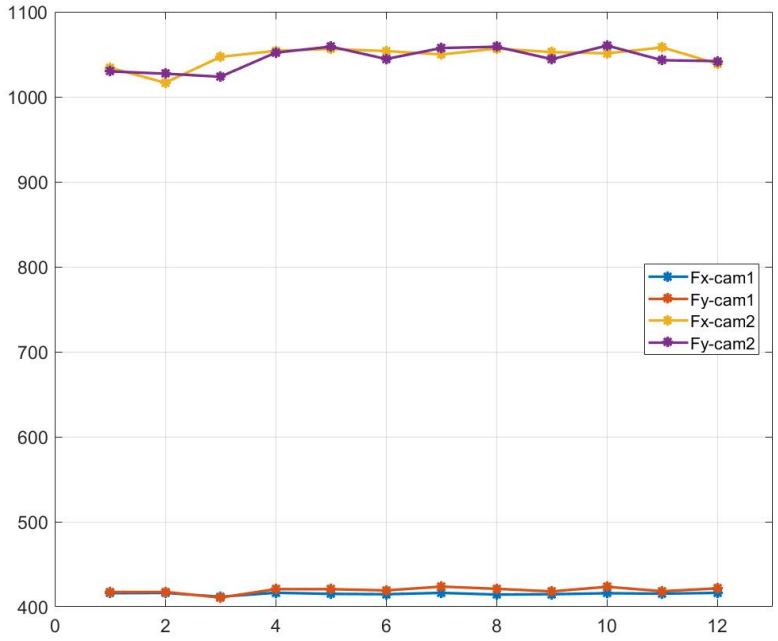


Figure 3.16: OpenCv model with 2 cameras. Coefficients fx, fy from all cameras. X axis - number of reconstruction

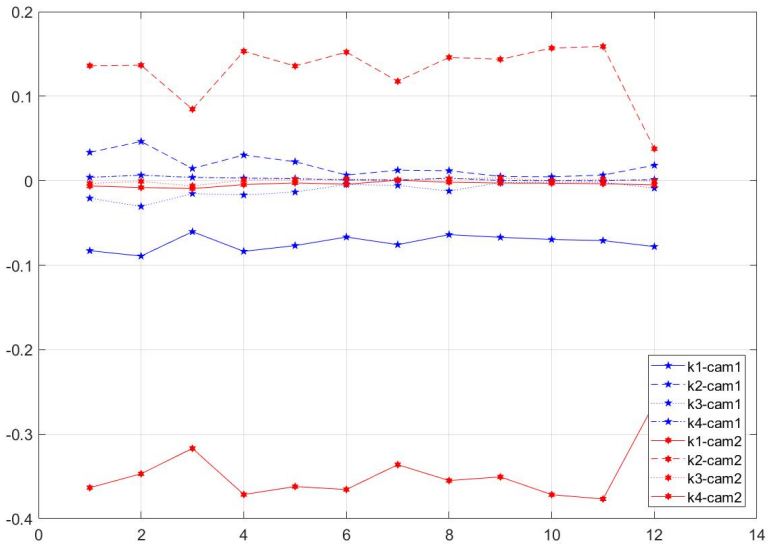
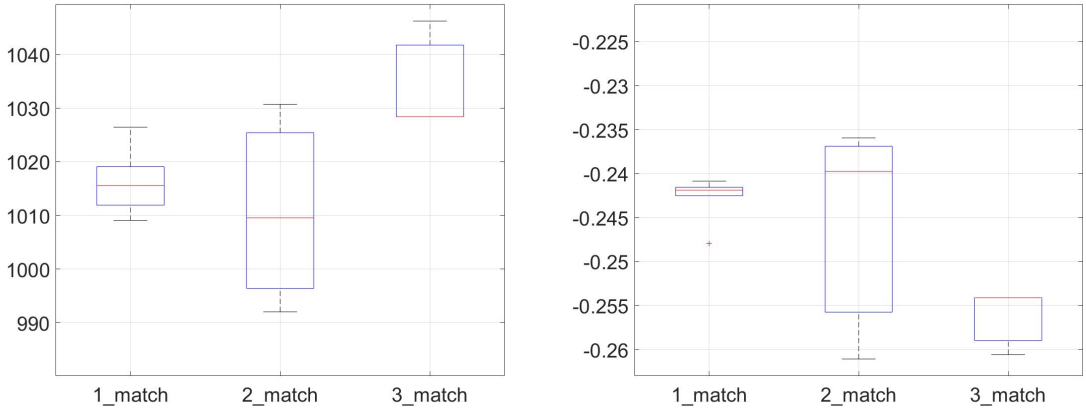


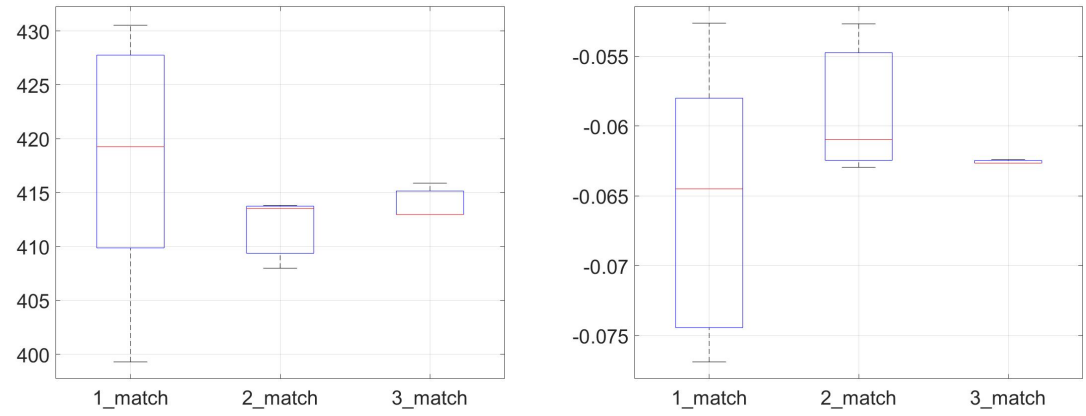
Figure 3.17: OpenCv model with 2 cameras. Coefficients k1-k4 from all cameras. X axis - number of reconstruction

3.3.3 Simple_Radial , 4 cameras used



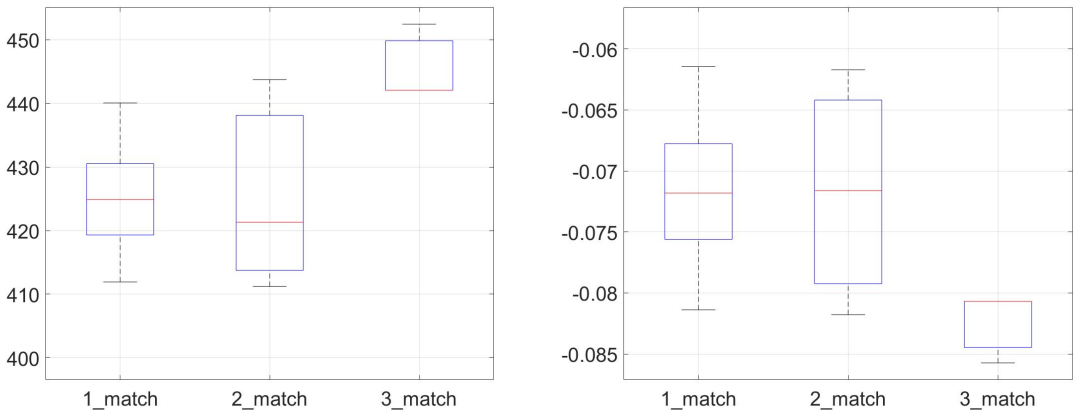
(a) F (b) K

Figure 3.18: Camera 1 constants; Simple_Radial camera model



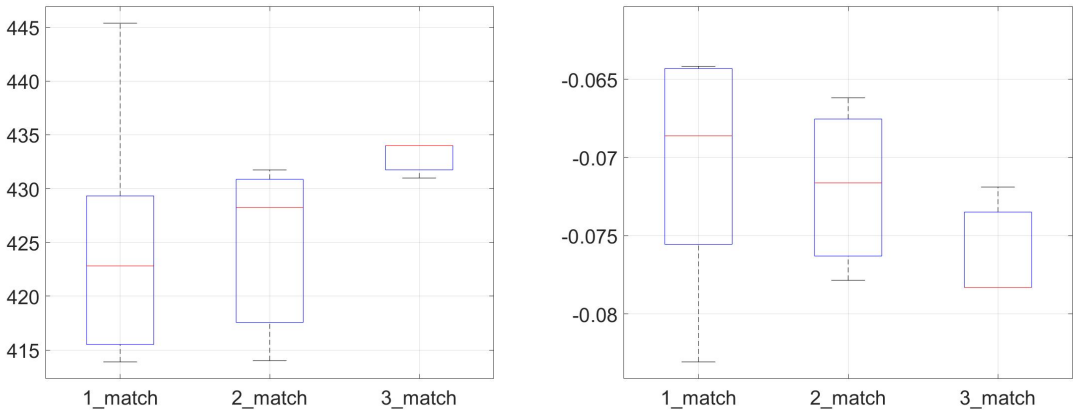
(a) F (b) K

Figure 3.19: Camera 2 constants; Simple_Radial_Fisheye camera model



(a) F (b) K

Figure 3.20: Camera 3 constants; Simple_Radial_Fisheye camera model



(a) F (b) K

Figure 3.21: Camera 4 constants; Simple_Radial_Fisheye camera model

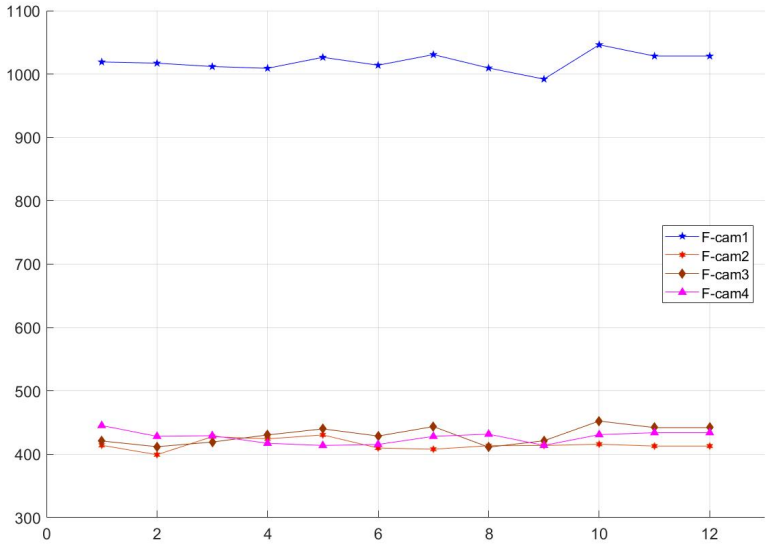


Figure 3.22: Simple_Radial camera models. Coefficients f from all cameras. X axis - number of reconstruction

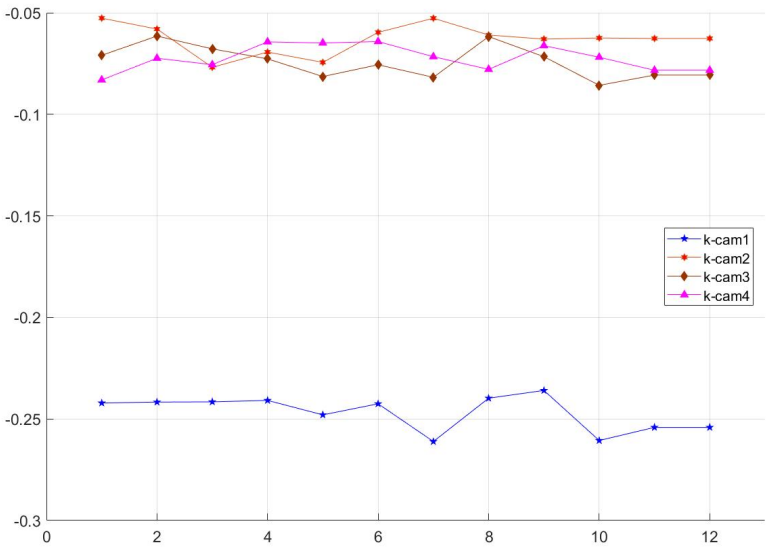


Figure 3.23: Simple_Radial camera models. Coefficients k from all cameras. X axis - number of reconstruction

Chapter 4

Discussion

From graphs 3.12, 3.16 and 3.22 we calculate mean value and standard deviation for focal length parameter for each camera.

OpenCV, 4 cameras				
∼	Cam1	Cam2	Cam3	Cam4
Mean fx	1091.9	440.4	393.8	392.7
Mean fy	1039.2	441.5	410.6	402.6
Std(fx)/ %	44.43 / 4.07%	27.29 / 6.31%	28.20 / 7.16%	20.62 / 5.25%
Std(fy)/ %	15.05 / 1.44%	8.24 / 1.87%	32.55 / 7.90%	21.63 / 5.37%

Table 4.1: Table of mean and std values. OpenCv, 4 cameras model. 3.12

OpenCV, 2 cameras		
∼	Cam1	Cam2
Mean fx	1047.4	440.4
Mean fy	1045.2	441.5
Std(fx)/ %	12.01 / 1.16%	1.31 / 0.03%
Std(fy)/ %	12.93 / 1.24%	3.46 / 0.08%

Table 4.2: Table of mean and std values. OpenCv, 2 cameras model. 3.16

Simple_radial model				
∼	Cam1	Cam2	Cam3	Cam4
Mean f	1019.4	415.2	430.3	426.8
Std / %	13.82 / 1.36%	8.65 / 2.08%	13.59 / 3.16%	9.76 / 2.29%

Table 4.3: Table of mean and std values. Simple_Radial camera model. 3.22

These tables show the stability of determination the internal parameters of cameras and we can conclude that the most stable model is the second one(OpenCv, 2 cameras) where the biggest standard deviation is 1.24%.

From graphs 3.7, 3.6 we assume that coefficients of radial distortion k_n make compensation for $f_{x,y}$ so value of k_n coefficients can be vary. All graphs that shows compensation of the coefficients can be found at addition:
/Colmap_results/Type_Of_Model/Pictures/Compensation.

Chapter 5

Conclusion

We have shown that Colmap can determinate internal camera parameters with standard deviation about 3-7%. On the other hand, we were not able to achieve good poses of cameras so it is impossible to restore the travelled path with the obtained reconstructions. This is caused by the fact, that was described in section 2.1: Colmap and other SfM pipelines determinate 3D position of cameras by operating on a static object, what is not possible with the car driving. Summary:

+ Stable determination of internal parameters.
+ Could restore short car trajectory (up to 20-30 photos from each camera).
+ Open-source flexible software - you can add your own camera model, or custom matching. Also it stably works at all OS(Windows/Linux/Mac).
- Not suitable for online use. Even sequential matching takes about 3-4 hours with sequences that are longer then 200-300 images.
- Could not restore 3d scene of car movement with big sequences.
- Worst results are expected for cars in real conditions(dusty cameras, bad weather, non static scene, ect.)

Bibliography

[1] Beder and Richard Steffen. *Determining an Initial Image Pair for Fixing the Scale of a 3D Reconstruction from an Image Sequence*. Sept. 2006. URL: https://link.springer.com/chapter/10.1007/11861898_66.

[2] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2017.

[3] *Epipolar geometry*. May 2018. URL: https://en.wikipedia.org/wiki/Epipolar_geometry.

[4] *Photogrammetry*. May 2018. URL: https://en.wikipedia.org/wiki/Photogrammetry.

[5] Johannes L. Schönberger et al. *Pixelwise View Selection for Unstructured Multi-View Stereo*. 2018. URL: https://www.researchgate.net/publication/305655847_Pixelwise_View_Selection_for_Unstructured_Multi-View_Stereo.

[6] *A Unifying Omnidirectional Camera Model and its Applications - IEEE Conference Publication*. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4409207.

[7] *Boxplot matlab*. URL: https://uk.mathworks.com/help/stats/box-plots.html.

[8] *Bundler - Structure from Motion (SfM) for Unordered Image Collections*. URL: http://www.cs.cornell.edu/~snavely/bundler/.

[9] *Camera Calibration and 3D Reconstructionf*. URL: http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html.

[10] *Camera Calibrator*. URL: https://www.mathworks.com/help/vision/ug/structure-from-motion.html.

[11] *Colmap*. URL: https://demuc.de/colmap/.

[12] *OCamCalib: Omnidirectional Camera Calibration Toolbox for Matlab - Davide Scaramuzza*. URL: https://sites.google.com/site/scarabotix/ocamcalib-toolbox.

[13] *OpenMVG*. URL: http://openmvg.readthedocs.io/en/latest/openMVG/sfm/sfm/.

[14] *Oxford robotcar dataset*. URL: http://robotcar-dataset.robots.ox.ac.uk/datasets/2014-05-06-13-09-52/.

[15] Utkarsh Sinha. *Introduction*. URL: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction.

[16] *Theia*. URL: http://theia-sfm.org/.

[17] *VisualSFM : A Visual Structure from Motion System*. URL: http://ccwu.me/vsfm/.

Addition. Contents of CD.

There you can find all data we worked with.

1. Site of Oxford robotcar dataset: http://robotcar-dataset.robots.ox.ac.uk/datasets/2014-05-06-13-14-58.
2. CD contents:
    - Colmap_results
        - OpenCv_2cameras
            - Pictures
            - Figures
            - Data
            - OpenCv_2cams_code(matlab code)
        - OpenCv_4cameras
            - Pictures
            - Figures
            - Data
            - OpenCv_4cams_code(matlab code)
        - Simple_Radial
            - Pictures
            - Figures
            - Data
            - Simple_Radial_code(matlab code)
    - Colmap (folder with Colmap installation)
    - Readme.txt (tutorial how to repeat our tests)