Spring 2019, MT-MA2

Start: February 2019
Finish: June 2019

Dual Fisheye Camera Calibration

Darko Lukic

Professor: Jan Skaloud
Assistant: Davide Antonio Cucci

Contents

Chapter 1

Introduction

Laboratory TOPO[1] from EPFL aims to use a dual fisheye camera for environment mapping. For this purpose camera Ricoh Theta V[2] from company Theta360 is chosen. However, for accurate 3D mapping accurate estimation of intrinsic parameters of the camera is required. The estimated camera parameters have to be compatible with Pix4D camera model since Pix4D will be used for 3D reconstruction of the environment. Currently, there is a robust software for the camera calibration, but it is not capable of calibrating the dual camera, like Ricoh Theta V, and it doesn't produce parameters compatible with Pix4D camera model.

The objective of this project is to choose an appropriate camera model to describe Ricoh Theta V dual lenses and to develop a calibration software. In addition, simulation of the calibration procedure and detailed analysis of the obtained results have to be done.

In the first chapter (Chapter 2) a basic theoretical background required for camera calibration will be given. A simple camera model will be described, different projection models compared and two fisheye camera models (that are important for the project) introduced. In the same chapter non-linear least squares optimisation will be explained with a focus on aspects important for the camera calibration.

In order to avoid the uncertainty of the real-world environment simulation is created (Chapter 3). Position and orientation of cameras are visualised to enable visual inspection of the results. More importantly, a procedure of generating different constellations (for a quality data acquisition) is described.

In the following chapter (Chapter 4) process of the fisheye camera calibration will be reported. For the camera calibration observed points are required and those are gathered from a chessboard. The points are used to calibrate two cameras of Ricoh Theta V independently. After parameters of the independent cameras are known rotation between two cameras can be estimated. Mentioned procedures will be described in the chapter about calibration.

[1]`https://www.epfl.ch/labs/topo/`

[2]`https://theta360.com/en/about/theta/v.html`

Results with detailed explanation will be given in Chapter 5. Results for the optimal procedure of data acquisition, independent camera calibration and dual camera calibration will be given independently. Additional material, matrices, residual analysis and variance will be provided as well to support the obtained results.

Finally, the results and the project will be discussed in Chapter 6.

Chapter 2

Background

In this chapter, the basic knowledge of camera calibration will be given. Firstly, image projection will be explained for a simple camera without lenses - pinhole camera (Section 2.1). Different types of camera lenses have different projections, therefore there are many projection models to describe them. Also, different entities can impose different camera models. The models will be explained in the following sections (Section 2.2 and 2.3). Finally, a theoretical approach for camera calibration using non-linear least square optimisation will be given (Section 2.4).

2.1 Pinhole Camera

A pinhole camera is a very basic camera without lenses. It has one tiny hole (aperture) which allows the camera to receive light from a scene. The scene is then projected to the other side of the camera. The pinhole camera is important because it doesn't introduce distortion, can be explained using simple equations and it is a basis for more complex camera models. A model of the pinhole camera is given by Fig. 2.1.

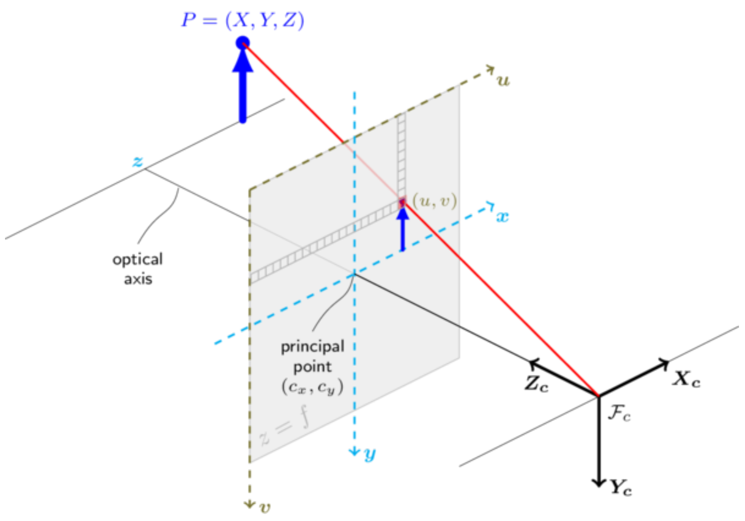


Figure 2.1: Model of a pinhole camera [12]

In the given pinhole camera model point P is located in a world coordinates X, Y, Z. The point is projected on an image plane at image coordinates u, v (mathematical coordinates x, y). Principal point (c_x, c_y) is a centre of a mathematical coordinate system of the image and a focal length is a distance between the image plane and the focal point (F_c). The camera is located at world coordinates X_c, Y_c, Z_c with Z_c pointed at the scene.

This gives us enough information, so we can calculate where the point P projected on the image plane. First, we can express mathematical coordinates on the image plane as:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + T \tag{2.1}$$

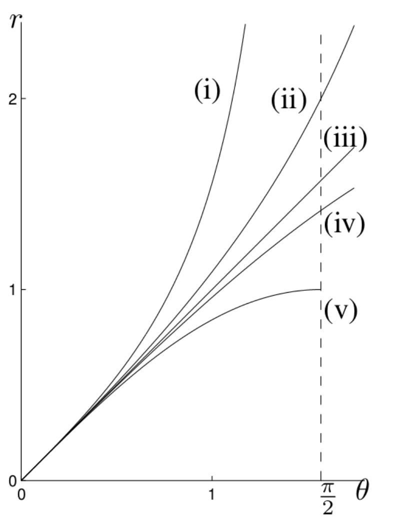
where R and T are rotation and translation of camera in respect to the world respectively. The rotation R is expressed as a 3D rotation matrix. Now, we can express it in the image coordinates:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{pmatrix} \tag{2.2}$$

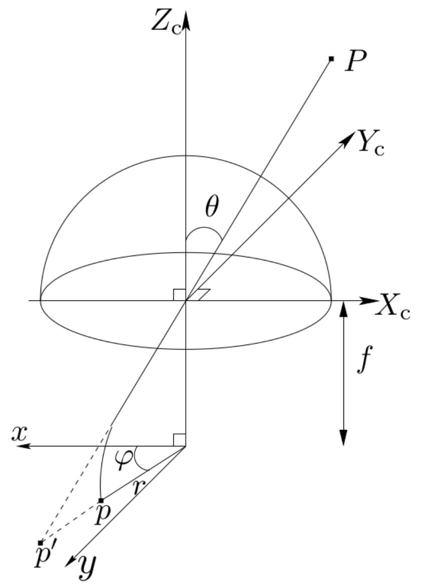
Unfortunately, the pinhole camera model is usually not applicable to real-world scenarios. Almost all modern cameras have some distortion which will be explained in the following sections.

2.2 Projection Models

Lenses enable the camera to catch more light, ensuring images of higher quality, but they also introduce deviation from ideal mapping. Most notably those are distortions (e.g. barrel, pincushion and moustache) and spherical aberration. In order to describe distortions a few projection models are proposed (see Fig. 2.2).



(a) Relation between angle θ and radial distance r



(b) A model of a camera with a radial distortion

Figure 2.2: Different projection models [10]

In Fig. 2.2a angle θ represents an angle between Z axis of camera (Z_c) and a vector from camera origin to a point in the world (P, see Fig. 2.2b). Radial distance r is a distance between principle point and projected point P on the image plane. The plot represents relation between those two parameters (θ and r) for different projection models. The following projections are shown:

• $r = ftan(\theta)$	i. perspective projection
• $r = 2ftan(\frac{\theta}{2})$	ii. stereographic projection
• $r = f\theta$	iii. equidistance projection
• $r = 2fsin(\frac{\theta}{2})$	iv. equisolid angle projection
• $r = fsin(\theta)$	v. orthogonal projection

According to Juho Kannala and Sami S. Brandt [10] stereographic, equidistance, equisolid and orthogonal projections are suitable for describing fisheye projection. However, further, in the project, we are going to use the equidistance projection. The equidistance projection is chosen because it complies with the further described fisheyecamera models.

2.3 Fisheye Camera Models

A fisheye camera lens is ultra-wide angle lens which creates a high visual distortion. The distortion is created by purpose in order to obtain hemispherical images. Those lenses have found many applications but they lack an accurate, generic, and easy-to-use calibration [10]. In order to solve the issue, different entities proposed mathematical models to describe the distortion. Since this project aims to provide calibration parameters for Pix4D we will further use and describe the model proposed by Pix4D. Also, since OpenCV provides calibration tools for fisheye lenses we will use their fisheye camera model to validate our results.

2.3.1 OpenCV Camera Model

As previously stated OpenCV fisheye camera calibration tools [12] are used as a reference for some of the further experiments. In the following text, OpenCV fisheye camera model will be described. A convention used to describe the following and all other models is:

- Capital letters (e.g. X) describe points and transformations in the three-dimensional world.
- Lowercase letters (e.g. x) represent scalars and vectors.
- Index c describes coordinates of points in respect to the camera coordinate system.
- Index h describes homogeneous coordinates.
- Points without an index are points in the world coordinate system.

For OpenCV fisheye camera model first we find rotation (R) and translation (T) of camera in respect of to the world coordinates (X)

$$X_c = RX + T \tag{2.3}$$

Then, we find pinhole projection coordinates as follows:

$$X_h = \begin{pmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{pmatrix} \tag{2.4}$$

$$r = \sqrt{x_h^2 + y_h^2} \tag{2.5}$$

$$\theta = arctan(r) \tag{2.6}$$

Fisheye distortion is described with 4 polynomial coefficients ($k_{1..4}$), therefore we can obtain angle θ as follows:

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \tag{2.7}$$

Finally, we can write expression of x_h and y_h in the image plane (x_{image}):

$$x_{image} = \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} \frac{\theta_d x_h}{r} \\ \frac{\theta_d y_h}{r} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \tag{2.8}$$

2.3.2 Pix4D Camera Model

The Pix4D fisheye camera model is published on their webpage [9], but in the following text, it will be expressed within the same convention as OpenCV fisheye camera model. The goal is to emphasise difference and show similarities.

Again, we find a point X in camera coordinate system by applying rotation (R) and translation (T) of the camera in respect of to the world coordinates (X)

$$X_c = RX + T \tag{2.9}$$

Pinhole projection coordinates are not changed:

$$X_h = \begin{pmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{pmatrix} \tag{2.10}$$

$$r = \sqrt{x_h^2 + y_h^2} \tag{2.11}$$

However, angle θ is scaled by $\frac{2}{\pi}$:

$$\theta = \frac{2}{\pi} arctan(r) \tag{2.12}$$

Fisheye distortion is described with 3 instead of 4 polynomial coefficients in OpenCV fisheye camera model. Also, powers for angle θ are in the range of 1 — 3 instead of 2 — 8.

$$\theta_d = \theta(1 + k_1\theta + k_2\theta^2 + k_3\theta^3) \tag{2.13}$$

Finally, for image projection two additional parameters (D and E) are added to describe affine transformations (e.g. skewing):

$$x_{image} = \begin{pmatrix} f_x & D \\ E & f_y \end{pmatrix} \begin{pmatrix} \frac{\theta_d x_h}{r} \\ \frac{\theta_d y_h}{r} \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \tag{2.14}$$

Given camera models are similar, but the most notably OpenCV uses more polynomial coefficients while Pix4D uses additional D and E coefficients to describe more affine transformations.

2.4 Non-linear Least Squares Optimisation

Non-linear least squares is a standard approach to approximate the solution of sets of equations in which there are fewer unknowns (x_i) than equations. It is adjusting parameters (β) of function $f(x, \beta)$ in order fit the best to the given actual values (y_i). The method aims to minimise residuals. The residuals (r_i) are the differences between actual values and values predicted by the model:

$$r_i = y_i - f(x_i, \beta) \tag{2.15}$$

Non-linear least squares method minimises the sum of squared residuals:

$$S = \sum_{i=1}^{m} r_i^2 \tag{2.16}$$

Minimisation is done in multiple iterations by calculating gradient and correctioing parameters accordingly:

$$\frac{\partial S}{\partial \beta_j} = 2 \sum_{i=1}^{m} r_i^2 \frac{\partial r_i}{\partial \beta_j} \tag{2.17}$$

For camera calibration, residuals are differences between observed projected points (X) on the image plane ($x_{observed}$) and calculated projections using camera calibration matrix (x_{image}). Parameters β of function $f(x, \beta)$ are intrinsic ($f_x,\ f_y,\ c_x,\ c_y,\ k_1,\ k_2...$) and extrinsic ($T_1, R_1, T_2, R_2...$) parameters of the camera. Function $f(x, \beta)$ for camera calibration is a function which determines coordinates of projected points on image plane (x_{image}) with desired camera model (e.g. OpenCV Camera Model or Pix4D Camera Model mentioned in previous sections (Section 2.3.1 and Section 2.3.2). We can use f_p to name the function and we can rephrase Equation 2.15 as follows:

$$r_i = x_{observed} - f_p(X, \beta) \tag{2.18}$$

There are multiple algorithms solving these equations iteratively (e.g. Gauss-Newton method) and all of them aim to converge as fast as possible to the solution and reach the global minimum.

2.4.1 Jacobian Matrix

Algorithms that implement non-linear least squares usually uses a Jacobian matrix. The Jacobian matrix consists of first-order partial derivatives of a vector-valued function. In general, the Jacobian matrix can be expressed as:

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \tag{2.19}$$

In the image calibration, m is a number of points and n is number of parameters of β. Therefore, we can construct Jacobian matrix as:

$$J = \begin{pmatrix} \frac{\partial f_{x1}}{\partial k_1} & \frac{\partial f_{x1}}{\partial k_2} & \frac{\partial f_{x1}}{\partial k_3} & \frac{\partial f_{x1}}{\partial c_x} & \cdots & \frac{\partial f_{x1}}{\partial T_1} & \frac{\partial f_{x1}}{\partial R_1} & \frac{\partial f_{x1}}{\partial T_2} & \frac{\partial f_{x1}}{\partial R_2} & \cdots \\ \frac{\partial f_{y1}}{\partial k_1} & \frac{\partial f_{y1}}{\partial k_2} & \frac{\partial f_{y1}}{\partial k_3} & \frac{\partial f_{y1}}{\partial c_x} & \cdots & \frac{\partial f_{y1}}{\partial T_1} & \frac{\partial f_{y1}}{\partial R_1} & \frac{\partial f_{y1}}{\partial T_2} & \frac{\partial f_{y1}}{\partial R_2} & \cdots \\ \frac{\partial f_{x2}}{\partial k_1} & \frac{\partial f_{x2}}{\partial k_2} & \frac{\partial f_{x2}}{\partial k_3} & \frac{\partial f_{x2}}{\partial c_x} & \cdots & \frac{\partial f_{x2}}{\partial T_1} & \frac{\partial f_{x2}}{\partial R_1} & \frac{\partial f_{x2}}{\partial T_2} & \frac{\partial f_{x2}}{\partial R_2} & \cdots \\ \frac{\partial f_{y2}}{\partial k_1} & \frac{\partial f_{y2}}{\partial k_2} & \frac{\partial f_{y2}}{\partial k_3} & \frac{\partial f_{y2}}{\partial c_x} & \cdots & \frac{\partial f_{y2}}{\partial T_1} & \frac{\partial f_{y2}}{\partial R_1} & \frac{\partial f_{y2}}{\partial T_2} & \frac{\partial f_{y2}}{\partial R_2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_{xn}}{\partial k_1} & \frac{\partial f_{xn}}{\partial k_2} & \frac{\partial f_{xn}}{\partial k_3} & \frac{\partial f_{xn}}{\partial c_x} & \cdots & \frac{\partial f_{xn}}{\partial T_1} & \frac{\partial f_{xn}}{\partial R_1} & \frac{\partial f_{xn}}{\partial T_2} & \frac{\partial f_{xn}}{\partial R_2} & \cdots \\ \frac{\partial f_{yn}}{\partial k_1} & \frac{\partial f_{yn}}{\partial k_2} & \frac{\partial f_{yn}}{\partial k_3} & \frac{\partial f_{yn}}{\partial c_x} & \cdots & \frac{\partial f_{yn}}{\partial T_1} & \frac{\partial f_{yn}}{\partial R_1} & \frac{\partial f_{yn}}{\partial T_2} & \frac{\partial f_{yn}}{\partial R_2} & \cdots \end{pmatrix} \tag{2.20}$$

In the Jacobian matrix, only a fraction of parameters are shown, missing parameters are intrinsic parameters and extrinsic parameters of all observations. The number of columns is equal to the number of intrinsic and extrinsic parameters of the camera and the number of rows is equal to the number of points of all observations.

Partial derivatives can be obtained analytically, but in this project, only numerical solution is used. The general expression for calculating numerical partial derivative can be expressed as:

$$\frac{\partial f}{\partial x_n} \approx \frac{f(x_1, x_2, ...x_n + \epsilon, x_{n+1}, ...x_m) - f(x_1, x_2, ...x_n - \epsilon, x_{n+1}, ...x_m)}{2\epsilon} \tag{2.21}$$

The Jacobian matrix 2.20 is calculated and β parameters adjusted accroding to partial derivatives in every iteration until determination criteria is met.

2.4.2 Covariance Matrix

As we have Jacobian matrix we can easily calculate covariance matrix as follows:

$$K = (J^T J)^{-1} \tag{2.22}$$

The covariance matrix is a matrix whos elements at i, j positions determine covariance between elements i, j. The variance of β parameters is placed on a diagonal of the covariance matrix. These values are important for further evaluation of the algorithms.

2.4.3 Correlation Matrix

Correlation mostly explains how much is a pair of variables linearly related. It can be expressed through covariance matrix (K) as follows:

$$N = diagonal(K)^{1/2} \tag{2.23}$$

$$C = NKN \tag{2.24}$$

Further, in the project, it will be extensively used to describe the correlation between two parameters of the camera model. In this project, the goal is to decrease correlation as it is usually a sign of inaccurately determined parameters.

Chapter 3

Simulation

Obtaining measurements from real-world is not reliable, therefore it may lead to wrong assumptions. However, simulation provides controllable environments and completely accurate values. In this project, calibration algorithms are firstly evaluated with simulated parameters. For this purpose, visualisation of the parameters (Section 3.1) and building blocks for generating different testing environments (Section 3.2) are developed.

3.1 Visualisation

In the first stage of simulation, a simple visualisation is created. Based on given translations (T) and rotations (R) of a camera in respect to the world, it draws 3D camera coordinate system (see Fig. 3.1).

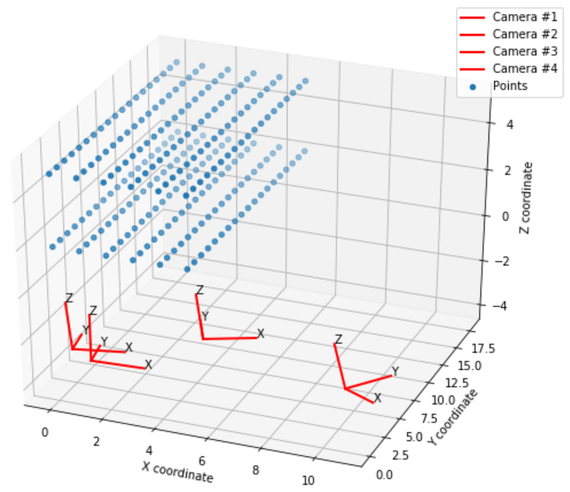


Figure 3.1: An example of 3D visualisation of cameras and points

Position of the camera in the world coordinate system is equal to $-T$, but the coordinate system of the cameras is determined as:

$$X_{cx} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} R^T - T \tag{3.1}$$

$$X_{cy} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} R^T - T \tag{3.2}$$

$$X_{cz} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} R^T - T \tag{3.3}$$

Camera #	T_x	T_y	T_z	R_ω	R_ϕ	R_κ
1	-9.5	-5.7	4	0.1	-0.2	-0.8
2	-4	-6	3	0.1	-0.1	0.3
3	-1	-1	3	0.0	0.0	0.1
4	0	-2	3	0.0	-0.1	0.1

Table 3.1: Rotation and translations of the cameras in respect to world in Fig. 3.1

In Fig. 3.1 4 cameras are placed in arbitrary positions (check Table 3.1) with arbitrary rotations. The only constraint was that the cameras are pointed to the points and that all points can be projected in the camera frame. Even though we needed to manually determine rotations and translations of cameras it helped us to visually verify our extrinsic parameters.

3.2 Building Blocks for Creating Layouts

In order to automatically generate and test different constellations in which all cameras are pointed at the given points, software building blocks are created. The purpose of those building blocks is to automatically find the optimal position of cameras with respect to points. In this case, "optimal" refers to obtaining calibration parameters that are not highly correlated and generate small residuals.

First we define a function $f_R(\omega, \phi, \kappa)$ which creates rotation matrix out of rotations around x axis (ω), y axis (ϕ) and z axis (κ):

$$f_R(\omega, \phi, \kappa) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\omega) & -sin(\omega) \\ 0 & sin(\omega) & cos(\omega) \end{pmatrix} \begin{pmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{pmatrix} \begin{pmatrix} cos(\kappa) & -sin(\kappa) & 0 \\ sin(\kappa) & cos(\kappa) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.4}$$

In the rest of the text this notation (ω), (ϕ) and (κ) will be used to create rotation matrices. Using just explained $f_R(\omega, \phi, \kappa)$ function we can determine translation of cameras:

$$T = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} f_R(0, \alpha, \gamma_i) \tag{3.5}$$

This will determine locations of N cameras in a circle around given origin O. Number of cameras N depends on difference of two successive angles γ:

$$N = \left\lfloor \frac{2\pi}{\gamma_{i+1} - \gamma_i} \right\rfloor \tag{3.6}$$

Angle α is a slope of between camera position, origin O and world plane where the given points are located. And, d is a distance between a camera and the origin O.

Next, the rotation of the camera has to be determined. First, we determine x_c, y_c and z_c one vectors that describe camera's coordinate system as following:

$$\vec{z_c} = \frac{T}{||T||} \tag{3.7}$$

$$\vec{x_c} = \begin{pmatrix} \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix} * \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 \end{pmatrix} \tag{3.8}$$

$$\vec{y_c} = \vec{x_c} \times \vec{z_c} \tag{3.9}$$

where angle θ is rotation of the camera around the camera's z axis.

Then, we can construct rotation matrix (R) as follows:

$$R = \begin{pmatrix} \vec{x}_0 & \vec{y}_0 & \vec{z}_0 \\ \vec{x}_1 & \vec{y}_1 & \vec{z}_1 \\ \vec{x}_2 & \vec{y}_2 & \vec{z}_2 \end{pmatrix} \tag{3.10}$$

Finally, we have a rotation matrix and we Euler angles can be extracted. For this purpose rotation matrix factorisation function is needed (f_R^{-1}) [4]:

$$f_R^{-1}(R) = \begin{cases} R_{1,3} < 1 \land R_{1,3} > -1 & \begin{pmatrix} atan(\frac{-R_{2,3}}{R_{3,3}}) \\ asin(R_{13}) \\ atan(\frac{-R_{1,2}}{R_{1,1}}) \end{pmatrix} \\ R_{1,3} < 1 \land R_{1,3} = -1 & \begin{pmatrix} -atan(\frac{R_{2,1}}{R_{2,2}}) \\ -\pi/2 \\ 0 \end{pmatrix} \\ R_{1,3} = 1 & \begin{pmatrix} -atan(\frac{R_{2,1}}{R_{2,2}}) \\ \pi/2 \\ 0 \end{pmatrix} \end{cases} \tag{3.11}$$

Therefore, Euler angles in ω, ϕ, κ notation are obtained as:

$$\begin{pmatrix} \omega \\ \phi \\ \kappa \end{pmatrix} = f_R^{-1}(R) \tag{3.12}$$

Obtained rotation angles and translations can be plugged into the visualisation (explained in the previous section, Section 3.1) in order to visualise the layout (see Fig. 3.2).

# cameras	d	α	$\gamma_{i+1} - \gamma_i$	# plains	plain size
7	5	$\pi/6$	0.9	2	6×6

Table 3.2: Required parameters for generating layout in Fig. 3.2

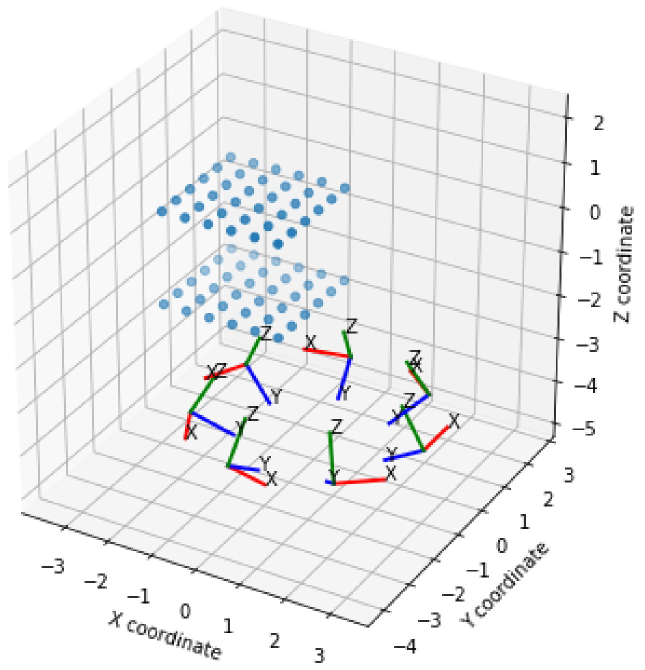


Figure 3.2: Example of layout with 7 cameras pointed towards two plains of points

This process will be very helpful for the experiments described in the following chapters. It allows automatic generation of layouts with an arbitrary number of cameras and configurable rotations (R) and translations (T).

Chapter 4

Calibration

In this chapter process of the camera, calibration will be explained. Firstly, we have to obtain a data set of points in the real world and the points projected on the image plane. Method used for the obtaining data set is explained the first section (Section 4.1). The obtained data set is then used for calibration of two cameras independently (Section 4.1). After the extrinsic and intrinsic parameters are estimated the parameters are used to calibrate rotation between dual cameras (Section 4.3). The software required for calibration is then packed into a reusable module (Section 4.4).

4.1 Chessboard Corner Extraction

One of the most popular ways to estimate the position of projected points from real-world to the image plane is by using chessboard [16]. Many scientific contributions are made in this area [6, 14] and many open-source robust tools for corner extraction are published [13]. Those were the main reasons for choosing this method.

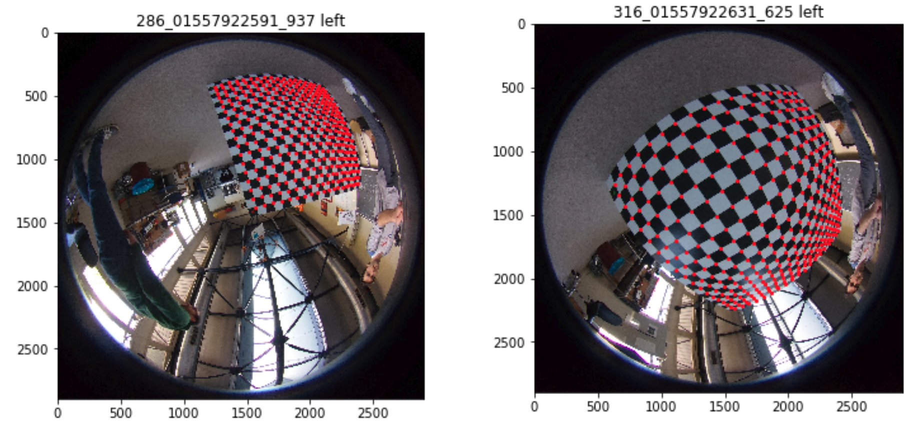


Figure 4.1: Example of extracted corners

In Fig. 4.1 an example of extracted corners is given. Extracted corners are shown as red dots. The chessboards are labelled with coloured squares, so we can keep track of the chessboard orientation. It is important to keep the order of the corners of all images, so we can match the points later in the optimisation. The first corner on all images is located next to the red square and the following corner is a neighbouring corner located on the same side as the green square. Therefore, the corners are ordered from the red square towards the green square until the end of the chessboard is reached. The sequence is continued again from the red square towards the green square until all corners are found.

4.1.1 Semi-automatic Tool for Highly Distorted Images

Our initial chessboard was 6 $\times$ 9 and OpenCV was detecting the corners very accurate. However, as we introduced chessboard with 19 $\times$ 19 corners and started taking highly distorted images of the chessboards, OpenCV wasn't effective anymore. Therefore, a semi-automatic tool for extracting corners from highly distorted images is developed (Fig. 4.2).

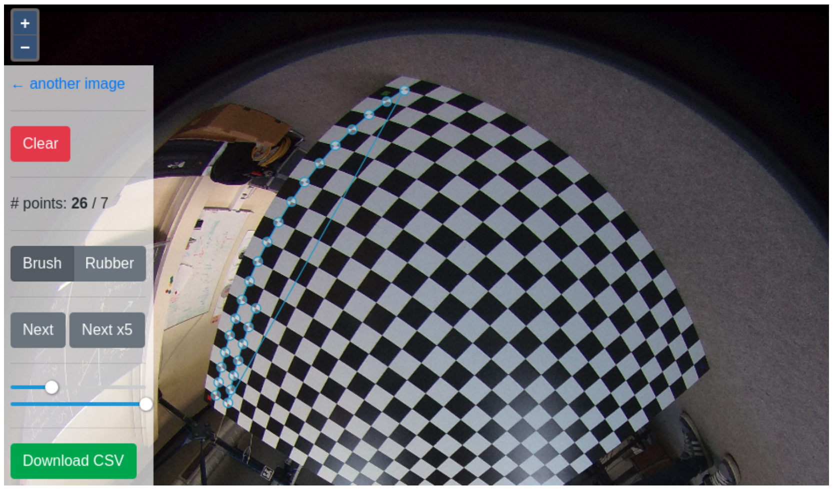


Figure 4.2: Screenshot of the tool for corner extraction

The tool has an intuitive, easy-to-use, user interface. It is semi-automatic which means it can automatically find corners but it still needs to be guided. The tool helped us to extract corners from many highly distorted images.

4.2 Single Camera Calibration

After data points ($x_{observed}$) are collected with chessboard corner extraction method enough data for calibration is collected. Fisheye camera models (Section 2.3) and non-linear least squares (Section 2.4) are utilised for an implementation of the camera calibration process. In this section, the camera calibration process will be described from an implementation point of view.

4.2.1 Initial Parameters

Algorithms that implement non-linear least squares optimisation are greedy and often converge to a local minimum. Therefore, it is important to properly determine the initial parameters. In our experiments, we were manually determined initial parameters based on observations in the real world. This approach was good enough for our use-case, but more automatic techniques are also available.

4.2.2 Non-linear Least Square Optimisation

After initial parameters are determined, we can proceed with non-linear least square optimisation. As described in Section 2.4.1 Jacobian matrix is constructed. In this section optimisation with Pix4D camera model (Section 2.3.2) is used.

Number of unknown intrinsic parameters for Pix4D camera model is 9 - $k_1, k_2, k_3, f_x, f_y, D, E, c_x, c_y$. Number of unknown extrinsic parameters depends on number of observations, for each observation translation (T_i) and rotation (R_i) have to be estimated. Therefore, number of extrinsic parameters is equal to $6 \times M$ where M is number of observations. Jacobian matrix size for Pix4D camera model and our chessboard (19×19 corners) is equal to $(M19 * 19) \times (9 + 6M)$.

To solve nonlinear least-squares problem *scipy.optimize.least_squares* function from SciPy is applied. The function is searching for optimal intrinsic and extrinsic camera parameters until at least one of the determination conditions is met. In our experiments terminations conditions are:

- Maximal number of iterations of 100 is reached.
- Cost function reached value less than 1^{-8}.
- Independent variables change is less than 1^{-8}.

4.2.3 Performance Optimisation

In our experiments, non-linear least square optimisation needed a lot of time to converge to the solution. Therefore, we applied a few techniques to decrease execution time.

Jacobian Matrix Sparsity

We discovered that calculating derivatives takes a lot of processing time. For each partial derivative in Jacobian matrix projection function f_p has to be called twice (see Equation 2.21). One of the solutions to reduce the number of f_p calls is to exploit Jacobian matrix sparsity. Partial derivatives in the Jacobian matrix that correspond to $R_i \wedge T_i$ where $i \neq j$ of observation j are equal to 0. Since we are aware of it we can construct a Jacobian matrix as in Table 4.1.

Img	Params Points	q_1	q_2	q_3	F	X_c	R_1	T_1	R_2	T_2	R_3	T_3	...
#1	P_1								X	X	X	X	X
	P_2								X	X	X	X	X
	P_3								X	X	X	X	X
	...								X	X	X	X	X
#2	P_1						X	X			X	X	X
	P_2						X	X			X	X	X
	P_3						X	X			X	X	X
	...						X	X			X	X	X
#3	P_1						X	X	X	X			X
	P_2						X	X	X	X			X
	P_3						X	X	X	X			X
	...						X	X	X	X			X

Table 4.1: Jacobian Matrix Sparsity

In Table 4.1 cells with X are partial derivatives for which we know the result is 0. Thus, the number of functional calls f_p is much smaller especially if calibration uses a lot of images.

Nonlinear Least-Squares Algorithm

Multiple algorithms that implement nonlinear least-squares optimisation are tested, but the best results are obtained with LSMR algorithm [5]. The algorithm is optimised for sparse and large Jacobian matrices which makes it suitable for our use-case.

Single Rotation Matrix per Image

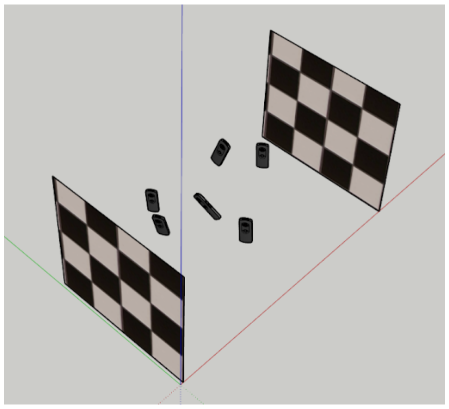
For each point, a rotation matrix is calculated. This is identified as an additional bottleneck since is called many times. Therefore rotation matrix is calculated once per each image and saved in memory. It enables the algorithm to run even faster.

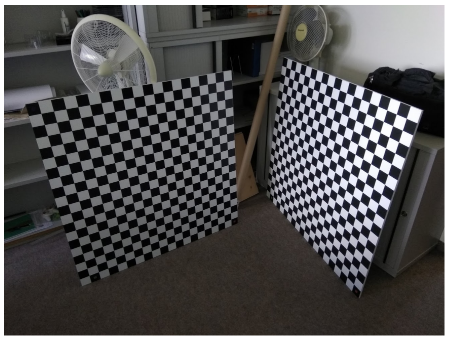
Additional Optimisation Techniques

Additional speed can be gained by calculating Jacobian analytically. This would reduce the number of f_p calls and the results would be more accurate. However, it would make the software less general since analytical solutions should be calculated after each change in the camera model. Nonetheless, the algorithms are implemented in Python which is slow compared to languages that can be compiled ahead-of-time.

4.3 Dual Camera Calibration

As mentioned before, Ricoh Theta V has two cameras. The important part of the project is develop a technique for determining rotation between two cameras. The process of acquiring data for dual camera calibration is shown in Fig. 4.3a.


(a) Process of dual camera calibration


(b) Actual chessboards created for calibration

In Fig. 4.3a two chessboards are positioned in parallel (as parallel as possible) to each other. In order to obtain rotation between dual cameras, multiple images are taken between two chessboards. For each image, from each camera, rotations in respect to a world frame (R_i) are determined as explained in Section 4.2. Those rotations are later used to express rotation between two cameras (see Fig. 4.4).

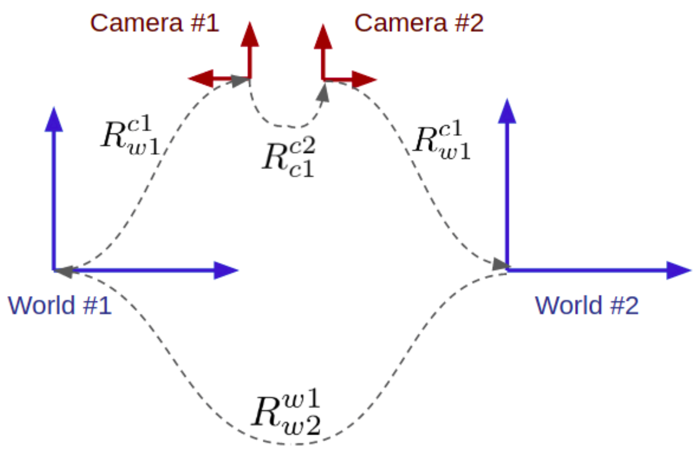


Figure 4.4: Rotation between two cameras

In Fig. 4.4 simplified 2D view of camera rotation is shown. Based on known rotations R_{w1}^{c1} and R_{w2}^{c2} for each image and partially know rotation R_{w2}^{w1} we should be able to determine R_{c1}^{c2}. Our goal is to minimise ΔR of all images:

$$\Delta R = R_{w1}^{c1} R_{c1}^{c2} (R_{c2}^{w2})^T (R_{w2}^{w1}) \tag{4.1}$$

4.3.1 Initial Parameters

Just like for single image calibration, bad initial parameters may lead to completely wrong results. Therefore, initial parameters are again determined by analysing how objects are placed in the real world. An example of the initial parameters used in our calibration:

$$\begin{pmatrix} \omega \\ \phi \\ \kappa \end{pmatrix}_{c1}^{c2} = \begin{pmatrix} 0 \\ \pi \\ 0 \end{pmatrix} \tag{4.2}$$

$$\begin{pmatrix} \omega \\ \phi \\ \kappa \end{pmatrix}_{w1}^{w2} = \begin{pmatrix} 0 \\ \pi \\ 0 \end{pmatrix} \tag{4.3}$$

4.3.2 Non-linear Least Square Optimisation

Since this is again optimisation problem we can construct Jacobian matrix as:

$$J = \begin{pmatrix}
\frac{\partial fr_{R1}}{\partial \omega_{c1c2}} & \frac{\partial fr_{R1}}{\partial \phi_{c1c2}} & \frac{\partial fr_{R1}}{\partial \kappa_{c1c2}} & \frac{\partial fr_{R1}}{\partial \omega_{w1w2}} & \frac{\partial fr_{R1}}{\partial \phi_{w1w2}} & \frac{\partial fr_{R1}}{\partial \kappa_{w1w2}} \\
\frac{\partial fr_{T1}}{\partial \omega_{c1c2}} & \frac{\partial fr_{T1}}{\partial \phi_{c1c2}} & \frac{\partial fr_{T1}}{\partial \kappa_{c1c2}} & \frac{\partial fr_{T1}}{\partial \omega_{w1w2}} & \frac{\partial fr_{T1}}{\partial \phi_{w1w2}} & \frac{\partial fr_{T1}}{\partial \kappa_{w1w2}} \\
\frac{\partial fr_{R2}}{\partial \omega_{c1c2}} & \frac{\partial fr_{R2}}{\partial \phi_{c1c2}} & \frac{\partial fr_{R1}}{\partial \kappa_{c1c2}} & \frac{\partial fr_{R2}}{\partial \omega_{w1w2}} & \frac{\partial fr_{R2}}{\partial \phi_{w1w2}} & \frac{\partial fr_{R2}}{\partial \kappa_{w1w2}} \\
\frac{\partial fr_{T2}}{\partial \omega_{c1c2}} & \frac{\partial fr_{T2}}{\partial \phi_{c1c2}} & \frac{\partial fr_{T2}}{\partial \kappa_{c1c2}} & \frac{\partial fr_{T2}}{\partial \omega_{w1w2}} & \frac{\partial fr_{T2}}{\partial \phi_{w1w2}} & \frac{\partial fr_{T2}}{\partial \kappa_{w1w2}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial fr_{Rm}}{\partial \omega_{c1c2}} & \frac{\partial fr_{Rm}}{\partial \phi_{c1c2}} & \frac{\partial fr_{Rm}}{\partial \kappa_{c1c2}} & \frac{\partial fr_{Rm}}{\partial \omega_{w1w2}} & \frac{\partial fr_{Rm}}{\partial \phi_{w1w2}} & \frac{\partial fr_{Rm}}{\partial \kappa_{w1w2}} \\
\frac{\partial fr_{Tm}}{\partial \omega_{c1c2}} & \frac{\partial fr_{Tm}}{\partial \phi_{c1c2}} & \frac{\partial fr_{Tm}}{\partial \kappa_{c1c2}} & \frac{\partial fr_{Tm}}{\partial \omega_{w1w2}} & \frac{\partial fr_{Tm}}{\partial \phi_{w1w2}} & \frac{\partial fr_{Tm}}{\partial \kappa_{w1w2}}
\end{pmatrix} \tag{4.4}$$

Just like for a single camera calibration we use *scipy.optimize.least_squares* function from SciPy to find the minimum. However, the estimated parameters are highly correlated (see Chapter 5) and additional techniques had to be applied.

Priors

By adding priors we are able to increase certainty to initial values of some parameters. Since the chessboards are mostly parallel we can increase certainty of rotation between two worlds (R_{w2}^{w1}, see Fig. 4.4). Therefore:

$$\lambda f_R^{-1}(R_{w2}^{w1}(\widetilde{R_{w2}^{w1}})^T) = 0 \tag{4.5}$$

where λ is a coefficient which describes magnitude of certainty of R_{w2}^{w1}. With this technique, correlation is significantly decreased.

4.4 Calibration Software Module

All functions required for visualisation, optimisation, models and layout generation are packaged into a Python module and can be reused. The structure of the Python package is shown in Fig. 4.5.

- topocalib
  - layout.py Building blocks for generating layouts
    - generate_layout() Generates layout
    - show_layout() Shows layout
  - optimiser.py Optimisation methods for calibration
    - optimise_pix4d() Calibration using Pix4D camera model
    - optimise() Calibration using OpenCV camera model
  - visualise.py Visualisation of calibration matrix, residuals, histograms...
    - show_correlation_matrix() Displays correlation matrix
    - show_residuals() Displays residuals and histograml
  - model.py Common functions for all models
    - get_rotation() Converts ω, ϕ, κ to rotation matrix
    - get_angles() Converts rotation matrix to ω, ϕ, κ
  - model_cv.py Implements OpenCV camera model
    - generate_projections() Projects points to the image plane
  - model_pix4d.py Implements Pix4D camera model
    - generate_projections() Projects points to the image plane

Figure 4.5: The most important functions in the Python module

The Python module is then used in Jupyter Lab to verify, compare and visualise the calibration process.

Chapter 5

Results

The projected is done in 3 parts. The first goal was to find optimal constellation for camera calibration. This process is described in Section 3.2 and results will be shown in Section 5.1. Afterwards, the results obtained with the calibration of a single camera will be shown in Section 5.2. Finally, the results for dual camera calibration will be described in Section 5.3.

5.1 Optimal Calibration Layout

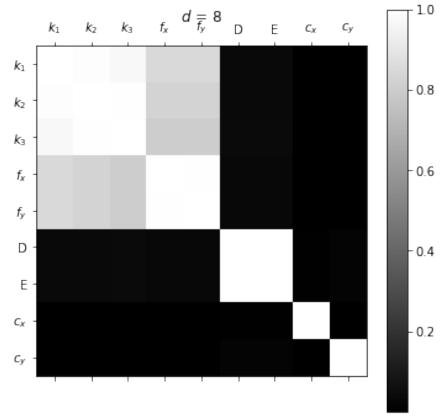
In this process, we wanted to simulate different calibration approaches in order to get accurate camera parameters. The main goals were to reduce the correlation between parameters and to reduce residuals. The reason for those experiments is to determine the best technique for acquiring real data. For that purpose the following parameters are alternated:

- Distance d (see Equation 3.5) in range of 2 — 15.
- Angle α (see Equation 3.5) in range of $\frac{\pi}{14}$ — $\frac{\pi}{4}$ radians.
- Different plain of points setup, $2 \times 6 \times 6$ and $1 \times 8 \times 9$.

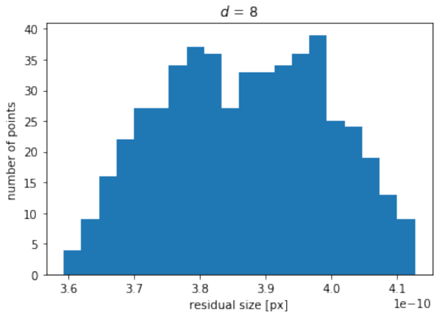
Note that d is unitless, which means we can scale distances as far as other distances are also proportionally scaled.

5.1.1 Distance between cameras and origin O

In this experiment parameter d is changed and the correlation matrix and histogram of residuals are observed.

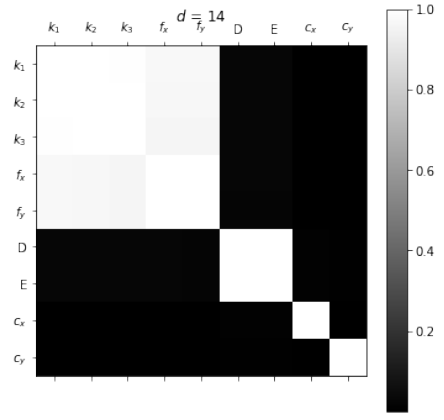


(a) Correlation matrix of intrinsic parameters

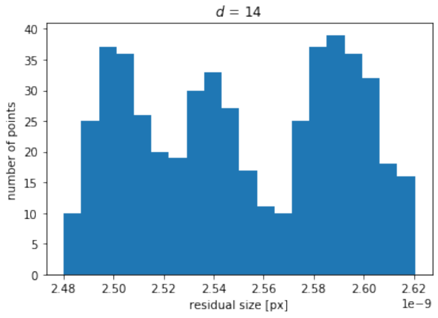


(b) Histogram of residuals

Figure 5.1: Results obtained with a layout with the following configuration $\alpha = \frac{\pi}{6}$, $d = 8$, camera positions = 6, number of plains = 2, size of plains = 6×6



(a) Correlation matrix of intrinsic parameters



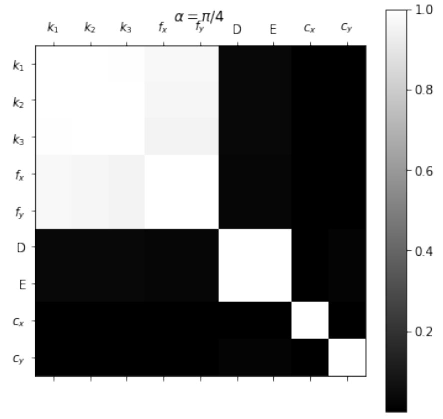
(b) Histogram of residuals

Figure 5.2: Results obtained with a layout with the following configuration $\alpha = \frac{\pi}{6}$, $d = 14$, camera positions = 6, number of plains = 2, size of plains = 6×6

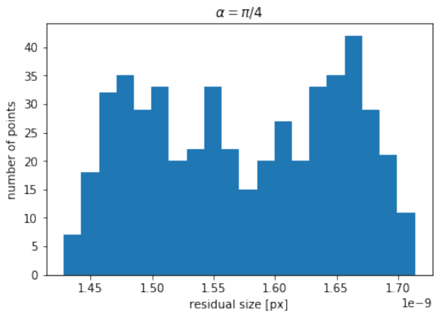
In Fig. 5.1a and Fig. 5.2a are shown correlation matrices for which d is equal to 8 and 14 respectively. Dark colour represents parameters that are not correlated and opposite for correlated parameters. From both correlation matrices we conclude that D and E are correlated, also f_x, f_y, k_1, k_2 and k_3 are correlated to each other. However, for $d = 8$ parameters f_x and f_y are slightly less correlated than for $d = 14$. Since we are looking for solutions that are less correlated we will choose $d = 8$. Also, another important measurement is a histogram of residuals, and the residuals are generally smaller for $d = 8$ which probably means $d = 8$ is a more appropriate distance.

5.1.2 Different values of angle α

Again, we want to see the effect of angle α on correlation matrix and residuals.



(a) Correlation matrix of intrinsic parameters



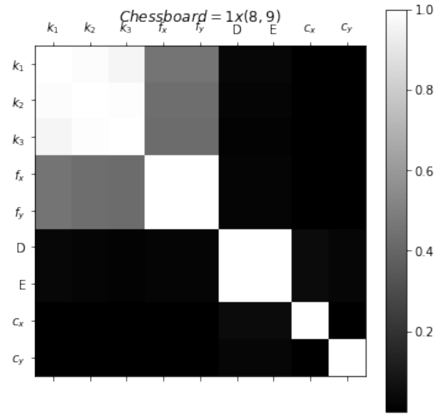
(b) Histogram of residuals

Figure 5.3: Results obtained with a layout with the following configuration $\alpha = \frac{\pi}{4}$, $d = 8$, camera positions $= 6$, number of plains $= 2$, size of plains $= 6 \times 6$

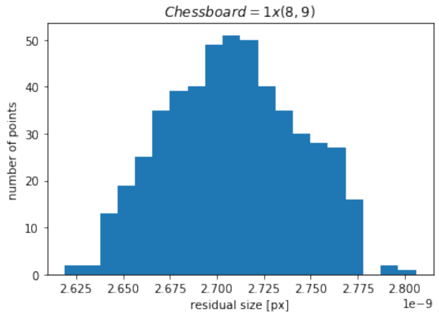
In the Fig. 5.3 angle α is increased and it caused bigger correlation between f_x and f_y even though a distance d is still equal to 8. The big angle α has negative effect on residuals as well. In experiments with smaller angles ($\leq \alpha$) correlation is not increased.

5.1.3 Different locations of points

In this experiment we wanted to investigate if it is better to have m points in two levels or m points in a single level (bigger area).

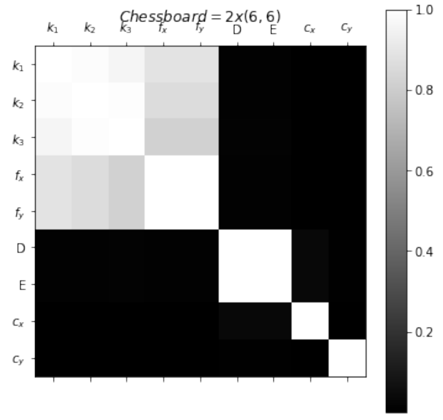


(a) Correlation matrix of intrinsic parameters

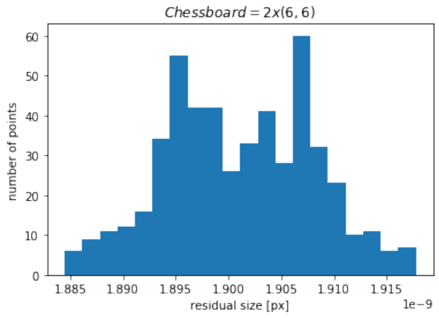


(b) Histogram of residuals

Figure 5.4: Results obtained with a layout with the following configuration $\alpha = \frac{\pi}{6}$, $d = 8$, camera positions = 6, **number of plains = 1, size of plains = 8 $\times$ 9**



(a) Correlation matrix of intrinsic parameters



(b) Histogram of residuals

Figure 5.5: Results obtained with a layout with the following configuration $\alpha = \frac{\pi}{6}$, $d = 8$, camera positions = 6, **number of plains = 2, size of plains = 6 $\times$ 6**

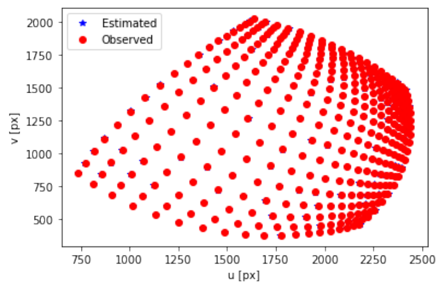
Based on correlation in Fig. 5.4a and Fig. 5.5a bigger definitely has a better effect on correlation matrix. Even though number of points is about the same points spread across wider are more appropriate for the fisheye camera calibration.

5.1.4 Gaussian noise

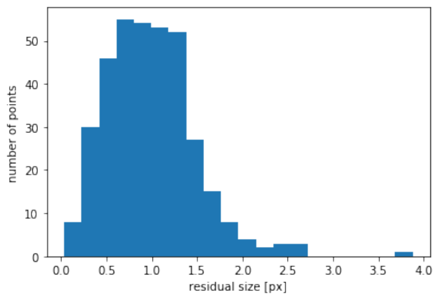
A few more experiments is performed with Gaussian noise. Gaussian noise is added to the cameras translations (range 0.1 — 10), points positions (range 0.1 — 10) and rotations of the cameras (range 0.1 — 0.5). Neither of those changes haven't had significant influence on correlation or residuals.

5.2 Intrinsic and Extrinsic Parameters of Single Camera

Here, results of a single fisheye camera calibration will be shown and analysed. For the calibration of the cameras Pix4D model is used, but parameters f_y, D and E are disabled since those parameters were highly correlated.

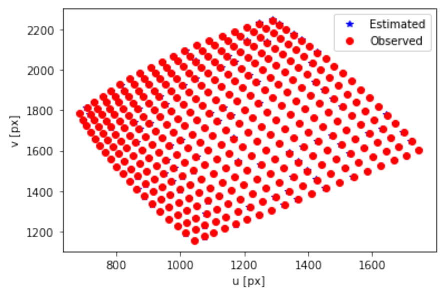


(a) Observed and estimated values shown on the same picture

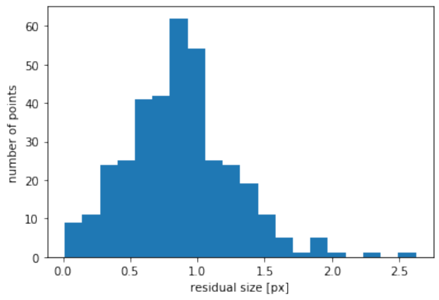


(b) Histogram of residuals

Figure 5.6: A sample of the residuals of the left camera (left lens of RICOH-T camera)



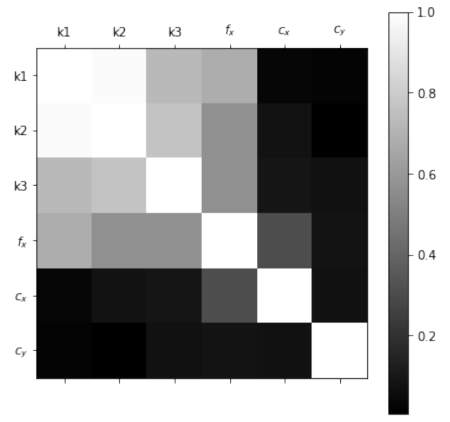
(a) Observed and estimated values shown on the same picture



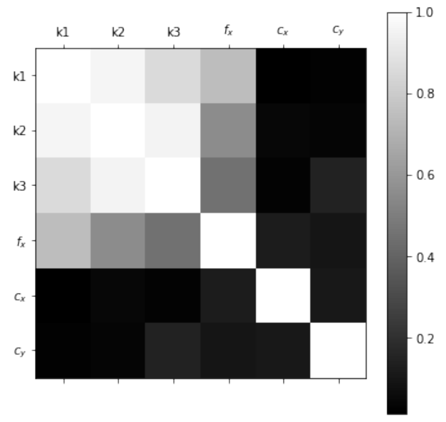
(b) Histogram of residuals

Figure 5.7: A sample of the residuals of the right camera (right lens of RICOH-T camera)

In Fig. 5.6a and 5.7 are shown residuals of left and right lens respectively. In the figures only a single sample per camera is shown, but the residuals of the other images are very similar. Residuals are mostly around 1px with a very few outliers.



(a) Correlation of the intrinsic parameters of the left image



(b) Correlation of the intrinsic parameters of the right image

Figure 5.8: Correlation matrices

Correlation matrices the left and right cameras show high correlation between polynomial coefficients. Also, slightly less correlation is present for f_x and f_y. Similarly, in Section 5.1 we were able to reduce the correlation (of f_x and f_y) but it was still present.

In addition, we can create a correlation matrix for extrinsic parameters as well (see Fig. 5.9).

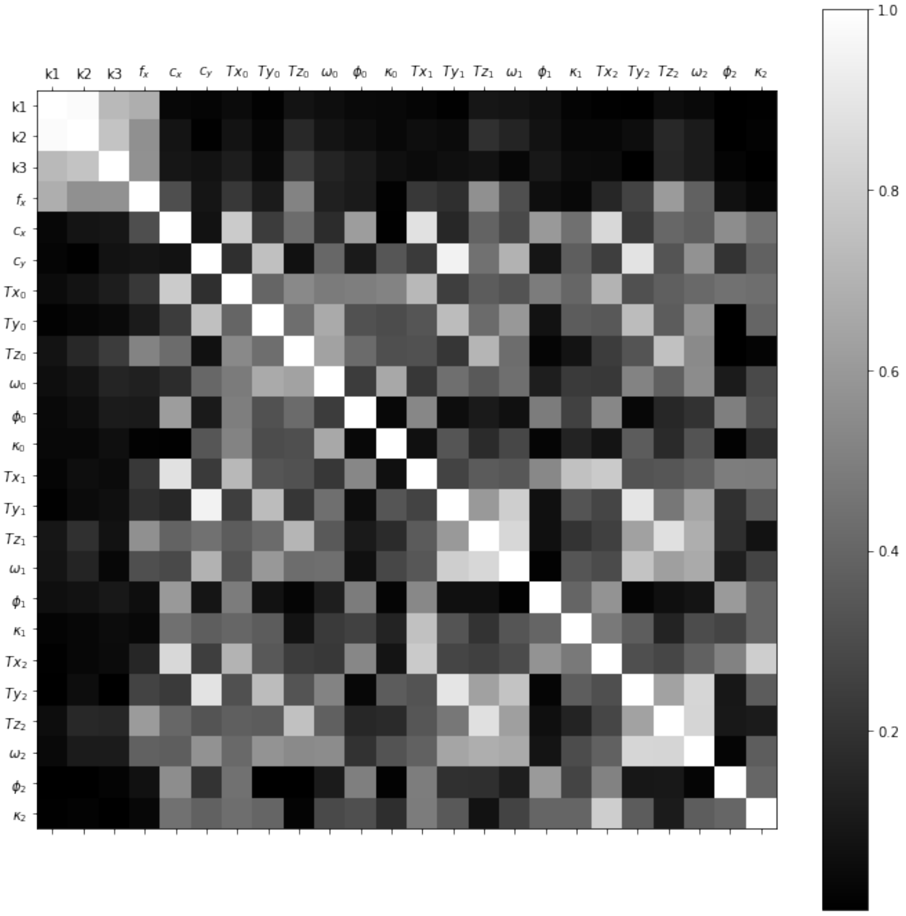


Figure 5.9: Correlation matrix of intrinsic and some extrinsic parameters

Obtained calibration matrices in this experiment are:

$$K_{left} = \begin{pmatrix} 1333.73277 & 0 & 1443.87723 \\ 0 & 1333.73277 & 1443.38526 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.1}$$

$$D_{left} = \begin{pmatrix} -0.08045513 \\ 0.14854105 \\ -0.07786538 \end{pmatrix} \tag{5.2}$$

$$K_{right} = \begin{pmatrix} 1361.56688 & 0 & 1440.86097 \\ 0 & 1361.56688 & 1443.30730 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

$$D_{right} = \begin{pmatrix} -0.02956731 \\ 0.03790681 \\ -0.05503708 \end{pmatrix} \tag{5.4}$$

After calibration done, extrinsic parameters are determine we are able to reconstruct positions and rotations of the cameras (see Fig. 5.10).

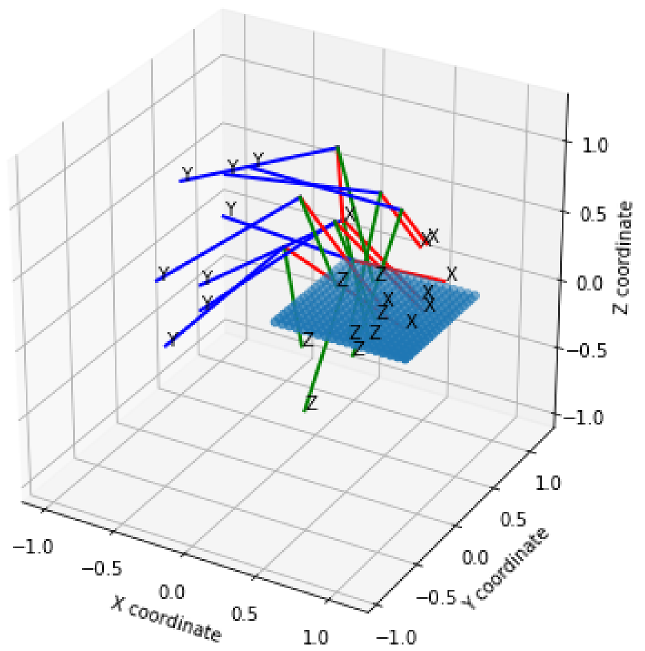
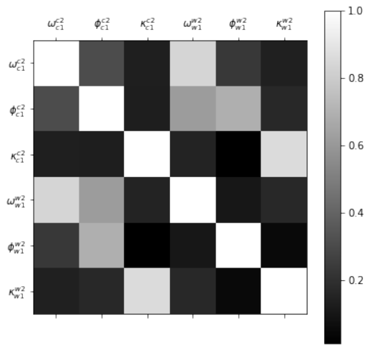


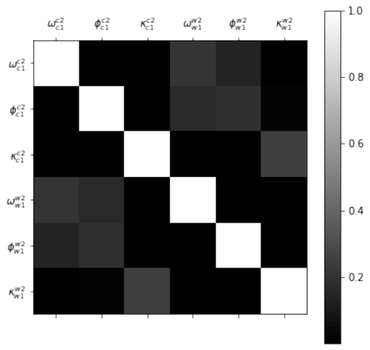
Figure 5.10: Position of cameras based on estimated extrinsic parameters

5.3 Dual Camera Calibration

Finally, since rotations (R_i) of the cameras in respect to the worlds are available it is possible to determine rotation between cameras (R_{c1}^{c2}, see Equation 4.1).



(a) Correlation matrix for $\lambda = 1$



(b) Correlation matrix for $\lambda = 10$

Figure 5.11: Correlation matrices for different values of λ

Unfortunately, as already mentioned in Section 4.3.2 rotation between world and rotation between cameras are highly correlated (see Fig. 5.11a). However, after λ is increased (e.g. to 10) correlations are decreased (see Fig. 5.11b).

With $\lambda = 10$ results shown in Equation 5.5 are obtained.

$$\begin{pmatrix} \omega_{c1}^{c2} \\ \phi_{c1}^{c2} \\ \kappa_{c1}^{c2} \\ \omega_{w1}^{w2} \\ \phi_{w1}^{w2} \\ \kappa_{w1}^{w2} \end{pmatrix} = \begin{pmatrix} 0.0424 \\ 3.1551 \\ 0.0062 \\ 0 \\ 3.1416 \\ 0.0001 \end{pmatrix} \tag{5.5}$$

After further analysis, it is discovered that a variance of rotation angles (of R_{c1}^{c2}) is quite high. This shows the uncertainty of the estimated rotation.

$$var(\begin{pmatrix} \omega_{c1}^{c2} \\ \phi_{c1}^{c2} \\ \kappa_{c1}^{c2} \\ \omega_{w1}^{w2} \\ \phi_{w1}^{w2} \\ \kappa_{w1}^{w2} \end{pmatrix})^{-0.5} = \begin{pmatrix} 0.37113658 \\ 0.35845027 \\ 0.36354734 \\ 0.09974951 \\ 0.09915489 \\ 0.09932535 \end{pmatrix} \tag{5.6}$$

However, the variance can be decreased by increasing value of λ and physically putting two chessboards more parallel to each other.

Chapter 6

Conclusion

This project greatly helped me to understand photogrammetry and moreover, to understand it's importance. With this project, I understood the process of image creation and how to model and estimate the parameters of a camera. I believe this will be important for robotics projects in the future, especially the process of determining the extrinsic camera parameters.

The implementation is done in multiple stages, simulation, single camera calibration, dual camera calibration and in the end everything with real-data. This approach allowed me to slowly meet the problems and learn techniques to solve them.

Building blocks for layout generation are developed and optimal layouts are discovered. Many layouts are explored, but there is an opportunity for exploring more parameters. This supposed to be relatively easy to accomplish since building blocks are ready. This part helped me to get a better intuition on how to create a good data set for camera calibration.

Calibration of the independent fisheye cameras is done. Unfortunately, only 10 observations are used for the calibration since no efficient automatic corner extraction algorithm is found for our use-case. Less correlated parameters are possible to obtain by using more observations. This should be the first next step for the project continuation.

Rotation between the two cameras is also calculated. However, the variance is big indicates that the obtained results are not accurate enough. The more accurate result could be calculated if prior knowledge of rotation between chessboards is known at higher certainty. In addition, another way to improve the quality of the results would be by obtaining more data points.

Bibliography

[1] S. Aghayari et al. "GEOMETRIC CALIBRATION OF FULL SPHERICAL PANORAMIC RICOH-THETA CAMERA". en. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-1/W1 (May 2017), pp. 237–245. ISSN: 2194-9050. DOI: `10.5194/isprs-annals-IV-1-W1-237-2017`. URL: `http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-1-W1/237/2017/` (visited on 02/24/2019).

[2] L. Barazzetti, M. Previtali, and F. Roncoroni. "CAN WE USE LOW-COST 360 DEGREE CAMERAS TO CREATE ACCURATE 3D MODELS?" en. In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2 (May 2018), pp. 69–75. ISSN: 2194-9034. DOI: `10.5194/isprs-archives-XLII-2-69-2018`. URL: `https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2/69/2018/` (visited on 05/21/2019).

[3] Mariana Batista Campos et al. "Geometric model and assessment of a dual-fisheye imaging system". en. In: *The Photogrammetric Record* 33.162 (2018), pp. 243–263. ISSN: 1477-9730. DOI: `10.1111/phor.12240`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1111/phor.12240` (visited on 05/21/2019).

[4] David Eberly. "Euler Angle Formulas". en. In: (), p. 18.

[5] David Fong and Michael Saunders. "LSMR: An iterative algorithm for sparse least-squares problems". In: *International Journal of Heat and Mass Transfer* 55.9-10 (Apr. 2012). arXiv: 1006.0758, pp. 2636–2646. ISSN: 00179310. DOI: `10.1016/j.ijheatmasstransfer.2011.12.029`. URL: `http://arxiv.org/abs/1006.0758` (visited on 06/05/2019).

[6] Andreas Geiger et al. "Automatic camera and range sensor calibration using a single shot". en. In: *2012 IEEE International Conference on Robotics and Automation*. St Paul, MN, USA: IEEE, May 2012, pp. 3936–3943. ISBN: 978-1-4673-1405-3 978-1-4673-1403-9 978-1-4673-1578-4 978-1-4673-1404-6. DOI: `10.1109/ICRA.2012.6224570`. URL: `http://ieeexplore.ieee.org/document/6224570/` (visited on 05/26/2019).

[7] J. Heikkila and O. Silven. "A four-step camera calibration procedure with implicit image correction". en. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* San Juan, Puerto Rico: IEEE Comput. Soc, 1997, pp. 1106–1112. ISBN: 978-0-8186-7822-6. DOI: `10.1109/CVPR.1997.609468`. URL: `http://ieeexplore.ieee.org/document/609468/` (visited on 02/24/2019).

[8] T. Ho and M. Budagavi. "Dual-fisheye lens stitching for 360-degree imaging". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Mar. 2017, pp. 2172–2176. DOI: `10.1109/ICASSP.2017.7952541`.

[9] *How are the Internal and External Camera Parameters defined?* en-US. URL: `http://support.pix4d.com/hc/en-us/articles/202559089-How-are-the-Internal-and-External-Camera-Parameters-defined-` (visited on 06/03/2019).

[10] J. Kannala and S.S. Brandt. "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses". en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.8 (Aug. 2006), pp. 1335–1340. ISSN: 0162-8828, 2160-9292. DOI: `10.1109/TPAMI.2006.153`. URL: `http://ieeexplore.ieee.org/document/1642666/` (visited on 02/24/2019).

[11] *OpenCV: Camera Calibration and 3D Reconstruction.* URL: `https://docs.opencv.org/4.1.0/d9/d0c/group__calib3d.html` (visited on 06/04/2019).

[12] *OpenCV: Fisheye camera model.* URL: `https://docs.opencv.org/4.1.0/db/d58/group__calib3d__fisheye.html` (visited on 06/03/2019).

[13] *OpenCV: Harris Corner Detection.* URL: `https://docs.opencv.org/4.1.0/dc/d0d/tutorial_py_features_harris.html` (visited on 06/05/2019).

[14] M. Rufli, D. Scaramuzza, and R. Siegwart. "Automatic detection of checkerboards on blurred and distorted images". en. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems.* Nice: IEEE, Sept. 2008, pp. 3121–3126. ISBN: 978-1-4244-2057-5 978-1-4244-2058-2. DOI: `10.1109/IROS.2008.4650703`. URL: `http://ieeexplore.ieee.org/document/4650703/` (visited on 06/05/2019).

[15] J. Weng, P. Cohen, and M. Herniou. "Camera calibration with distortion models and accuracy evaluation". en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.10 (Oct. 1992), pp. 965–980. ISSN: 01628828. DOI: `10.1109/34.159901`. URL: `http://ieeexplore.ieee.org/document/159901/` (visited on 03/27/2019).

[16] Zhengyou Zhang. “A Flexible New Technique for Camera Calibration”. en-US. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (Dec. 2000). URL: `https://www.microsoft.com/en-us/research/publication/a-flexible-new-technique-for-camera-calibration/` (visited on 06/05/2019).