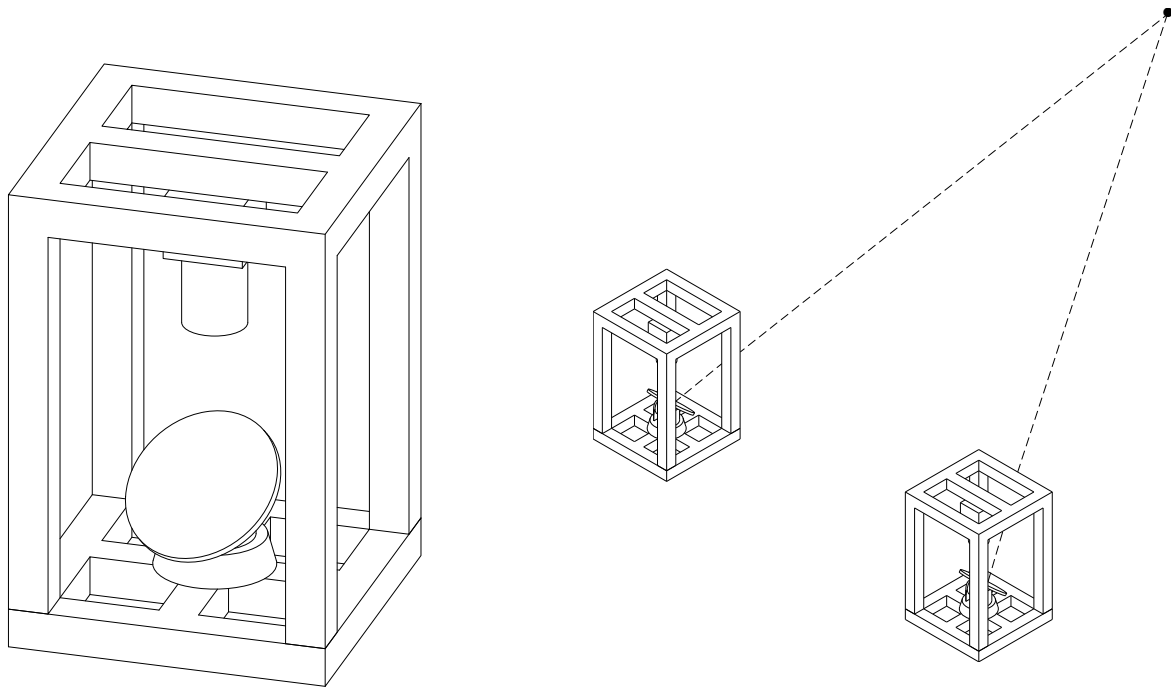




CHALMERS
UNIVERSITY OF TECHNOLOGY



Calibration of Multiple Non-Gimbal Mounted Camera-Mirror Systems

Master's thesis in Physics and Systems, Control and Mechatronics

Anna Lundqvist
Johan Wheeler

Department of Electrical Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

MASTER'S THESIS 2025

Calibration of Multiple Non-Gimbal Mounted Camera-Mirror Systems

Anna Lundqvist
Johan Wheeler



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Calibration of Multiple Non-Gimbal Mounted Camera-Mirror Systems

Anna Lundqvist
Johan Wheeler

© ANNA LUNDQVIST, JOHAN WHEELER, 2025.

Supervisors: Simon Arkeholt, SWEDISH DEFENCE RESEARCH AGENCY, FOI
Mikael Karelid, SWEDISH DEFENCE RESEARCH AGENCY, FOI
Examiner: Tomas McKelvey, DEPARTMENT OF ELECTRICAL ENGINEERING

Master's Thesis 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A schematic illustration of a single sensor rig and a the complete system consisting of two sensor rigs.

Gothenburg, Sweden 2025

Calibration of Multiple Non-Gimbal Mounted Camera-Mirror Systems

Anna Lundqvist and Johan Wheeler
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The primary objective of this Master's thesis is to investigate methods for calibrating multiple sensor rigs designed to track the position of an object in 3D space. Each rig consists of a statically mounted sensor directed downward at a controllable pan-tilt mirror, allowing for adjustment of the sensor's field of view. The calibration method compensates for misalignments both within individual rigs and between different rigs, and enables all rigs to observe the same 3D point. This thesis project focuses on calibrating a setup consisting of two rigs.

Based on a mathematical model of the system and measurement data collected with the sensor rigs, the positions and orientations of the rig components, which define the system parameters, are estimated. Once these parameters are known, the orientation required for one of the mirrors can be predicted so that its corresponding sensor observes the same 3D point as the other sensor. The parameter estimation is performed by minimizing a cost function, and two optimization methods have been evaluated for this purpose. The proposed calibration method has been tested on both simulated and real-world systems.

The performance of the parameter estimation was assessed in several aspects, including accuracy dependence on the number of calibration points, approximate angular errors in various calibration and test scenarios for both simulated and real-world systems, and some investigation of the sensitivity of the parameter estimation to variations in initial parameter estimates. Results show promising accuracy in some cases, with real-world tests indicating angular errors within $\pm 0.3^\circ$. However, a key limitation of the current method is its sensitivity to the initial parameter guess, which must be close to the true values for accurate parameter estimation. Addressing this limitation could improve the method's applicability.

Keywords: Multi-sensor calibration, Parameter estimation, Model based calibration, Non-gimbal mounted mirrors.

Acknowledgements

We would like to express our gratitude to our supervisors at FOI, Simon Arkeholt and Mikael Karelid, for their valuable guidance and support throughout the course of this thesis project. We are also thankful to our examiner at Chalmers, Tomas McKelvey, for providing us thoughtful input which helped us navigate the thesis process. Additionally, we appreciate our colleagues at FOI for their advice and encouragement, which contributed to a stimulating work environment. Finally, we would like to thank our partners, families, and friends for their continuous support during our time at Chalmers.

Anna Lundqvist and Johan Wheeler, Gothenburg, June 2025

List of Abbreviations

Below is the list of abbreviations that have been used throughout this thesis listed in alphabetical order:

DOF	Degrees of freedom.
PTU	Pan-tilt unit.
RPY	Roll, pitch and yaw.

Nomenclature

Below is the nomenclature that have been used throughout this thesis.

Model Vectors and Matrices

\mathbf{p}	Position vector of an object in 3D space.
\mathbf{R}	Rotation matrix representing the orientation of the object.
$\mathbf{M}_{\text{reflection}}$	Reflection matrix representing a mirror transformation.
$\hat{\mathbf{n}}$	Unit normal vector perpendicular to a surface or object.
\mathbf{I}	Identity matrix.
$\mathbf{R}_{xyz}(\phi_x, \phi_y, \phi_z)$	Rotation matrix constructed from Euler angles ϕ_x , ϕ_y , and ϕ_z about the x-, y-, and z-axes respectively.
\mathbf{r}_1 and \mathbf{r}_2	Vectors representing the arm segments of a pan-tilt unit.

Superscript Notation

$_L$	Denotes that the quantity is expressed in a local coordinate system.
$_G$	Denotes that the quantity is expressed in the global coordinate system.

Model

θ	Model parameters.
ϕ_{pan} and ϕ_{tilt}	Pan and tilt angles for a gPTU.
ϕ	Pan and tilt values for all PTUs in a system.
$\hat{\phi}$	Pan and tilt values calculated from estimated $\hat{\theta}$.
$f(\phi \theta) = [\mathbf{p}_{\text{virtual}}^G, \mathbf{R}_{\text{virtual}}^G]$	Forward model.
$f^{-1}(\mathbf{p}_{\text{vir.}}^G, \mathbf{R}_{\text{vir.}}^G \theta) = \phi$	Inverse model.
$f^{-1}(\mathbf{p}_{\text{point}}^G \theta) = \phi$	Inverse model from measured point.

Parameter Estimation

$C(\boldsymbol{\theta} \boldsymbol{\phi})$	Cost function for parameters $\boldsymbol{\theta}$ and measured PTU angles $\boldsymbol{\phi}$.
$\varphi(\boldsymbol{\phi} \boldsymbol{\theta})$	Approximate angular error.
$d(\boldsymbol{\phi}_n \boldsymbol{\theta})$	Distance of closest crossing.
$L_{n,\text{avg}}(\boldsymbol{\phi}_n \boldsymbol{\theta})$	Mean distance to closest crossing.
$\boldsymbol{\theta}_{\text{true}}$	True model parameters for system.
$\boldsymbol{\theta}^*$	Model parameters minimizing cost function.
$\hat{\boldsymbol{\theta}}$	Estimated model parameters.

Contents

List of Abbreviations	v
Nomenclature	vi
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Background	1
1.2 Objective	2
1.3 Method	2
1.4 Limitations	3
1.5 Main Research Questions	3
2 Theory	4
2.1 Local Coordinate System	4
2.2 Mirror and Virtual Camera	4
2.3 Parameter Estimation	5
2.3.1 Least Squares Estimator	6
2.3.2 Optimization of Cost Function	7
2.3.3 Optimization Using Gradient Descent	7
3 Methods	9
3.1 Model of System	9
3.1.1 Pan-Tilt Unit Model	10
3.1.2 Forward Model	12
3.2 Inverse Model	12
3.2.1 Aligned Camera and Gimbal Mounted Mirror	14
3.2.2 Pan-Tilt Unit Mounted Mirror	16
3.3 Error Function for Two Sensor Rigs	18
3.3.1 Approximate Angular Error	19
3.4 Degrees of Freedom for Two Sensor Rigs	19
3.4.1 Information From Measurements of Points	21
3.5 Parameterization of System	21

3.6	Parameter Estimation	22
3.6.1	System Initialization	23
3.6.2	Parameter Estimation Using Least Squares	23
3.6.3	Cost Function Minimization Using Gradient Descent	23
3.6.4	Cost Function Minimization Using SciPy Optimizer	24
3.7	Aim Calibrated System at Point	24
3.8	Evaluation	25
3.8.1	Calibration and Test with Split Dataset	25
3.8.2	Use Case Testing in Simulated Environment	25
3.8.3	Use Case Testing with Real-World System	27
3.9	Effect of Calibration Point Quantity	28
3.10	Sensitivity to Initial Parameter Estimates	29
3.11	Setup for Real-World Testing	29
3.11.1	Measurement Accuracy for Real-World Testing	31
3.12	Simulated Calibration and Test Scenario	31
4	Results	34
4.1	Effect of Calibration Point Quantity	34
4.1.1	Effect of Calibration Points: Gradient Descent	34
4.1.2	Effect of Calibration Points: SciPy Optimizer	35
4.2	Use Case Testing in Simulated Environment	38
4.2.1	Simulated Use Case: Gradient Descent	38
4.2.2	Simulated Use Case: SciPy's Least Squares Optimizer	42
4.3	Use Case Testing with Real-World System	45
4.4	Sensitivity to Initial Parameter Estimates	48
4.4.1	Sensitivity for Gradient Descent Optimization	48
4.4.2	Sensitivity for SciPy's Least Squares Optimizer	50
5	Discussion	53
5.1	Effect of Calibration Point Quantity	53
5.2	Use Case Testing in Simulated Environment	54
5.2.1	Simulated Use Case: Gradient Descent	55
5.2.2	Simulated Use Case: SciPy's Least Squares Optimizer	55
5.3	Use Case Testing with Real-World System	56
5.4	Sensitivity to Initial Parameter Estimates	57
5.5	Future Work	58
5.5.1	Expand to Systems with More Sensor Rigs	58
5.5.2	Supplementary Measurement Data	58
5.5.3	Expand to Complete Camera Calibration	59
5.5.4	Parameterization	59
5.5.5	Evaluate Other Optimization Methods	60
6	Conclusion	61
	Bibliography	62

List of Figures

1.1	Illustrations of the sensor rig and the complete sensor system.	2
2.1	Illustration of global and local coordinate systems.	5
2.2	Simple illustration of a virtual camera.	6
3.1	Illustration of the three main components of a single sensor rig.	9
3.2	Schematic illustration of the forward model of the system.	10
3.3	Schematic for the PTU including the two rotation axes and radiuses.	10
3.4	Illustration of the aiming at a target point problem.	13
3.5	Schematic illustration of the inverse model.	14
3.6	Example setup with a gimbal mounted mirror.	14
3.7	Vectors related to $\hat{\mathbf{n}}_{\text{mirror}}$ required to aim a virtual camera at $\mathbf{p}_{\text{point}}^G$	16
3.8	Example setup with a PTU mounted mirror.	16
3.9	Example setup showing d for a misaligned measurement.	18
3.10	Illustration of DOF for two sensor rigs.	20
3.11	Schematic of parameter estimation and validation.	26
3.12	Schematic of testing of the predicted $\hat{\phi}_{\text{PTU}_1}$	27
3.13	Schematic of testing of predicted $\hat{\phi}_{\text{PTU}_1}$ for a real-world system.	28
3.14	Measured points and system parameters for the real-world system.	30
3.15	Illustration of the simulated calibration system.	32
3.16	Illustration of the simulated test system.	33
4.1	Dependence of mean angular error on number of calibration points: Gradient descent.	36
4.2	Dependence of mean angular error on number of calibration points: SciPy's least squares with small θ_{offset}	36
4.3	Dependence of mean angular error on number of calibration points: SciPy's least squares with larger θ_{offset}	37
4.4	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and $r_{\text{meas}} \in (1, 10 \text{ m})$ for gradient descent.	39
4.5	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and $r_{\text{meas}} \in (1, 100 \text{ m})$ for gradient descent.	40
4.6	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and tilt values when $r_{\text{meas}} = 10 \text{ m}$ for gradient descent.	40
4.7	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and tilt values when $r_{\text{meas}} = 100 \text{ m}$ for gradient descent.	41

4.8	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for calibration with gradient descent on points 100 ± 25 m away. . . .	41
4.9	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and $r_{\text{meas}} \in (1, 10$ m) for SciPy least squares.	42
4.10	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and $r_{\text{meas}} \in (1, 100$ m) for SciPy least squares.	43
4.11	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and tilts when $r_{\text{meas}} = 10$ m for SciPy least squares. .	43
4.12	Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} \theta)$ for different pan and tilts when $r_{\text{meas}} = 100$ m for SciPy least squares.	44
4.13	Illustration of estimated virtual cameras optical axes for $\hat{\phi}_{\text{PTU}_1}$ and ϕ_{PTU_1} for real-world verification measurements.	45
4.14	Comparison of the predicted $\hat{\phi}_{\text{PTU}_1}$ and real-world measured ϕ_{PTU_1} . .	46
4.15	Comparison of the predicted $\hat{\phi}_{\text{PTU}_1}$ and real-world measured ϕ_{PTU_1} for no calibration.	47
4.16	Mean angular error for different θ_0 : Gradient descent with small offsets.	49
4.17	Mean angular error for different θ_0 : Gradient descent with larger offsets.	50
4.18	Mean angular error for different θ_0 : SciPy least squares with small offsets.	51
4.19	Mean angular error for different θ_0 : SciPy least squares with larger offsets.	52

List of Tables

3.1	Measured parameters for the real-world system.	31
3.2	True parameters used in simulations.	32
3.3	Placement of points in the standard calibration and validation systems.	33
4.1	Set of small parameter offsets.	35
4.2	Set of larger parameter offsets.	35
4.3	Value ranges for evaluation with fixed r_{meas} and varying pan and tilt.	38
4.4	Value ranges for evaluation with fixed tilt and varying pan and r_{meas} .	38
4.5	Ranges for placement of points in the additional calibration systems.	38
4.6	Mean approximate angular errors for prediction in all tested cases: Gradient descent.	39
4.7	Mean approximate angular error for prediction in all tested cases: SciPy least squares.	42
4.8	Summary of estimated system parameters for the real-world system. .	47
4.9	Difference between estimated and measured system parameters for the real-world system.	47
4.10	First offset configuration for sensitivity evaluations.	48
4.11	Second offset configuration for sensitivity evaluations.	49
4.12	Percentages of mean angular error below 1° and 0.1° for the offset cases.	49
4.13	Third offset configuration for sensitivity evaluations.	51

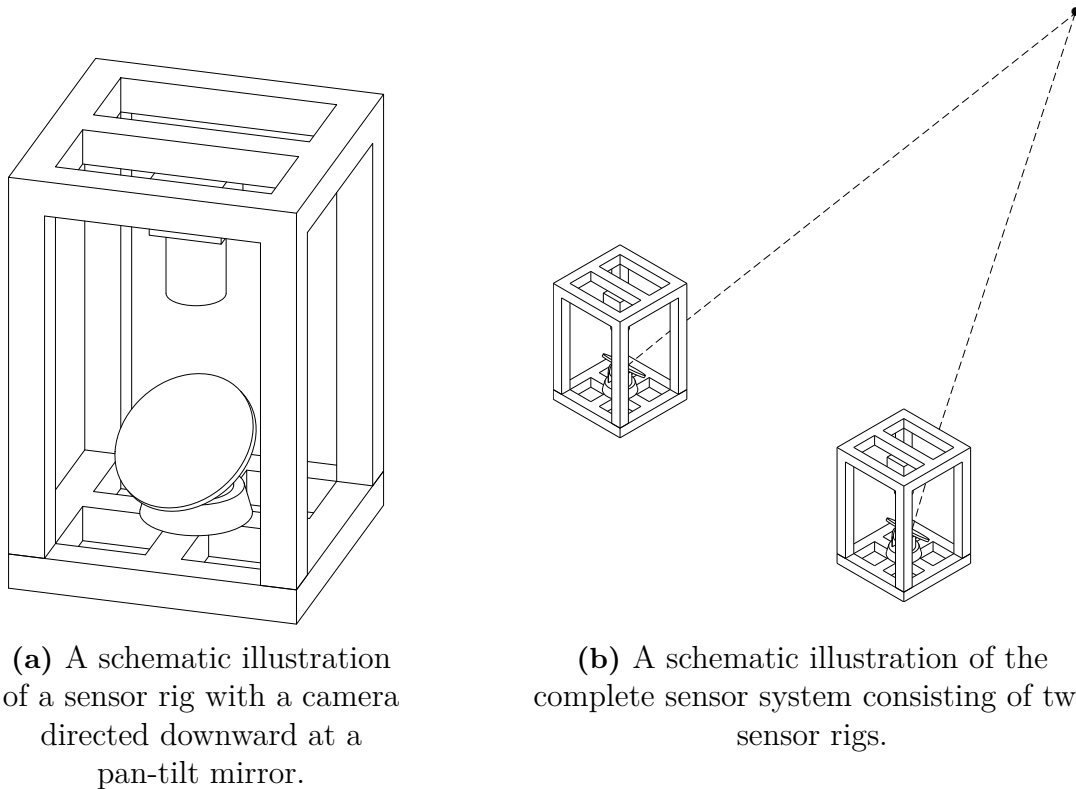
1

Introduction

Sensors are widely used in applications such as testing and evaluation to support data collection and enable informed decision-making. Depending on the measurement objectives, appropriate adaptations are required in both sensor type selection and how they are used. This thesis project focuses on a setup involving sensors used to measure points in 3D space.

1.1 Background

Central to this thesis project is a measurement setup where multiple sensors measure, by viewing, the same object in parallel. In the physical setup, each individual sensor is statically mounted on a rig and directed at a mirror below the sensor. The mirror is mounted on a pan-tilt unit (PTU), allowing for horizontal and vertical rotation of the mirror and thus enabling control of the sensor's field of view. An illustration of the physical setup is presented in Figure 1.1. To ensure the usefulness of the measurements, the sensor rigs must be properly calibrated so they accurately observe the same object. Currently, parts of the calibration are performed manually, which result in tracking errors and require considerable intervention by the user. Therefore, there is a need to replace this manual process with a more systematic method that is less time-consuming and improves measurement accuracy.



(a) A schematic illustration of a sensor rig with a camera directed downward at a pan-tilt mirror.

(b) A schematic illustration of the complete sensor system consisting of two sensor rigs.

Figure 1.1: Illustrations of the sensor rig and the complete sensor system.

1.2 Objective

The objective of this thesis is to investigate methods for calibrating and aligning sensors that observe the same object in 3D space. The goal of the calibration is to ensure that, after calibration, if one sensor observes a point in 3D space, the mirror orientations of the other sensor rigs are adjusted so they observe that same point. The calibration method should be capable of compensating for misalignments, both within individual rigs and between different rigs.

1.3 Method

The calibration requires a model of the system that describes where each rig component is positioned and aimed, based on the relative poses of the sensor and mirror, as well as their relation to the other rigs in the setup. The poses of the rig components constitute the system's parameters, which are numerous and difficult to manually align. Based on the model and a test sequence of measurements from the sensor rigs, the system's unknown parameters are estimated and optimized to accurately represent the physical setup. Once the parameters are known, misalignments can be compensated for and effective control of the system is possible.

1.4 Limitations

The long-term aim of this Master's thesis project is to implement a method that uses estimated parameters to adjust control of the system. However, the primary focus of this thesis will be on the most theoretically interesting aspects: modeling and parameter estimation. These aspects provide valuable insights, and the parameter estimation can be evaluated without integrating full control of the physical system.

While using a system with an arbitrary number of sensors may be useful, this project focuses on calibrating a setup consisting of two sensors. The proposed method may be applicable to a setup consisting of a larger number of sensors, but testing and evaluation of such cases are beyond the scope of this project.

The proposed method is designed to support various sensor types, provided it is possible to determine whether the sensor is directed at a given point. While multiple sensor types may meet this requirement, this thesis specifically focuses on ordinary color cameras. With cameras, measurements are limited to detecting whether the target point appears at the center of the camera image.

1.5 Main Research Questions

The main research questions are as follows:

- How can a calibration process be designed to calibrate and align an imaging system consisting of two sensors?
 - How can a model be created that accounts for the degrees of freedom of the system?
 - How can the extrinsic parameters for each sensor rig be estimated and used to ensure the system functions coherently?
 - How can misalignment of the setup components relative each other be compensated for?
- How large are the errors when observing points in 3D space after applying the calibration method?

2

Theory

This chapter presents theoretical background relevant to understanding the work in this Master's thesis. It includes explanations of local coordinate systems, the relationship between cameras, mirrors, and virtual cameras used in the project, as well as theory related to parameter estimation.

2.1 Local Coordinate System

When measuring positions relative to different objects, for example a camera, it can sometimes be useful to describe observed objects with the use of local coordinate systems. In 3D, these can be defined with a position for the local system's origin as well as the orientation of its coordinate axes, relative to the global frame. The position can be described by a vector with the following notation:

$$\mathbf{p}_{\text{origin}}^L = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^L = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}^G = \mathbf{p}_{\text{origin}}^G,$$

where superscript L or G refers to the coordinate in the local respectively global frame and $[p_x, p_y, p_z]$ is the coordinate of the origin in the global frame.

As for the orientation of the system, one way to represent it is with a rotation matrix. The rotation matrix for a local system is then the matrix that, multiplied with the global coordinate axes, leads to the local systems axes. In this report, a right hand coordinate system is used, where forward for an object (e.g. optical axis for a camera) is defined as positive x , positive y is left and positive z is up in its local coordinate system. An example of this is shown in Figure 2.1.

2.2 Mirror and Virtual Camera

In a system with a camera aimed at a planar mirror, the camera's view through the mirror can be equivalently described with the use of a virtual camera. This virtual camera is a camera with identical properties, but with another position and orientation such that its optical axis lines up with the reflected optical axis from the

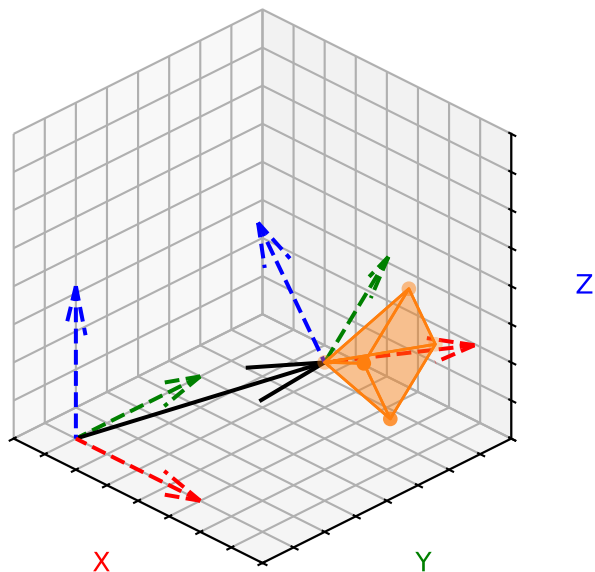


Figure 2.1: Axes of a camera's local coordinate system, translated and rotated relative to the global coordinate frame shown to the left. The vector for the camera's origin in the global coordinate system is indicated by a black arrow.

original camera. This is illustrated in Figure 2.2. Starting off with the orientation of the virtual camera, it can be calculated using the following equations:

$$\mathbf{R}_{\text{virtual}}^G = \mathbf{M}_{\text{reflection}} \mathbf{R}_{\text{camera}}^G, \text{ with } \mathbf{M}_{\text{reflection}} = \mathbf{I} - 2\hat{\mathbf{n}}_{\text{mirror}}\hat{\mathbf{n}}_{\text{mirror}}^T, \quad (2.1)$$

where $\mathbf{R}_{\text{camera}}^G$ and $\mathbf{R}_{\text{virtual}}^G$ are the rotation matrices representing the orientation of the camera and virtual camera, respectively, and $\hat{\mathbf{n}}_{\text{mirror}}$ is the mirror's normal vector [1]. The matrix $\mathbf{M}_{\text{reflection}}$ is a Householder transformation which is a general reflection about a plane.

Regarding the position $\mathbf{p}_{\text{virtual}}^G$ of the virtual camera, it can be calculated as follows:

$$\mathbf{p}_{\text{virtual}}^G = \mathbf{M}_{\text{reflection}}(\mathbf{p}_{\text{camera}}^G - \mathbf{p}_{\text{mirror}}^G) + \mathbf{p}_{\text{mirror}}^G, \quad (2.2)$$

where $\mathbf{p}_{\text{camera}}^G$ is the position of the camera and $\mathbf{p}_{\text{mirror}}^G$ is the position of an arbitrary point on the mirror surface [1]. This equation can be interpreted as reflecting the vector between a point on the mirror and the camera with $\mathbf{M}_{\text{reflection}}$ and then adding the mirror point position.

2.3 Parameter Estimation

Parameter estimation is a fundamental technique in statistical modeling and data analysis, which provides a way to estimate unknown parameters of a system. The

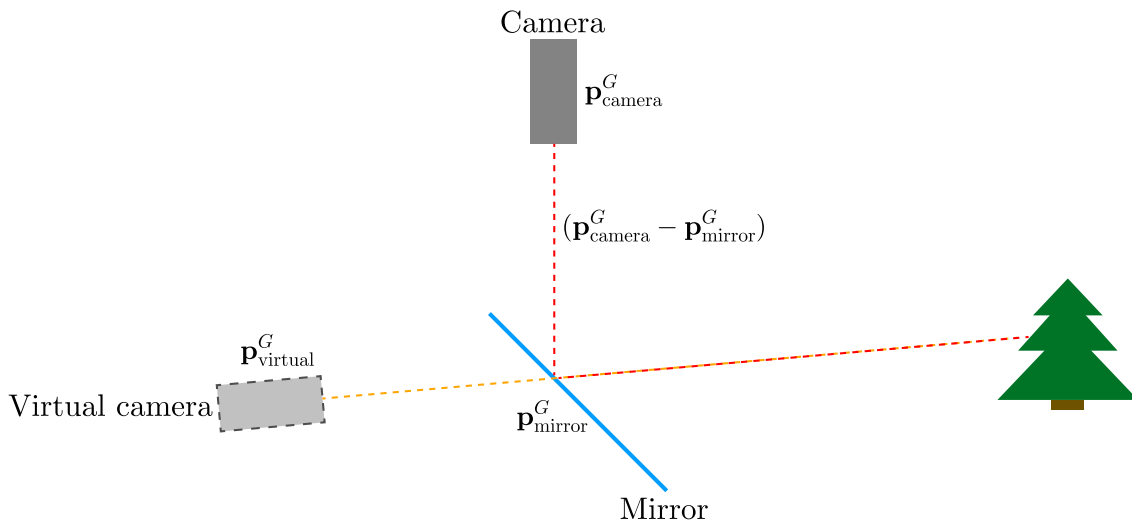


Figure 2.2: System with a camera aimed at a mirror and the corresponding virtual camera, which sees the same view as the original camera through the mirror. The positions and vectors used in Equation 2.2 are marked out.

first step in describing a real-world system is usually to create a mathematical model of the system containing the unknown parameters [2]. However, mathematical models of reality are often built on some approximations, which means that the model is not an exact representation of reality [3]. Parameter estimation methods aim to minimize the difference between model predictions and observed data, i.e. find the parameter values that makes the model best fit reality. The function describing this difference is called the cost function, or discrepancy function [4]. Which estimation method to apply to a certain problem depends on several factors, such as the model characteristics (e.g. linear or non-linear model), data properties (e.g. continuous or discrete data), and the goal of the analysis. Some estimation methods include maximum likelihood estimators, least squares, or minimum variance unbiased estimators [2], [4], [5].

2.3.1 Least Squares Estimator

The least squares estimator is a parameter estimation method that is widely used, partly because of its straightforward implementation [2]. The idea behind the least squares estimator is to find the parameter values that minimize the sum of the squared differences between the values predicted by the model and the observed data. This sum is known as the cost function, denoted $C(\boldsymbol{\theta})$, and is defined as:

$$C(\boldsymbol{\theta}) = \sum_{n=1}^N (y_n - d(x_n|\boldsymbol{\theta}))^2, \quad (2.3)$$

where $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \dots)$ represents the unknown parameters, N is the number of data points, y_n the observed data at the n th point, x_n the corresponding input values, and $d(x_n|\boldsymbol{\theta})$ is the model's prediction based on the current parameter and input values [2]. The goal is to find model parameters, $\boldsymbol{\theta}^*$, that minimizes the cost

function, as described below:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbb{R}}{\operatorname{argmin}} C(\boldsymbol{\theta}), \quad (2.4)$$

and thereby ensuring that the model fits the observed data as well as possible.

Least squares can be applied to both linear and non-linear models [2]. For the linear case, an analytical solution can be obtained directly, but for the non-linear case some optimization technique is typically required.

2.3.2 Optimization of Cost Function

As noted above, when the least squares method is applied to a non-linear model, an analytical solution is typically not available. Instead, the parameter values that minimize the cost function can be found using some numerical optimization technique. To solve the optimization problem described in Equation 2.4, several numerical methods may be applied that iteratively update the parameter estimates to minimize the cost function. Some examples include gradient descent, the Newton-Raphson method, and the Gauss-Newton method [6], [7].

One important consideration in optimization is the presence of local minima, which are points where the cost function is lower than the surrounding values, but not necessarily the lowest of all cost function values. The point across the entire parameter space where the cost function obtains its lowest value is called the global minimum, as described in Equation 2.4. The cost function may contain several local minima, making it challenging for optimization algorithms to converge into the global minimum [6], [7]. As a result, the choice of initial parameter estimates can significantly influence the result of the optimization, since the algorithm can get stuck in a local minimum.

2.3.3 Optimization Using Gradient Descent

Gradient descent is a commonly used iterative optimization method for minimizing differentiable cost functions. The algorithm operates by updating the parameter estimates in the direction opposite to the gradient of the cost function, thereby moving towards a minimum [3]. The gradient, denoted ∇C , is a vector of partial derivatives of the cost function with respect to each parameter [7]:

$$\nabla C = \left(\frac{\partial C}{\partial \theta_1}, \frac{\partial C}{\partial \theta_2}, \frac{\partial C}{\partial \theta_3}, \dots \right). \quad (2.5)$$

Each partial derivative describes the rate of change of the cost function with respect to a particular parameter, and can be approximated numerically as:

$$\frac{\partial C}{\partial \theta} \approx \frac{C(\theta_0 + h) - C(\theta_0)}{h}, \quad (2.6)$$

where h is a small perturbation. Once the gradient is computed, the parameters are updated according to the following rule:

$$\theta^{(i+1)} = \theta^{(i)} - \gamma \nabla C(\theta^{(i)}), \quad (2.7)$$

where γ denotes the step size, and i is the iteration number [3]. The choice of γ is important, since a too small step size can lead to slow convergence, while a too large step size can cause the algorithm to overshoot and miss the minimum [8].

3

Methods

This chapter describes the methods used to design and evaluate the calibration process for an imaging system consisting of two sensor rigs. It includes the creation of a model that describes the relationship between each rig’s main components: the camera, mirror, and pan-tilt unit. An illustration of a sensor rig is provided in Figure 3.1, which shows these components along with the virtual camera described in Section 2.2. The chapter also covers error functions, the system’s degrees of freedom, the parameterization approach, and the estimation of the system parameters. Finally, the evaluation procedures for various aspects of the calibration method and the real-world test setup are described.

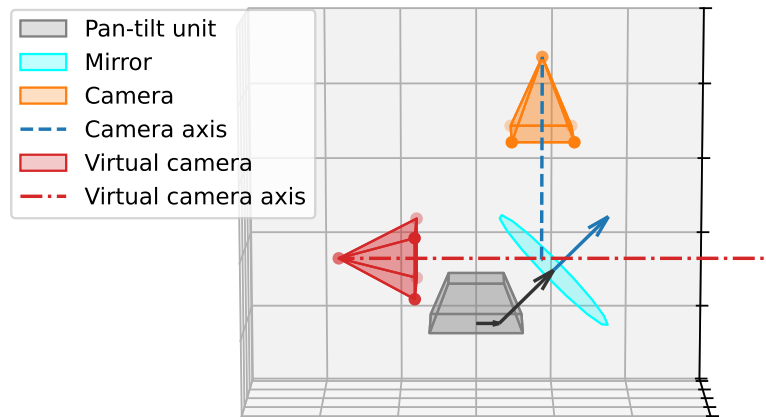


Figure 3.1: Illustration of the three main components of a single sensor rig: the pan-tilt unit (PTU), mirror and sensor. The sensor used in this thesis is a camera. The camera and mirror can be represented by a virtual camera, which is also illustrated in the figure.

3.1 Model of System

A model of the system is necessary for multiple parts of this project, for example as a foundation for the parameter estimation as well as enabling simulation and testing without the need to set up the physical system.

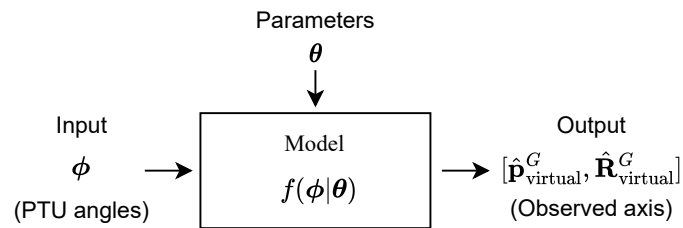


Figure 3.2: Schematic illustration of the forward model of the system.

The goal of the model is to describe where the view of the camera, or more specifically its optical axis, is aimed. This is dependent both on variables which are seen as static for a given system, such as positions and orientations of the camera and PTU base, as well as the pan-tilt angle set for the PTU, which is seen as an input since it can be varied for different measurements. For this system model, the varying pan-tilt angle will be referred to as the model input, the static system properties are called model parameters and the observed direction is the model output. The observed direction can be given by the virtual camera’s position and orientation, as explained in Section 2.2. Therefore, the goal is to describe the virtual camera. This model structure is shown in Figure 3.2.

When the positions and orientations of a camera and a mirror are known, a virtual camera can be calculated as described in Section 2.2. For this, the position and orientation of the mirror need to be determined. These are dependent on the PTU.

3.1.1 Pan-Tilt Unit Model

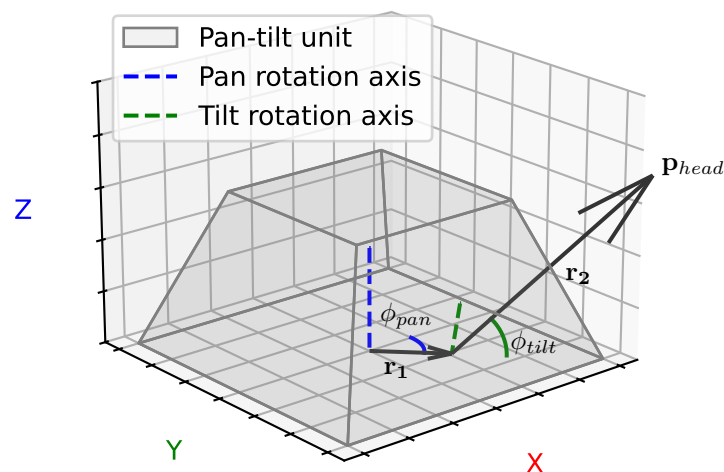


Figure 3.3: Schematic for the PTU including the two rotation axes and radiuses from them. \mathbf{r}_1 rotates around the PTU’s pan rotation axis, shown in blue, and leads to position in the xy -plane from which \mathbf{r}_2 is rotated in tilt.

The PTU consists of three notable parts: a stationary base, a rotatable arm and the PTU head at the end of the arm. It is on the head where objects, such as a

mirror in this project, are mounted. We therefore want to calculate the position and orientation of the head. This can be done similarly to how spherical coordinates are used but with slight modifications due to the geometry of the PTU.

The PTU can rotate around two different axes: pan (yaw) and tilt (pitch), but these are not aligned in a single point, which is illustrated in Figure 3.3. To model this, two separate radii \mathbf{r}_1 and \mathbf{r}_2 are used. The radius \mathbf{r}_1 originates from the PTU's origin and is rotated around the pan rotation axis by the value ϕ_{pan} . This gives a point in the xy -plane from which \mathbf{r}_2 extends in the same direction as \mathbf{r}_1 but rotated in tilt by ϕ_{tilt} . The sum of these vectors gives the position of the PTU head, whose orientation is given by the direction of \mathbf{r}_2 .

Working in the PTU's local coordinate system, the first vector can be calculated as:

$$\mathbf{r}_1 = \mathbf{R}_{xyz}(0, 0, \phi_{\text{pan}}) [r_1, 0, 0]^T, \quad (3.1)$$

where $\mathbf{R}_{xyz}(0, 0, \phi_{\text{pan}})$ is a rotation matrix for a ϕ_{pan} rotation around z , and r_1 is the length of the radius. Similarly, \mathbf{r}_2 can be calculated as:

$$\mathbf{r}_2 = \mathbf{R}_{xyz}(0, \phi_{\text{tilt}}, \phi_{\text{pan}}) [r_2, 0, 0]^T, \quad (3.2)$$

where $\mathbf{R}_{xyz}(0, \phi_{\text{tilt}}, \phi_{\text{pan}})$ includes an additional rotation in tilt. The sum of these two radius vectors give the position of the PTU head in the local system:

$$\mathbf{p}_{\text{head}} = \mathbf{r}_1 + \mathbf{r}_2 = \mathbf{R}_{xyz}(0, 0, \phi_{\text{pan}}) [r_1, 0, 0]^T + \mathbf{R}_{xyz}(0, \phi_{\text{tilt}}, \phi_{\text{pan}}) [r_2, 0, 0]^T, \quad (3.3)$$

and the orientation of the head can be described by the rotation matrix:

$$\mathbf{R}_{\text{head}} = \mathbf{R}_{xyz}(0, \phi_{\text{tilt}}, \phi_{\text{pan}}). \quad (3.4)$$

In the global system, these can be expressed with the help of the PTU's base position and orientation:

$$\mathbf{p}_{\text{head}}^G = \mathbf{R}_{\text{PTU}}^G \mathbf{p}_{\text{head}}^L + \mathbf{p}_{\text{PTU}}^G, \quad (3.5)$$

$$\mathbf{R}_{\text{head}}^G = \mathbf{R}_{\text{PTU}}^G \mathbf{R}_{\text{head}}^L. \quad (3.6)$$

3.1.2 Forward Model

As previously mentioned, the output of the model should be the position and orientation of the virtual camera, which describes where the sensor is aimed. In Section 2.2, equations for this are given as a function of camera and mirror position and orientation. In the setup used in this project, the camera is regarded as fixed and the mirror is directly mounted on the PTU head, which means that its pose is described by the equations in the previous Section (3.1.1). Combining these equations means that one sensor rig can be model with the following:

$$\mathbf{p}_{\text{virtual}}^G = \mathbf{M}_{\text{reflection}}(\mathbf{p}_{\text{camera}}^G - \underbrace{\mathbf{R}_{\text{PTU}}^G \mathbf{p}_{\text{head}}^L + \mathbf{p}_{\text{PTU}}^G}_{\mathbf{p}_{\text{head}}^G}) + \underbrace{\mathbf{R}_{\text{PTU}}^G \mathbf{p}_{\text{head}}^L + \mathbf{p}_{\text{PTU}}^G}_{\mathbf{p}_{\text{head}}^G}, \quad (3.7)$$

$$\mathbf{R}_{\text{virtual}}^G = \mathbf{M}_{\text{reflection}} \mathbf{R}_{\text{camera}}^G, \quad (3.8)$$

where $\mathbf{M}_{\text{reflection}}$, $\mathbf{R}_{\text{head}}^L$ and $\mathbf{p}_{\text{head}}^L$ are:

$$\mathbf{M}_{\text{reflection}} = I - 2 \underbrace{(\mathbf{R}_{\text{PTU}}^G \mathbf{R}_{\text{head}}^L \hat{x})}_{\mathbf{n}_{\text{mirror}}} \underbrace{(\mathbf{R}_{\text{PTU}}^G \mathbf{R}_{\text{head}}^L \hat{x})^T}_{\mathbf{n}_{\text{mirror}}^T}, \quad (3.9)$$

$$\mathbf{R}_{\text{head}}^L = \mathbf{R}_{xyz}(0, \phi_{\text{tilt}}, \phi_{\text{pan}}), \quad (3.10)$$

$$\mathbf{p}_{\text{head}}^L = \mathbf{R}_{xyz}(0, 0, \phi_{\text{pan}}) r_1 \hat{x} + \mathbf{R}_{xyz}(0, \phi_{\text{tilt}}, \phi_{\text{pan}}) r_2 \hat{x}. \quad (3.11)$$

In these equations, the pose of the camera ($\mathbf{p}_{\text{camera}}^G$, $\mathbf{R}_{\text{camera}}^G$) and the PTU base ($\mathbf{p}_{\text{PTU}}^G$, $\mathbf{R}_{\text{PTU}}^G$) are seen as model parameters which are fixed for a given system, and ϕ_{pan} and ϕ_{tilt} are inputs to give us the model outputs $\mathbf{R}_{\text{virtual}}^G$ and $\mathbf{p}_{\text{virtual}}^G$. This model will be referenced to as f in this report:

$$f(\phi_{\text{pan}}, \phi_{\text{tilt}} \mid \underbrace{\mathbf{p}_{\text{camera}}^G, \mathbf{R}_{\text{camera}}^G, \mathbf{p}_{\text{PTU}}^G, \mathbf{R}_{\text{PTU}}^G}_{\theta}) = [\mathbf{p}_{\text{virtual}}^G, \mathbf{R}_{\text{virtual}}^G]. \quad (3.12)$$

3.2 Inverse Model

In Section 3.1.1, a forward model from PTU mirror angles to virtual camera pose was described. The forward model is, for example, used during the parameter estimation, but some cases requires solving the inverse problem instead. This is when the input to our forward model, the PTU mirror angles, is unknown and is instead the wanted output. Examples of this are cases such as generating measurements for a simulated scenario, or aiming the virtual camera at a known 3D point.

If an inverse model were to be created directly from the forward model in Equation 3.12, the following function would be obtained:

$$f_{\text{direct}}^{-1}(\mathbf{p}_{\text{virtual}}^G, \mathbf{R}_{\text{virtual}}^G | \theta) = [\phi_{\text{pan}}, \phi_{\text{tilt}}]. \quad (3.13)$$

Although this is the inverse of the forward model, it comes with problems that will motivate slight modifications. One challenge is that this inverse is only defined for the restricted set of combinations of $\mathbf{p}_{\text{virtual}}^G$ and $\mathbf{R}_{\text{virtual}}^G$ that is in the range of f . In some situations, finding a pose that corresponds to the range of f can be challenging.

In this project, solving the inverse problem is usually related to aiming the virtual camera at a known 3D point $\mathbf{p}_{\text{point}}$, shown in Figure 3.4. To aim the virtual camera at $\mathbf{p}_{\text{point}}$, we select a pose such that:

$$\alpha \underbrace{\mathbf{R}_{\text{virtual}}^G \hat{x}}_{\text{optical axis}} + \mathbf{p}_{\text{virtual}}^G = \mathbf{p}_{\text{point}}^G \quad (3.14)$$

is fulfilled for a (positive) scaling factor α , since there is a whole line of points that correspond to a single pose. For this thesis, it is this combined problem of finding $[\phi_{\text{pan}}, \phi_{\text{tilt}}]$ from a point $\mathbf{p}_{\text{point}}^G$ which needs to be solved. This will first be done for a simplified case with a gimbal mounted mirror and a camera whose optical axis is aligned with the mirror's rotation point. For this case, a closed form solution is possible. After this, an iterative method for a more general camera mirror setup will be presented. The found solution will be referred to as $f^{-1}(\mathbf{p}_{\text{point}}^G | \theta)$. A schematic illustration of f^{-1} is shown in Figure 3.5.

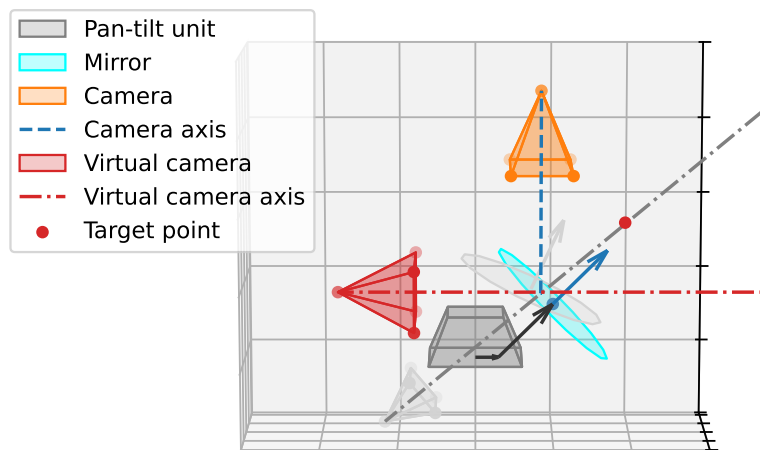


Figure 3.4: Example setup with a target point marked in red. The system starts aimed straight forward and the inverse problem is to calculate which PTU mirror angles result in aiming the virtual camera at the point. The solution is shown in grey.

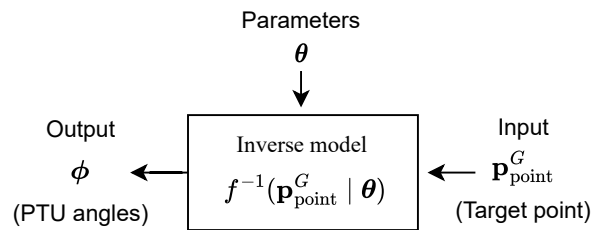


Figure 3.5: Schematic illustration of the inverse model for calculating the PTU mirror angles required to look at a target point $\mathbf{p}_{\text{point}}^G$.

3.2.1 Aligned Camera and Gimbal Mounted Mirror

In this simplified system, the mirror is gimbal mounted, which means that the mirror rotates around a single point that is placed in the center of the mirror's surface. To further simplify, the camera's optical axis is assumed to go through this point. A reflected ray will always go through the same point on the mirror surface as the incident ray. Since the optical axis goes through the mirror's rotation point ($\mathbf{p}_{\text{mirror}}$) which is constant, this will always be on the virtual camera's optical axis. This is shown in Figure 3.6.

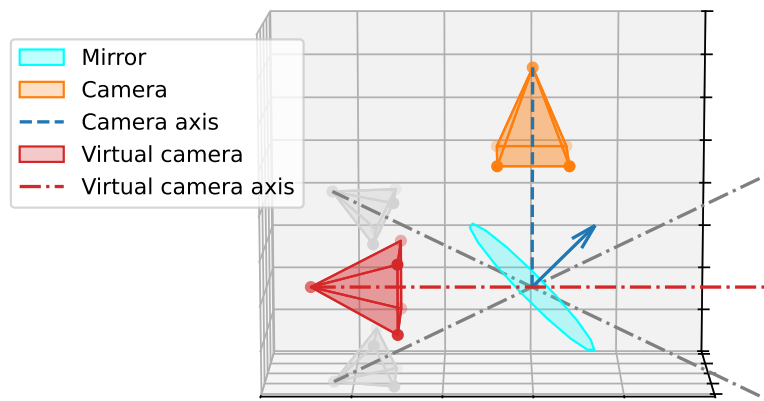


Figure 3.6: Example setup with a gimbal mounted mirror and a camera whose optical axis is aimed at the mirror's rotation point. With this setup the optical axis of the virtual camera will also always go through the mirror's rotation point. Shown in grey are virtual cameras for other mirror tilts.

The first step in the combined problem of aiming at a point is to know what virtual camera pose fulfills Equation 3.14. Due to our constraint that the optical axis passes through $\mathbf{p}_{\text{mirror}}$ we can reformulate the problem as:

$$\alpha' \underbrace{\mathbf{R}_{\text{virtual}}^G}_{\text{optical axis}} \hat{x} + \mathbf{p}_{\text{mirror}}^G = \mathbf{p}_{\text{point}}^G, \quad (3.15)$$

where $\mathbf{p}_{\text{mirror}}^G$ is the mirror's rotation point and α' is still a scaling factor (but which

value will differ from Equation 3.14). In this equation, if $\mathbf{p}_{\text{mirror}}^G$ and $\mathbf{p}_{\text{point}}^G$ are known then $\mathbf{R}_{\text{virtual}}^G \hat{x}$ can be solved for as:

$$\mathbf{R}_{\text{virtual}}^G \hat{x} \propto \mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{mirror}}^G, \quad (3.16)$$

which simply is that the optical axis of the virtual camera should be parallel to the vector between the mirror's rotation point and the target point. From the forward model (Equation 3.8) we have:

$$\begin{aligned} \mathbf{M}_{\text{reflection}} \mathbf{R}_{\text{camera}}^G \hat{x} &\propto \mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{mirror}}^G \\ \Leftrightarrow (\mathbf{I} - 2\hat{\mathbf{n}}_{\text{mirror}} \hat{\mathbf{n}}_{\text{mirror}}^T) \mathbf{R}_{\text{camera}}^G \hat{x} &\propto \mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{mirror}}^G. \end{aligned} \quad (3.17)$$

In this equation the only unknown is $\hat{\mathbf{n}}$ which can be calculated from the plane of reflection. Relations that help with this is that the sum of the normalized incident and reflected vector will be located in the reflection plane, and that the reflection plane will be normal to the plane spanned by these vectors. In our case, the incident vector is $\mathbf{R}_{\text{camera}}^G \hat{x}$ and reflected is parallel to $\mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{mirror}}^G$ which means that $\hat{\mathbf{n}}_{\text{mirror}}$ can be found from:

$$\hat{\mathbf{n}}_{\text{mirror}} \propto (\mathbf{R}_{\text{camera}}^G \hat{x} + (\mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{mirror}}^G)) \times (\mathbf{R}_{\text{camera}}^G \hat{x} \times (\mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{mirror}}^G)). \quad (3.18)$$

Vectors related to this problem are illustrated in Figure 3.7. With the normal vector for the mirror known, the mirror angles ϕ_{pan} and ϕ_{tilt} can be calculated which solves the inverse problem in this simplified case.

The presented method was the one found and used during the project to find $\hat{\mathbf{n}}_{\text{mirror}}$, but close to the end we were made aware of alternate solutions, which can be more straight forward.

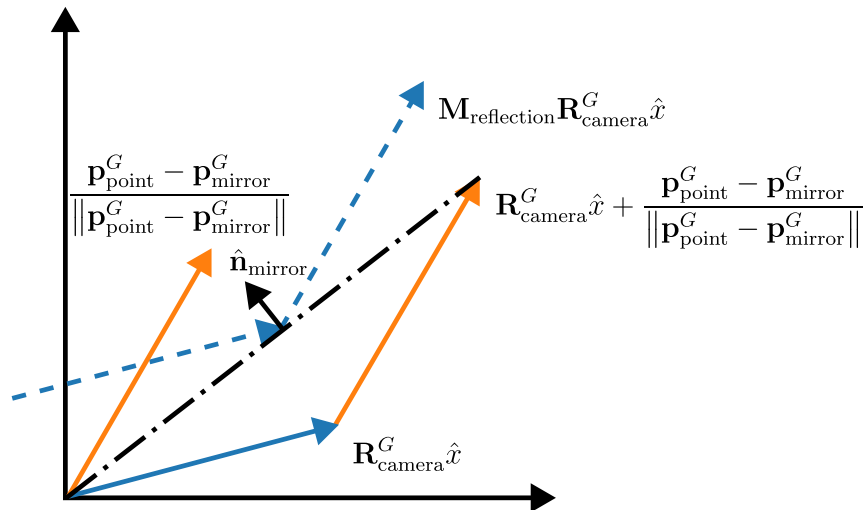


Figure 3.7: Vectors related to the calculation of $\hat{\mathbf{n}}_{\text{mirror}}$ required to aim a virtual camera towards a point $\mathbf{p}_{\text{point}}^G$. The drawing is in the plane spanned by $\mathbf{R}_{\text{camera}}^G \hat{x}$ and $\mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{mirror}}^G$.

3.2.2 Pan-Tilt Unit Mounted Mirror

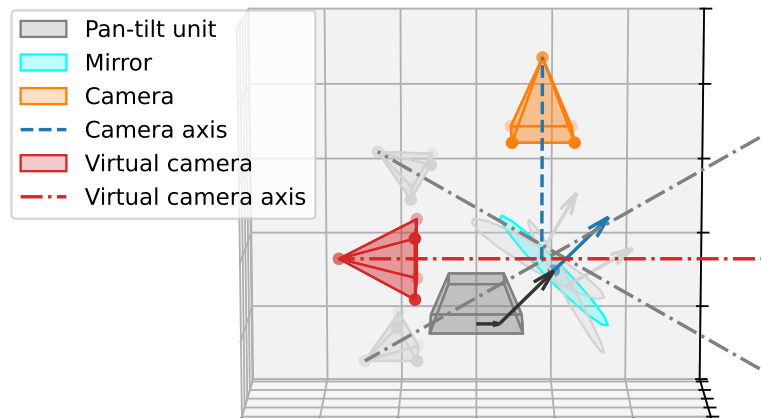


Figure 3.8: Example setup with a PTU mounted mirror. Unlike the simplified case in Figure 3.6 the virtual cameras optical axis no longer passes through a single constant point. This can be seen from the grey virtual cameras for other PTU tilts.

In the physical system, the mirror is not gimbal mounted but instead rotates around a point outside the mirror, as was explained in Section 3.1.1. This means that the constraint of the optical axis passing through a single constant point is no longer true (this is also the case if the camera would be misaligned), something which can be seen in Figure 3.8. This is problematic since this was used in Equation 3.16 in

order to calculate the wanted $\mathbf{R}_{\text{virtual}}^G$. Now we instead have to use $\mathbf{p}_{\text{virtual}}^G$ as in Equation 3.14 which is unknown, in addition to $\mathbf{R}_{\text{virtual}}^G$ that we were trying to solve for. Including more equations from the model to try and find a closed form solution for it has proven troublesome, but an alternative is to find an iterative method.

For the iterative method, we make use of that Equation 3.18 can still give us the mirror normal vector, given a wanted direction for the virtual camera, if rewritten as:

$$\hat{\mathbf{n}}_{\text{mirror}} \propto (\mathbf{R}_{\text{camera}}^G \hat{x} + (\mathbf{v}_{\text{dir}})) \times (\mathbf{R}_{\text{camera}}^G \hat{x} \times (\mathbf{v}_{\text{dir}})), \quad (3.19)$$

where \mathbf{v}_{dir} is the wanted direction for the virtual camera. The correct value is $\mathbf{v}_{\text{dir}} \propto \mathbf{p}_{\text{point}}^G - \mathbf{p}_{\text{virtual}}^G$, but since $\mathbf{p}_{\text{virtual}}^G$ is unknown we approximate $\tilde{\mathbf{v}}_{\text{dir}} \propto \mathbf{p}_{\text{point}}^G - \tilde{\mathbf{p}}_{\text{virtual}}^G$ with an approximate position $\tilde{\mathbf{p}}_{\text{virtual}}^G$, which can be chosen as the starting/current calculated virtual camera position in an iterative method. From $\tilde{\mathbf{v}}_{\text{dir}}$ we calculate $\tilde{\hat{\mathbf{n}}}_{\text{mirror}}$ using Equation 3.19. From this, $\mathbf{M}_{\text{reflection}}(\tilde{\hat{\mathbf{n}}}_{\text{mirror}})$ can be calculated and then the pose for the virtual camera with the forward model, which is the starting point for the next iteration. The equations, where superscript $_{\text{ }}^{(k)}$ refers to the variable value for iteration k , for this procedure are as follows:

$$\begin{aligned} \tilde{\mathbf{v}}_{\text{dir}}^{(k)} &= \frac{\mathbf{p}_{\text{point}}^G - \tilde{\mathbf{p}}_{\text{virtual}}^{G,(k-1)}}{\|\mathbf{p}_{\text{point}}^G - \tilde{\mathbf{p}}_{\text{virtual}}^{G,(k-1)}\|}, \\ \tilde{\hat{\mathbf{n}}}_{\text{mirror}}^{(k)} &= (\mathbf{R}_{\text{camera}}^G \hat{x} + \tilde{\mathbf{v}}_{\text{dir}}^{(k)}) \times (\mathbf{R}_{\text{camera}}^G \hat{x} \times \tilde{\mathbf{v}}_{\text{dir}}^{(k)}), \\ \tilde{\hat{\mathbf{n}}}_{\text{mirror}}^{(k)} &\implies [\tilde{\phi}_{\text{pan}}^{(k)}, \tilde{\phi}_{\text{tilt}}^{(k)}], \\ [\phi_{\text{pan}}^{(k)}, \phi_{\text{tilt}}^{(k)}] &= (1 - \beta)[\tilde{\phi}_{\text{pan}}^{(k)}, \tilde{\phi}_{\text{tilt}}^{(k)}] + \beta[\phi_{\text{pan}}^{(k-1)}, \phi_{\text{tilt}}^{(k-1)}], \\ f(\phi_{\text{pan}}^{(k)}, \phi_{\text{tilt}}^{(k)} \mid \theta) &= [\tilde{\mathbf{p}}_{\text{virtual}}^{G,(k)}, \tilde{\mathbf{R}}_{\text{virtual}}^{G,(k)}], \end{aligned} \quad (3.20)$$

where β is a scalar to adjust step size to keep the method stable. This is calculated for multiple iterations until the difference $[\phi_{\text{pan}}^{(k)}, \phi_{\text{tilt}}^{(k)}] - [\phi_{\text{pan}}^{(k-1)}, \phi_{\text{tilt}}^{(k-1)}]$ is less than a required tolerance.

3.3 Error Function for Two Sensor Rigs

To enable parameter estimation, an error function for given parameters and measurements needs to be defined. In this project, the parameters correspond to two sensor rigs at once. This can be modeled with the model presented in Section 3.1.2 by using θ containing different parameters for the two rigs which gives us the pose for both virtual cameras.

When measuring a point with both sensors, one constraint is that the optical axes for the virtual cameras cross at the measured point. During calibration, the position of the measured point is unknown, but the knowledge that the axes should cross at some point is useful.

One possible error function is to calculate the minimum distance between the optical axes $d(\phi|\hat{\theta})$, where ϕ contains both sensor rigs pan and tilt values for one measurement and $\hat{\theta}$ the rigs parameter values. This choice of error function is advantageous since the true value is known to be zero (assuming ideal measurements). For incorrect values of $\hat{\theta}$, there is generally no true crossing between the axes and the goal is instead to calculate the minimum distance between the axes, the location which will be referred to as the “closest crossing”. This is illustrated in Figure 3.9.

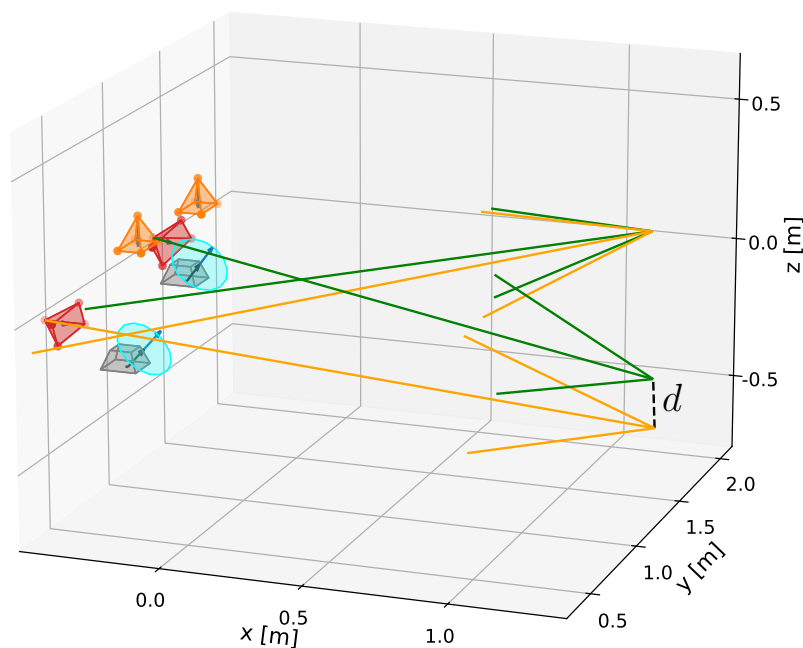


Figure 3.9: Example setup showing one measurement where the optical axes of the virtual cameras intersect (the two upper vectors), and another where the minimal distance between the two optical axes is d (lower vectors). The distance d is referred to as “closest crossing”.

To calculate $d(\phi|\hat{\theta})$, we use several properties, beginning with the requirement that the shortest distance is perpendicular to both virtual cameras' optical axes:

$$\hat{\mathbf{v}}_3 = \frac{\mathbf{v}_1 \times \mathbf{v}_2}{|\mathbf{v}_1 \times \mathbf{v}_2|},$$

where \mathbf{v}_1 and \mathbf{v}_2 are the direction vectors of the optical axes, and $\hat{\mathbf{v}}_3$ is the unit vector perpendicular to both. We want to find the length d for $\hat{\mathbf{v}}_3$ where $d \cdot \hat{\mathbf{v}}_3$ is equal to the difference between two points on the lines defined by the optical axes. These points can be found by scaling the virtual cameras' direction vectors and adding it to the virtual cameras' positions \mathbf{p}_1 and \mathbf{p}_2 . Finding these three scalars can, among other methods, be done by solving an equation system on the form $A\mathbf{x} = \mathbf{b}$ where:

$$A = \begin{bmatrix} \mathbf{v}_1^T \\ -\mathbf{v}_2^T \\ \hat{\mathbf{v}}_3^T \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \mathbf{p}_1 - \mathbf{p}_2.$$

The solution to this system is:

$$\mathbf{x} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ d \end{bmatrix} = A^{-1}\mathbf{b},$$

where α_i are the scalings for the the virtual cameras' direction vectors, and $|d|$ is minimum the distance for the closest crossing $d(\phi|\theta)$ we wanted. This distance is used as the error in the least squares cost function to be optimized in the parameter estimation. The result can also be used in order to calculate $L_{\text{avg}} = (\alpha_1 + \alpha_2)/2$, which is used to normalize the error for different measuring distances.

3.3.1 Approximate Angular Error

The normalized error $\frac{d(\phi_n|\theta)}{L_{n,\text{avg}}(\phi_n|\theta)}$ works well to get a numerical value for the cost function, but can be hard to interpret. One alternative is to try to describe this as an angular error. From each sensor, the optical axis and vector \mathbf{v}_3 form the adjacent and opposite sides of a right angle triangle where the angle is $\arctan\left(\frac{d}{\alpha_i}\right)$. To get a single value per measurement the mean distance to the closest crossing L_{avg} is used as the length of the adjacent side which gives the approximate angle error $\varphi(\phi_n|\theta) = \arctan\left(\frac{d(\phi_n|\theta)}{L_{n,\text{avg}}(\phi_n|\theta)}\right)$. This error is used in the presentation of the results but not in the parameter estimation method.

3.4 Degrees of Freedom for Two Sensor Rigs

The degrees of freedom (DOF), which is the number of free parameters in the system, is a fundamental property that can impact how difficult it can be to estimate the

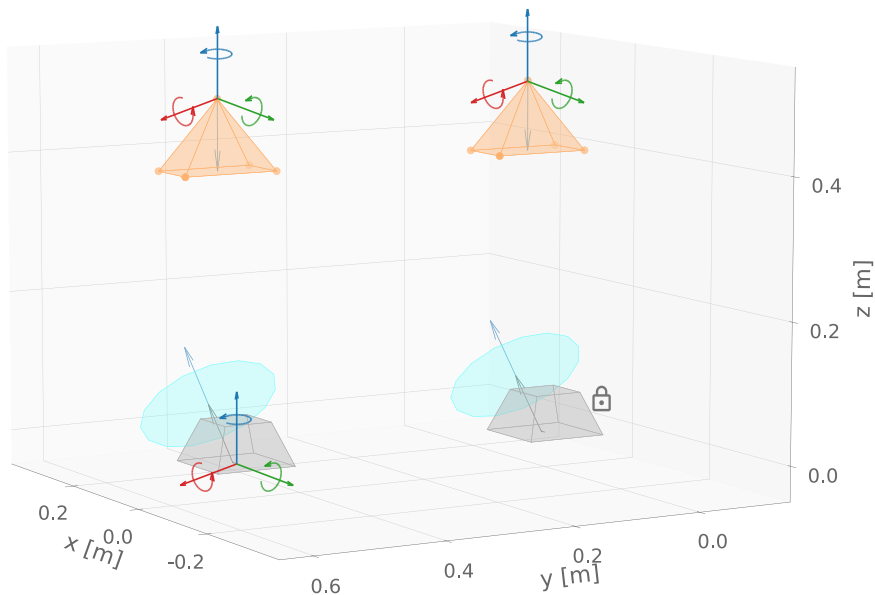


Figure 3.10: Example setup with two sensor rigs and the corresponding degrees of freedom where one PTU is chosen to be placed in the origin. All six DOF for each camera are marked out, although two of these do not impact the measurement of points. DOF from the PTU radiuses are not shown.

parameters θ for a system. In this project, the DOF is dependent on the model $f(\phi|\theta)$ and the number of sensor rigs used. For the case of two sensor rigs, we have two cameras and two PTUs, which is illustrated in Figure 3.10.

For a general object with a direction that is free to move in 3D, there exists six DOF: three translations and three rotations. This is the case for the cameras in the system, but even if the camera pose has these DOF they do not all necessarily need to impact the output.

For measurements of points, it is the optical axis of the virtual camera that impacts the result. The direction of the camera's optical axis $\mathbf{R}_{\text{camera}}^G \hat{x}$ is a vector that only has two DOF, even if $\mathbf{R}_{\text{camera}}^G$ has three DOF, since rotations around the optical axis do not have any impact on its direction. In addition to this, camera positions along the optical axis also result in the same optical axis for the virtual camera. This means that only four DOF for the camera affect measurements of points.

As for the PTUs, all six pose DOF impact the model output. This, in combination with the two arm radiuses in the PTU, lead to eight DOF for one PTU.

All objects are positioned in a global coordinate system. The origin and orientation of this coordinate system can be chosen arbitrarily, but an advantageous choice is to place one of the PTUs in the origin, aligned with the systems coordinate axes. By doing that, the six pose DOF are removed, as illustrated in Figure 3.10. Combining these modifications with the four DOF for each camera, the system's total degrees

of freedom that affect measurements, based on this high-level analysis, is:

$$\underbrace{2 \cdot 4}_{\text{Cameras}} + \underbrace{6 + 2}_{\text{PTU}} + \underbrace{0 + 2}_{\text{Origin PTU}} = 18 \text{ DOF.}$$

If prior knowledge about some parameters is available, for example exact measurements of the radiuses for the PTUs, these parameters can be locked. This reduces the number of DOF which can lead to faster calculations and can improve the result. Tests in this project have been carried out without locking any parameters, except that one PTU is placed at the origin.

3.4.1 Information From Measurements of Points

A requirement for the parameter estimation to work is that the input measurements contain enough information to reduce the possible solutions to a useful range. One simplified way to analyze this is to view the problem as a large equation system, where every DOF is an unknown variable and additional information has to be provided so that the system is solvable. When introducing a measured point to the system it gives us four measured values: $\phi_{\text{pan}}, \phi_{\text{tilt}}$ for each sensor rig. These angles are known to aim the system at a single point, although the position of this point is unknown. This means that a measured point introduces four known values (measured angles) simultaneously as three unknowns (point position), resulting in a net increase of one known value. If multiple independent points are measured this simplified analysis would predict that $n_{\text{points}} = n_{\text{DOF}}$ is needed to determine the values corresponding to the systems DOF, but it has to be stressed that this analysis does not necessarily have to be correct. A closer analysis of the model and equations introduced from measurements could result in another quantity.

3.5 Parameterization of System

To enable parameter estimation of the system, the parameters θ have to be expressed, something which can be done in multiple ways. The critical information θ carries is the positions and orientations of the objects and the PTU radiuses, which all need to be parameterized. The choice of parameterization can lead to different levels of complexity, but also advantages during parameter estimation. The choices in this thesis have prioritized simplicity due to time constraints, though this comes with potential disadvantages.

Beginning with the PTU radiuses r_1 and r_2 , these are simply parametrized as the length in millimeters, separately for each PTU in the system.

The positions of the objects are described using Cartesian coordinates in the global coordinate system, also in millimeters. Other alternatives exist, such as describing the position of a sensor rig in the global system and the position of the corresponding camera relative to this, something which is further discussed in Section 5.5.4.

As for the orientations, three-by-three rotation matrices containing a total of nine values are used inside the model. To avoid needing nine parameters for the three DOF that exist for a rotation matrix, different parameterizations can be used. Two common alternatives are Euler angles and quaternions.

Euler angles construct a rotation matrix from three separate rotations around the coordinate system axes. This only requires three parameters, but has a disadvantage of gimbal lock, which means that two orientations that are close on the unit sphere can have drastically different parameter values [9].

Quaternions, in comparison, require four parameters but do not exhibit gimbal lock, which can be advantageous for common optimization methods and therefore used in this project [9].

To enable the use of quaternions for parametrization together with the model in Section 3.1, quaternions are used for parameter estimation and converted to rotation matrices where required inside the system model. In code, this is implemented with the `scipy.spatial.transform.Rotation` class and corresponding methods [10].

The chosen parameterization results in a total of:

$$\underbrace{2 \cdot (3 + 4)}_{\text{Cameras}} + \underbrace{(3 + 4 + 2)}_{\text{PTU}} + \underbrace{2}_{\text{Origin PTU}} = 25 \text{ free parameters,}$$

for a two-rig system. This is seven more than the number of DOF presented in Section 3.4, meaning that the system is over-parameterized. A different parameterization could address this over-parameterization, but has not been implemented in this thesis. Possible improvements and consequences are further discussed in Section 5.5.4.

Although quaternions are chosen over Euler angles for the parameter estimation method in this thesis, they can sometimes be hard to interpret. As a consequence of this, orientation of objects in this report is often specified with Euler angles instead. This is done for extrinsic rotations with $x - y - z$ order (also referred to as roll, pitch and yaw (RPY)) [10].

3.6 Parameter Estimation

The parameter estimation is performed using the least squares estimator, where the cost function described in Equation 2.3 is minimized. The minimization is conducted using two different optimization methods, which are evaluated separately: a manually implemented gradient descent algorithm, and SciPy's built in least squares optimizer, `scipy.optimize.least_squares` [11].

3.6.1 System Initialization

To successfully estimate the unknown parameters of the system, it must first be clear what the known input data consists of. As mentioned in Section 3.1, what is known from measurements is the pan and tilt of each PTU in the setup such that each sensor is aimed at a common point in 3D space. These values are denoted by ϕ_n for measurement of the n th point.

When calibrating a real-world setup, the initial guess for the parameter values consist of roughly measured values of the physical system. The accuracy of these measurements are within a few centimeters for the positions, or a few degrees for the rotations. When calibrating a simulated system, offsets to the initial parameter estimates, θ_0 , are defined before the parameter estimation process starts.

3.6.2 Parameter Estimation Using Least Squares

The pose, i.e. position and direction, of the virtual camera must be known in order to determine where the optical axis of the camera is directed. Calculation of the current pose is performed as described in previous sections, and is based on the current parameter guess (initially the parameter starting guess) and measured angles, ϕ_n . This is performed for all N calibration points. When this is complete, it is possible to evaluate the cost function that is used for the parameter estimation.

The least squares estimator is used for parameter estimation, with a cost function defined in Equation 2.3. The model's prediction, which in our case corresponds to $d(\phi_n|\theta)$, represents the estimated closest distance between two direction vectors of the virtual cameras, based on the current input and parameter values. As described in Section 3.3, the direction vector of each virtual camera is scaled to the length where the vectors are the closest to each other, and then the distance between them is calculated to obtain $d(\phi_n|\theta)$. The true data, y_n , in Equation 2.3 is zero for all n , since the two direction vectors in the real system should cross each other at the observed point in 3D space. For each point, $d(\phi_n|\theta)$ is normalized with respect to the average length, $L_{n,\text{avg}}(\phi_n|\theta)$, of the two direction vectors. The cost for all N points is summed and normalized with respect to the total number of points. The normalized version of the cost function described in Equation 2.3 is described by:

$$C(\theta|\phi) = \frac{1}{N} \sum_{n=1}^N \left(\frac{d(\phi_n|\theta)}{L_{n,\text{avg}}(\phi_n|\theta)} \right)^2, \quad (3.21)$$

where ϕ contains the measured mirror angles for all N calibration points. The parameters that minimize this function are sought.

3.6.3 Cost Function Minimization Using Gradient Descent

With gradient descent, the goal is to find what parameter changes make the cost decrease the most, and then take a step in that direction. To get this direction, the gradient of the cost function is acquired by calculating the derivatives of the cost

function with respect to each parameter separately:

$$\nabla C = \left(\frac{\partial C}{\partial \theta_1}, \frac{\partial C}{\partial \theta_2}, \dots, \frac{\partial C}{\partial \theta_p} \right), \quad (3.22)$$

where p is the number of unknown parameters. When modifying a parameter representing a rotation, which is expressed using quaternions, the quaternion is re-normalized afterward to ensure it remains a unit quaternion.

The step is then calculated according to the update rule mentioned in Section 2.3.3:

$$\hat{\boldsymbol{\theta}}^{(i+1)} = \hat{\boldsymbol{\theta}}^{(i)} - \gamma \nabla C(\hat{\boldsymbol{\theta}}^{(i)} | \boldsymbol{\phi}). \quad (3.23)$$

The step size, γ , is chosen to be parameter dependent, with one step size for the position parameters, and another step size for the orientation parameters. During calibration using gradient descent, a step size of 25,000 is used for updating position parameters, while a step size of 0.05 is used for orientation parameters. When the parameter values have been updated, the process is performed again to lower the cost further and update the parameters again. The optimization of the cost function is carried out until some threshold value is reached or when a specific number of iterations have been performed.

3.6.4 Cost Function Minimization Using SciPy Optimizer

In addition to the manually implemented gradient descent algorithm, SciPy's built-in least squares optimizer, `scipy.optimize.least_squares`, is used for minimizing the cost function. The performance of the two optimization methods can be compared in terms accuracy, to evaluate which method is better suited for calibrating the system under consideration in this project.

When using this optimizer, the default settings are primarily used. Full documentation for the optimizer is available in [11]. However, bounds are manually specified from the initial guesses, with ± 200 mm for all position parameters and ± 30 mm for the PTU radiuses. The elements of all quaternions are also bounded, with values between -1 and 1 . Similar to the parameter dependent step sizes used in gradient descent, different characteristic scaling factors for the parameters are also specified. The values used are 100 for all position parameters, 10 for radius parameters, and 1 for orientation parameters. These scaling factors are defined to help the optimizer in handling parameters of different magnitudes effectively.

3.7 Aim Calibrated System at Point

When using the physical system with estimated parameters, the goal is that all rigs can be aimed at a single point. For given parameters and a point in 3D, this is the same problem that was solved in Section 3.2.2, since the PTUs in this project have encoders and can be set to a given angle.

A related scenario is when one rig is already looking at a point of interest, and the other rigs should be adjusted to look at the same point. In this case, ϕ_{pan} and ϕ_{tilt} are known for one rig, in addition to all estimated system parameters. The known mirror angles can together with the system model be used to calculate the virtual camera's pose $f(\phi_{\text{pan}}, \phi_{\text{tilt}} | \boldsymbol{\theta}) = [\mathbf{p}_{\text{virtual}}^G, \mathbf{R}_{\text{virtual}}^G]$. This specifies the line along the optical axis on which the point of interest lies. To determine where on the line this point is located requires more information, such as the distance between the measuring rig and the point. If this is known or measured in some way, the 3D-coordinates for the point of interest can be determined from the known mirror angles and distance. With the known 3D-coordinates, the mirror angles for the other rig can again be calculated as described in Section 3.2.2. How this distance is measured can be chosen based on the specific use case, but it has been outside the scope of this thesis project.

3.8 Evaluation

With a model and method for parameter estimation defined, an important remaining part is how the method as a whole can be evaluated. This can be done in multiple different ways with different types of tests. In the following sections, different evaluation methods will be presented.

3.8.1 Calibration and Test with Split Dataset

The basic structure for calibration is shown in Figure 3.11. The first step is obtaining measurements for the true system. This can be done in a simulation where $\boldsymbol{\theta}_{\text{true}}$ is defined, and $\boldsymbol{\phi}_{\text{true}} = f^{-1}(\mathbf{p}_{\text{point}}^G | \boldsymbol{\theta}_{\text{true}})$ is calculated for different points as described in Section 3.2.2. With a physical system, the pan and tilt values are manually adjusted for each PTU so that the sensors look at the points, a process that is further described in Section 3.8.3.

With $\boldsymbol{\phi}_{\text{true}}$ known, the measurements are split into two groups: $\boldsymbol{\phi}_{\text{cal}}$ for calibration and $\boldsymbol{\phi}_{\text{test}}$ for verification. The angles $\boldsymbol{\phi}_{\text{cal}}$ together with an initial parameter guess $\boldsymbol{\theta}_0$ is used as the input to the parameter estimation, which outputs the estimated parameters $\hat{\boldsymbol{\theta}}$ as well as the cost calculated for the calibration dataset, $C(\hat{\boldsymbol{\theta}} | \boldsymbol{\phi}_{\text{cal}})$. A low value for $C(\hat{\boldsymbol{\theta}} | \boldsymbol{\phi}_{\text{cal}})$ means that the parameter estimation found parameters $\hat{\boldsymbol{\theta}}$ that fit the input data well. This is positive, but does not necessarily mean that a low error can be expected for measurements in other directions. To address this, the error is also calculated for the verification set $\boldsymbol{\phi}_{\text{test}}$ which is the error used to evaluate the found parameters. This error can be presented as the cost $C(\hat{\boldsymbol{\theta}} | \boldsymbol{\phi}_{\text{test}})$ or, for easier interpretation, the approximate angular error $\varphi(\boldsymbol{\phi}_{\text{test}} | \hat{\boldsymbol{\theta}})$, as described in Section 3.3.1.

3.8.2 Use Case Testing in Simulated Environment

In addition to evaluating the estimated parameters for given measurements $\boldsymbol{\phi}_{\text{test}}$, the parameters and model can be used to predict PTU angles for different points.

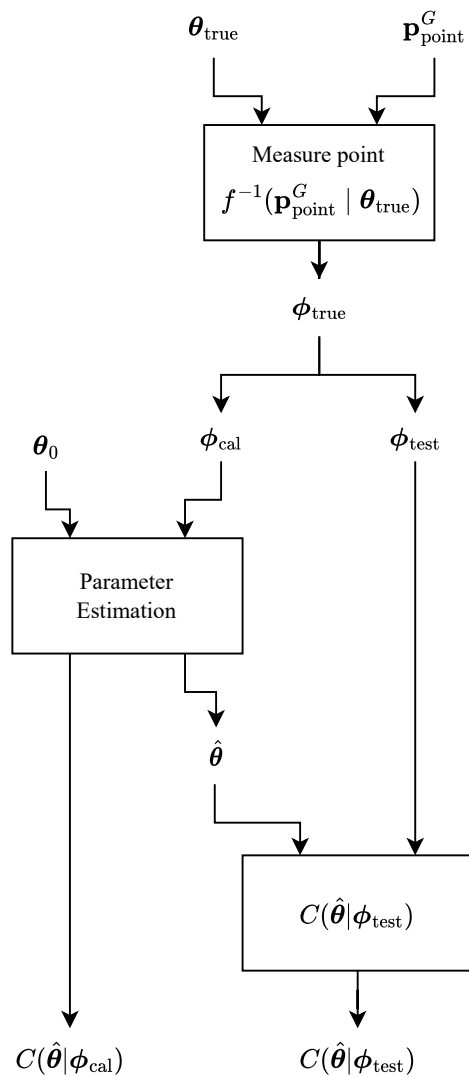


Figure 3.11: Schematic overview of parameter estimation and validation on separate measurements.

This corresponds to the proposed use case that, after calibration, one of the sensor rigs have found and is aimed at a point of interest, and the other sensor rig should also be aimed at this point. This is done as described in Section 3.7, and the steps are shown in Figure 3.12. The main step is that from a given PTU angle ϕ_{PTU_0} and distance r_{meas} to an arbitrary point of interest, an estimated point position $\hat{\mathbf{p}}_{\text{point}}^G$ is calculated based on the estimated parameters $\hat{\theta}$. The estimated values are then also used to calculate the required $\hat{\phi}_{\text{PTU}_1}$ angle to observe the estimated point position. The error of interest in this case is how well this predicted $\hat{\phi}_{\text{PTU}_1}$ will align with the input angle ϕ_{PTU_0} in the true system (in the $\hat{\theta}$ system the error will be zero). This error represents how the parameter estimation and prediction perform in this proposed use case, but requires knowledge of θ_{true} . As a result, this specific method is limited to testing in simulates systems.

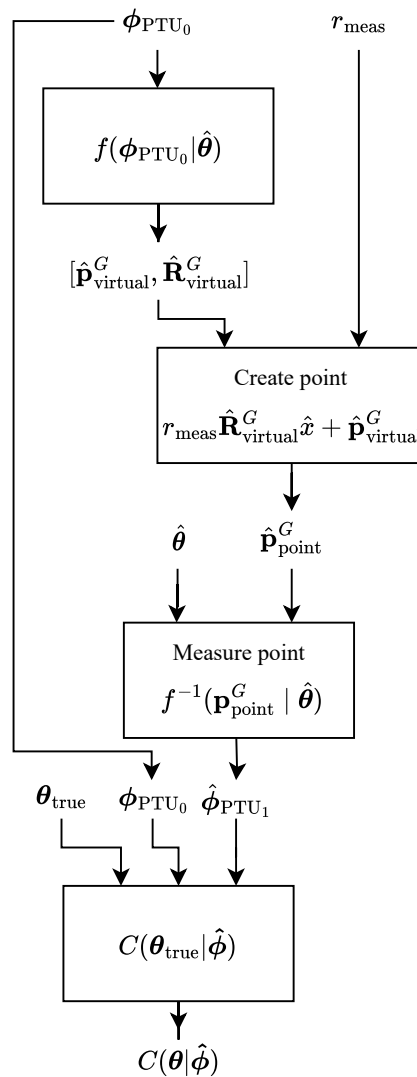


Figure 3.12: Schematic overview of testing of the predicted $\hat{\phi}_{\text{PTU}_1}$ with known θ_{true} .

3.8.3 Use Case Testing with Real-World System

To enable evaluation of use case testing with a physical system, where the value of θ_{true} is unknown, slight modifications to the method described in Section 3.8.2 have to be made. Measurements of real points are taken with both rigs for calibration and validation, where the distance to each validation point from PTU_0 is also measured. Calibration is performed on the calibration set from which $\hat{\theta}$ is estimated. This is, together with the validation measurements for PTU_0 , used to estimate $\hat{\mathbf{p}}_{\text{point}}^G$ and $\hat{\phi}_{\text{PTU}_1}$ in the same way as in the simulated environment. One way to evaluate this predicted angle $\hat{\phi}_{\text{PTU}_1}$ is to compare it with the measured ϕ_{PTU_1} . The discrepancy between these mirror angles are not directly how large the angular error for the virtual cameras are, but they are related through the model of the system.

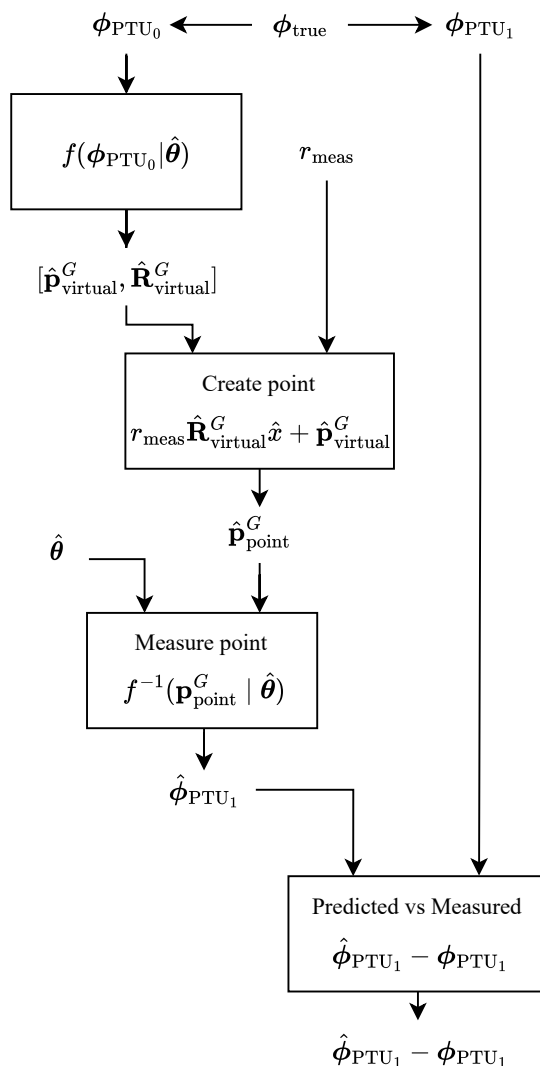


Figure 3.13: Schematic overview of testing of predicted $\hat{\phi}_{PTU_1}$ for a real-world system where θ_{true} is unknown.

3.9 Effect of Calibration Point Quantity

To assess the effect of number of calibration points on the accuracy of the solution, a series of experiments are conducted in which the number of points used for calibration, denoted by N , is varied. For each value of N , a corresponding set of calibration points is generated and used to perform parameter estimation. The resulting parameters are then evaluated on a separate set of test points that are not used during calibration. For each test point, the approximate angular error, $\varphi(\phi_n | \theta)$, is calculated according to Section 3.3.1. The mean of this error across all test points for a given N is then used as a measure of solution accuracy. This procedure is repeated multiple times for each value of N to account for outlying results. Finally, the average angular error is plotted as a function of the number of calibration points, providing insight into the relationship between calibration point set size

and solution accuracy. This analysis may help determine the minimum number of calibration points required to achieve a specific error threshold.

Both optimization methods described in Section 3.6, a manually implemented gradient descent algorithm and SciPy’s built-in least squares optimizer, are evaluated independently on a simulated system. The evaluation is performed under varying initial parameter offsets to assess the convergence behavior of each method. The offset cases include small perturbations which are applied to most or all system parameters to reflect realistic initial conditions. The detailed perturbation values for different offset cases are presented in Chapter 4.

3.10 Sensitivity to Initial Parameter Estimates

An important aspect to investigate is the tolerance of the parameter estimation method to variations in the initial parameter guesses. Ideally, the method should be able to start from an initial estimate that deviates significantly from the true values and still converge to a solution that provides accurate predictions for where to aim the sensor rigs. Understanding how accurate the initial guesses need to be for the calibration method to produce a good solution is crucial for the practical usefulness of the sensor rig setup.

Sensitivity is likely to vary between parameters, and dependencies among them may influence how errors in one parameter affect others. Exploring all combinations of initial offsets in every parameter would result in an unmanageably large number of test cases, even for a small set of parameter values. Therefore, this analysis is limited to investigating the effect of initial offsets in the orientation and position of one of the sensors.

Minimization of the cost function will be performed using both the manually implemented gradient descent algorithm and SciPy’s built-in least squares optimizer. These methods will be tested separately on a simulated system.

3.11 Setup for Real-World Testing

Real-world tests were conducted with two sensor rigs containing mostly the same hardware, but different sensors. The specific PTUs used were FLIR PTU-5 on which flat first-surface mirrors were mounted in the “top mount” configuration [12]. This means that the tilt rotation axis is placed outside of the mirror. Key specifications for this test is that the selected PTU has an accuracy of 0.1° for measured angles from its built-in encoder [12].

For the first rig, indexed as 0, a camera of model **Blackfly S GigE** with a 1.6 Mp color sensor was used [13]. For the second rig, due to resource constraints, a laser was used instead. In this setup the optical axis of the laser (where the laser ray propagates) is reflected on the mirror and can show where the equivalent virtual cameras optical axis would be aimed. This means that this setup still enabled the

same measurements used for the parameter estimation, as if two cameras had been used. For consistency with the rest of the report this “sensor” is still referred to as “Camera 1” in other sections.

Measurements were carried out by manually adjusting pan and tilt angles for both PTUs so that the optical axes of both rigs were aligned with points in 3D. The points were marked by 2 mm radius pinheads placed on a vertical board. Different board and pin positions were used resulting in a total of 38 measured points, shown in Figure 3.14. Relative to PTU_0 , these were at a radial distance of roughly 1 to 2 m, at a height of 0 to 1 m and spread up to $\approx \pm 60^\circ$ horizontally.

Measured points and system parameters in real world system

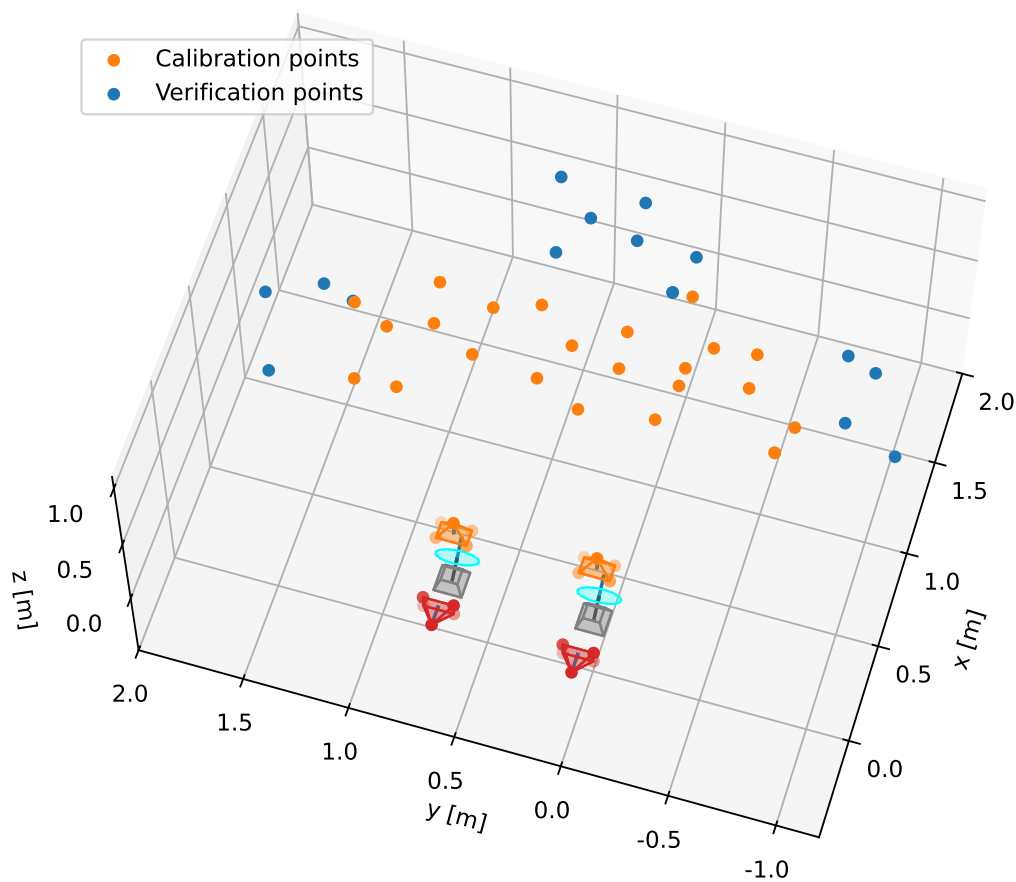


Figure 3.14: Measured point positions and system parameters for the real-world system. Measurements for these points were split into ϕ_{cal} and ϕ_{test} sets.

The parameters for the system were roughly measured with a measuring tape for positions or visually aligned for angles, and are presented in Table 3.1. The 3D position of the measured points relative to PTU_0 were also measured with a measuring tape, to enable calculation of r_{meas} for the case described in Section 3.8.3. This is shown in Figure 3.14.

Table 3.1: Summary of measured PTU and sensor parameters for the real-world system. Positions and radiuses are in millimeters [mm], and orientations in degrees [deg]. Values for r_1 and r_2 are taken from drawings of the PTU and mirror mount.

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
PTU 0	[0, 0, 0]	[0, 0, 0]	[7.2, 108.7]
Camera 0	[40, 0, 500]	[0, 90, 0]	–
PTU 1	[0, 660, 0]	[0, 0, 0]	[7.2, 108.7]
Camera 1	[50, 660, 450]	[0, 90, 0]	–

3.11.1 Measurement Accuracy for Real-World Testing

The accuracy of measurements in the real-world system were mainly impacted by the accuracy of the PTUs and the visual alignment, which differed slightly between the two rigs. For visual alignment of the laser sensor rig, the limiting factor was the spot size of the laser. Its width made it hard to differentiate the exact center of the lasers optical axis when the it hit different parts of the pin head. This lead to an estimated accuracy of 1 mm, which at 1 m distance is $\approx 0.06^\circ$.

For the camera sensor rig, the camera feed resolution was high and made it easier to tell which part of the pin head which was in the center. The limiting factor was instead that fine adjustments of pan and tilt were time consuming. A compromise was made where measurements were performed if the image center was aimed within one fourth of the pin head radius from its center. This corresponds to 0.5 mm, which at 1 m distance leads to an accuracy of $\approx 0.03^\circ$.

Including the PTU’s and the estimated alignment accuracy, an estimated worst case error can be calculated:

$$\underbrace{(0.1^\circ + 0.06^\circ)}_{\text{Laser rig}} + \underbrace{(0.1^\circ + 0.03^\circ)}_{\text{Camera rig}} = 0.29^\circ,$$

when using or comparing measurements from both sensor rigs. It should be noted that this value is dependent on multiple estimations and more thorough testing could result in a different value.

3.12 Simulated Calibration and Test Scenario

For several of the tests, a standard set of points and system parameters can be used for testing in a simulated environment. The system parameters used for these scenarios are presented in Table 3.2.

In both the calibration and verification scenarios, the points are placed with the help of spherical coordinates. The calibration scenario consists of a total of 35 points placed in a grid of different pan and tilt angles. Pan varies in 7 equal steps

Table 3.2: True PTU and camera parameters used in simulations. Positions are in millimeters [mm], orientations in degrees [deg], and radiuses in millimeters [mm].

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
PTU 0	[0, 0, 0]	[0, 0, 0]	[10, 100]
Camera 0	[50, 0, 500]	[0, 90, 0]	–
PTU 1	[0, 500, 0]	[0, 0, 0]	[10, 100]
Camera 1	[50, 500, 500]	[0, 90, 0]	–

from -65.7° to 65.7° , and tilt in 5 steps between 5° and 35° . The radius for each point is randomly picked from a uniform distribution from 1500 ± 250 mm giving the points a radial spread. This point distribution was chosen as an example coverage for measuring objects in front of the sensor rigs. The radial spread is introduced since an exact fixed radius is not expected for real-world calibration. The resulting system and points are shown in Figure 3.15.

The verification system is created similarly but in a grid where pan varies in 4 equal steps between from -50° to 50° , and tilt in 3 steps between 5° and 35° , resulting in different angles than the calibration system. For this scenario all points are placed at a radius of 2000 mm. This system can be seen in Figure 3.16. The values for the calibration and verification system are summarized in Table 3.3.

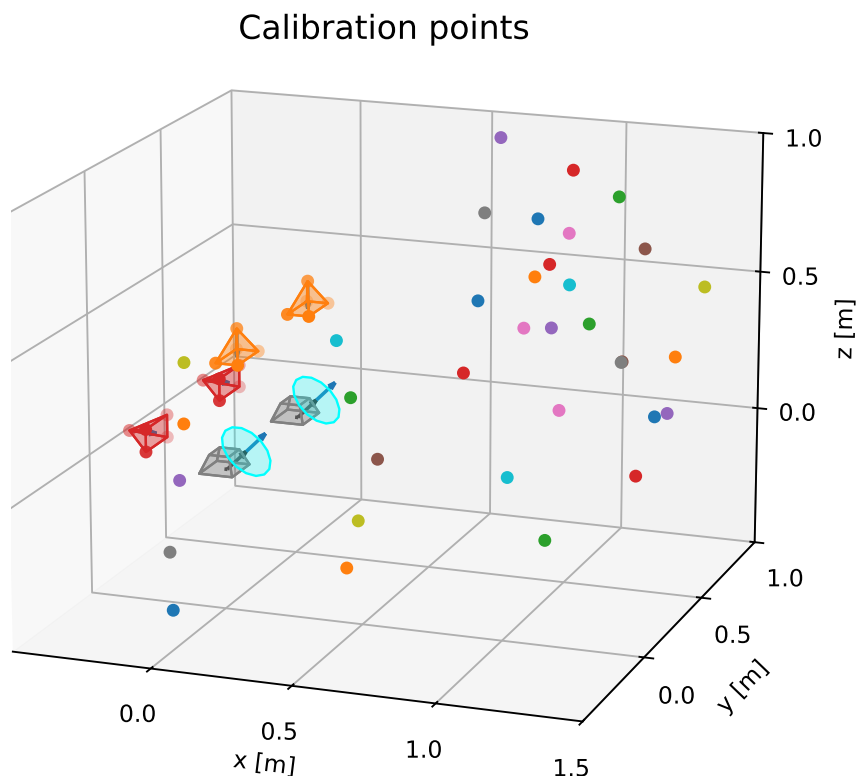


Figure 3.15: Illustration of the simulated calibration system. The component positions as well as the points placed with varying radiuses can be seen.

Table 3.3: Variable ranges for placement of points in the standard calibration and validation systems.

System	Radial range	Pan range	Pan steps	Tilt range	Tilt steps
Calibration	1.5 ± 0.25 m	$-67.5^\circ, 67.5^\circ$	7	$5^\circ, 35^\circ$	5
Verification	2 m	$-50^\circ, 50^\circ$	4	$5^\circ, 35^\circ$	3

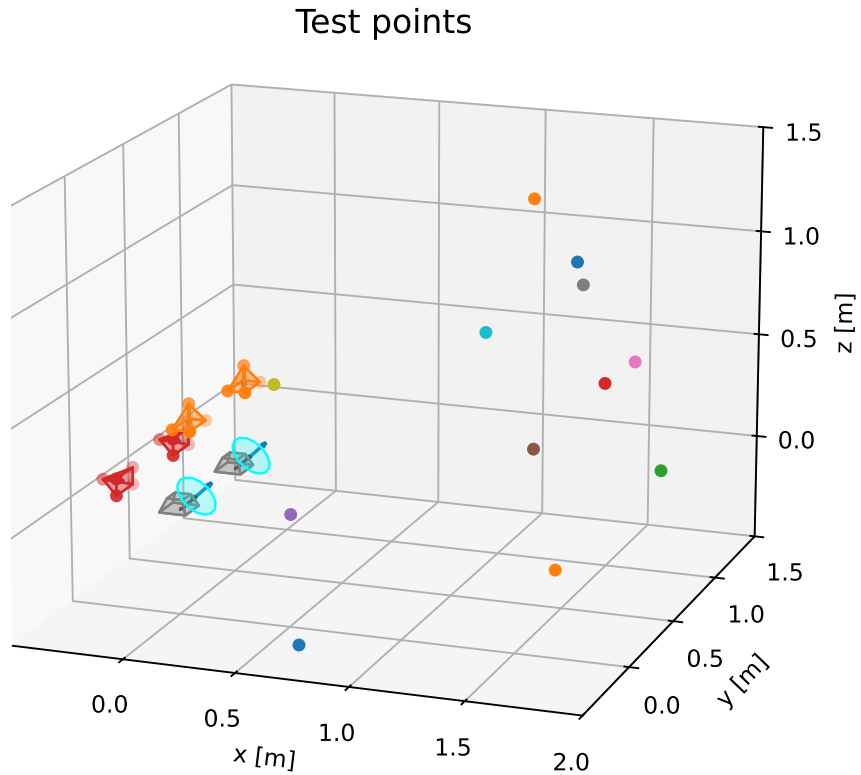


Figure 3.16: Illustration of the simulated test system. The component positions as well as the point positions are shown.

4

Results

The following chapter presents results from the evaluation of the calibration method. The evaluation covers several aspects, including the effect of the number of calibration points on the resulting angular error, use case testing in simulated environments, and the method’s sensitivity to errors in initial parameter guesses. These aspects are investigated using two optimization approaches for minimizing the cost function: gradient descent and SciPy’s least squares optimizer. Additionally, the use case testing is performed on a real-world scenario.

4.1 Effect of Calibration Point Quantity

The dependence of solution accuracy on the number of calibration points is evaluated using both the manually implemented gradient descent algorithm and SciPy’s built-in least squares optimizer, as described in Section 3.9. The accuracy for each number of calibration points is measured using the value of the cost function (Equation 2.3), which is then converted into a mean angular error, as described in Section 3.3.1, to make the result easier to interpret. The evaluation is conducted on a simulated system, consisting of one calibration system and one verification system, as described in Section 3.12. Both optimization methods are evaluated on a common dataset with small perturbations to the parameters, as presented in Table 4.1. Additionally, SciPy’s optimizer is evaluated on a set with slightly larger parameter offsets, shown in Table 4.2, which is the offset case that is applied in the simulated use case testing (Section 4.2). For all cases, parameter estimation is repeated five times for each number of calibration points, N , and the median value for each N is highlighted to reduce the influence of outlying results. The results of each optimization method are presented below.

4.1.1 Effect of Calibration Points: Gradient Descent

The mean angular error as function of number of calibration points is presented in Figure 4.1, where all five samples and their median value for each N are shown. For the initial parameter offsets described in Table 4.1, the error appears to stabilize at approximately 0.2° for $N \geq 5$, i.e. when five or more points are used for calibration. However, for most values of N , outlier samples with significantly higher errors are present, as shown in the figure. Notably, when 15 calibration points are used, the

Table 4.1: Small positional and rotational (RPY) offsets applied to the initial parameter estimates, $\hat{\theta}_0$, for rig components used in both the manually implemented gradient descent algorithm and SciPy’s built-in least squares optimizer.

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
Camera 0	$[10, -10, -10]$	$[2, 2, 2]$	–
PTU 0	$[0, 0, 0]$	$[0, 0, 0]$	$[0, 0]$
Camera 1	$[10, 10, 10]$	$[2, 2, 2]$	–
PTU 1	$[-10, -10, -10]$	$[2, 2, 2]$	$[0, 0]$

Table 4.2: Position, rotation (RPY), and radius offsets for each rig component for a more challenging initial offset case. The difference compared to the values in Table 4.1 is increased offset values and introduced offsets for the PTU radiuses.

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
Camera 0	$[-100, 100, -100]$	$[5, 5, 5]$	–
PTU 0	$[0, 0, 0]$	$[0, 0, 0]$	$[0.1, -1]$
Camera 1	$[100, -100, 100]$	$[5, -5, 5]$	–
PTU 1	$[-100, 100, -100]$	$[-3, -3, -3]$	$[-0.1, 1]$

median obtains a mean angular error of almost 8.7° for this specific offset scenario and under the simulated calibration and test conditions described in Section 4.1. In this calibration scenario, the outliers seems to become less common when a larger number of calibration points are used, as shown in Figure 4.1.

4.1.2 Effect of Calibration Points: SciPy Optimizer

Figures 4.2 and 4.3 present the corresponding evaluation of the mean angular error as function of the number of calibration points, like in Section 4.1.1, but using SciPy’s built-in least squares optimizer to minimize the cost function.

First, the SciPy optimizer is tested using the same initial parameter offsets as in the gradient descent case, as defined in Table 4.1. The results, shown in Figure 4.2, indicate that the mean angular error decreases with an increasing number of calibration points. Within the tested range ($1 \leq N \leq 35$), no clear convergence is observed. At $N = 35$, the median sample of the mean angular error reaches approximately $2 \cdot 10^{-4}^\circ$, for this specific scenario.

In a second experiment, the optimizer is evaluated on a scenario with larger initial parameter offsets, as specified in Table 4.2. In addition to larger perturbations in camera and PTU parameters, this case also includes offsets in the PTU radiuses. The corresponding result is presented in Figure 4.3, which shows a similar trend to Figure 4.2, but with mean angular errors of different magnitudes.

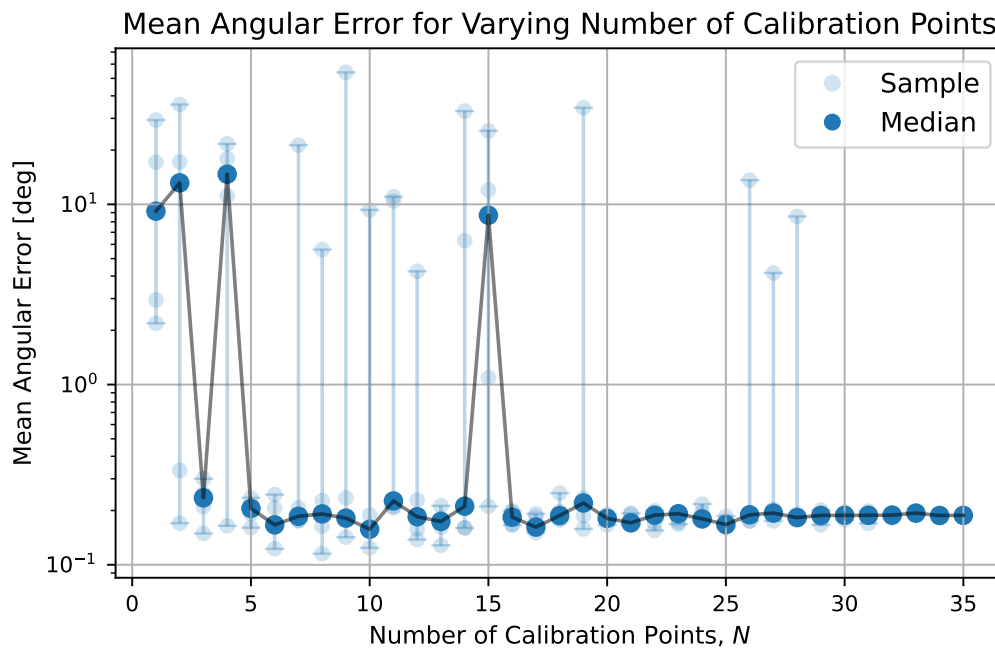


Figure 4.1: Dependence of mean angular error on the number of calibration points using gradient descent optimization. Calibration is performed using initial parameter offsets specified in Table 4.1, with 10 mm offsets applied to camera and PTU positions, and 2° offsets applied to their orientations.

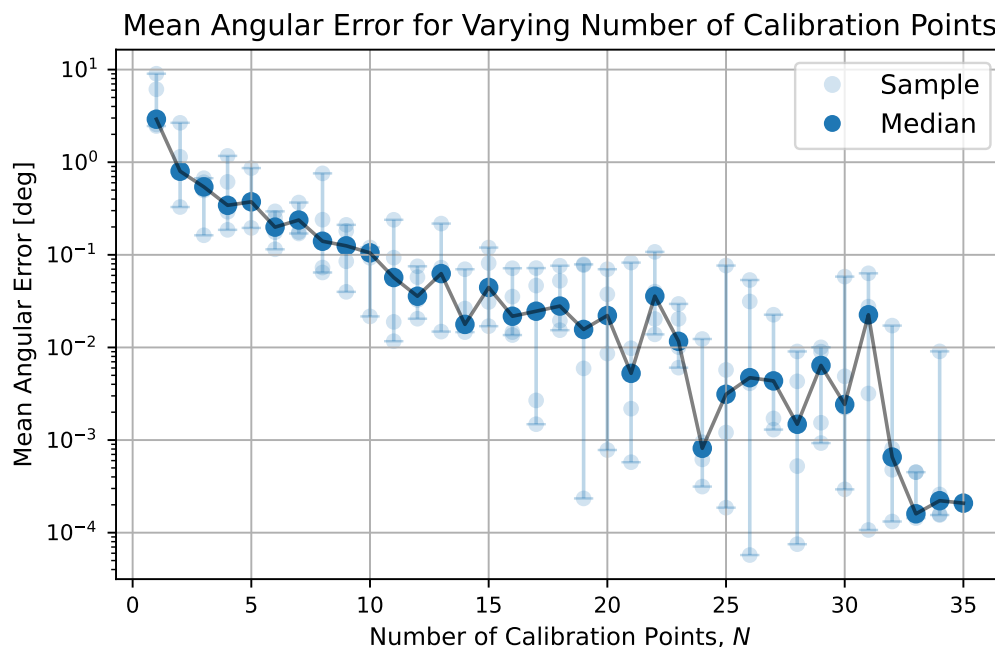


Figure 4.2: Dependence of mean angular error on the number of calibration points using SciPy's built-in least squares optimizer. Calibration is performed using the same initial parameter offsets as in Figure 4.1: 10 mm offsets applied to camera and PTU positions, and 2° offsets applied to their orientations (see Table 4.1).

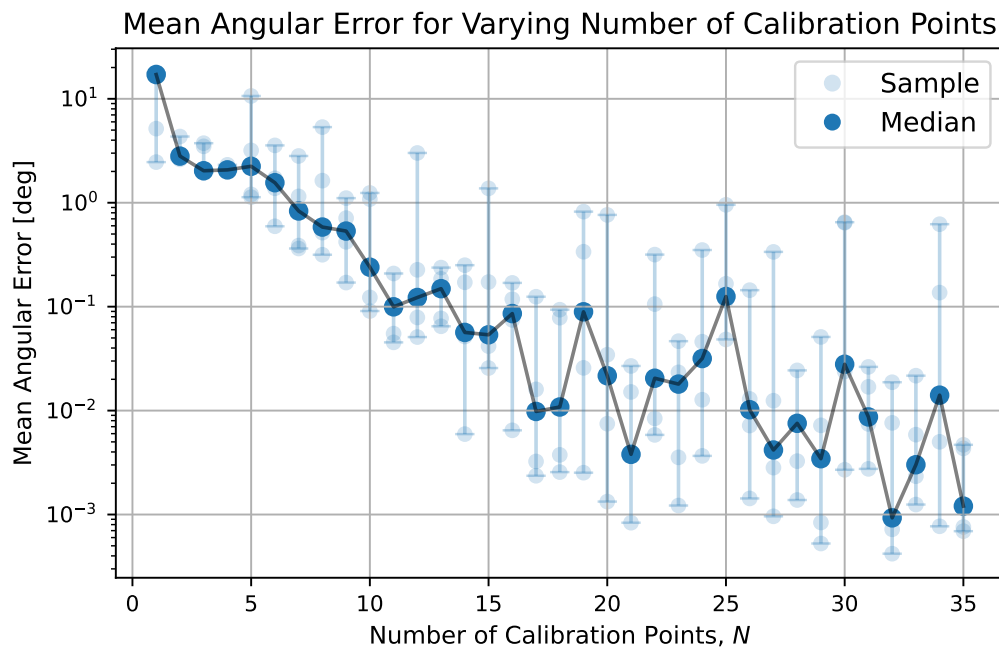


Figure 4.3: Dependence of mean angular error on the number of calibration points using SciPy’s built-in least squares optimizer. In this case, calibration is performed with larger initial parameter offsets than in Figure 4.2, as specified in Table 4.2. These include increased positional offsets for cameras and PTUs, as well as additional offsets to the PTU radiuses.

4.2 Use Case Testing in Simulated Environment

In this test, the method described in Section 3.8.2 is used. The approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ is calculated for two cases: for a fixed r_{meas} when ϕ_{PTU_0} is a grid of different pan and tilt values, as well as for different r_{meas} and pan values but fixed tilt. Ranges for the variables in these cases are presented in Tables 4.3 and 4.4. Measurements of r_{meas} up to 100 m are included to investigate how the method performs at longer distances as well as close range.

Table 4.3: Value ranges for evaluation with fixed r_{meas} and varying PTU₀ pan and tilt angles. The measurement range r_{meas} is changed for the two different cases.

r_{meas}	Pan range	Pan resolution	Tilt range	Tilt resolution
10 m	$-90^\circ, 90^\circ$	5°	$46^\circ, 60^\circ$	2°
100 m	$-90^\circ, 90^\circ$	5°	$46^\circ, 60^\circ$	2°

Table 4.4: Value ranges for evaluation with fixed tilt = 0° and varying pan and r_{meas} for PTU₀. The range and resolution of r_{meas} differs between the cases.

r_{meas} range	r_{meas} resolution	Pan range	Pan resolution	Tilt value
1 – 10 m	0.5 m	$-90^\circ, 90^\circ$	5°	0°
1 – 100 m	20 even steps	$-90^\circ, 90^\circ$	5°	0°

The parameter estimation is again performed for the standard calibration system from Section 3.12 but also for three additional systems. These systems vary at which distance and in which tilt range the points are generated. The ranges are presented in Table 4.5. These are included to investigate how performance for different distances are impacted by different calibration ranges. Cases where the tilt range is limited to $[0^\circ, 1.5^\circ]$ are included since real measurements at 100 m radius and 35° can be difficult to achieve without a flying calibration target.

Table 4.5: Ranges for placement of points in the additional calibration systems. These increase the tested scenarios to calibration at 100 m and limited spread in tilt.

Radial range	Pan range	Pan steps	Tilt range	Tilt steps
100 ± 25 m	$-67.5^\circ, 67.5^\circ$	7	$5^\circ, 35^\circ$	5
100 ± 25 m	$-67.5^\circ, 67.5^\circ$	7	$0^\circ, 1.5^\circ$	5
1.5 ± 0.25 m	$-67.5^\circ, 67.5^\circ$	7	$0^\circ, 1.5^\circ$	5

4.2.1 Simulated Use Case: Gradient Descent

Test with gradient descent were carried out with the offset θ_{offset} specified in Table 4.1. Heatmaps for the approximate angle error are shown for when calibrated with the standard calibration system are shown in Figures 4.4, 4.5, 4.6 and 4.7. The mean errors for all calibration systems are shown in Table 4.6.

Table 4.6: Mean approximate angular errors for prediction in all tested cases with gradient descent. The errors for the additional calibration cases are high.

Calibration scenario			Mean approximate angular errors [deg]			
Radius [m]	Pan [deg]	Tilt [deg]	Pan tilt 10 m	Pan radius 1 – 10 m	Pan tilt 100 m	Pan radius 1 – 100 m
1.5 ± 0.25	$[-67.5, 67.5]$	$[5, 35]$	1.42	0.53	5.98	1.32
100 ± 25	$[-67.5, 67.5]$	$[5, 35]$	41.8	42.7	37.3	32.3
100 ± 25	$[-67.5, 67.5]$	$[0, 1.5]$	42.8	39.8	42.8	39.0
1.5 ± 0.25	$[-67.5, 67.5]$	$[0, 1.5]$	29.9	22.7	26.0	20.7

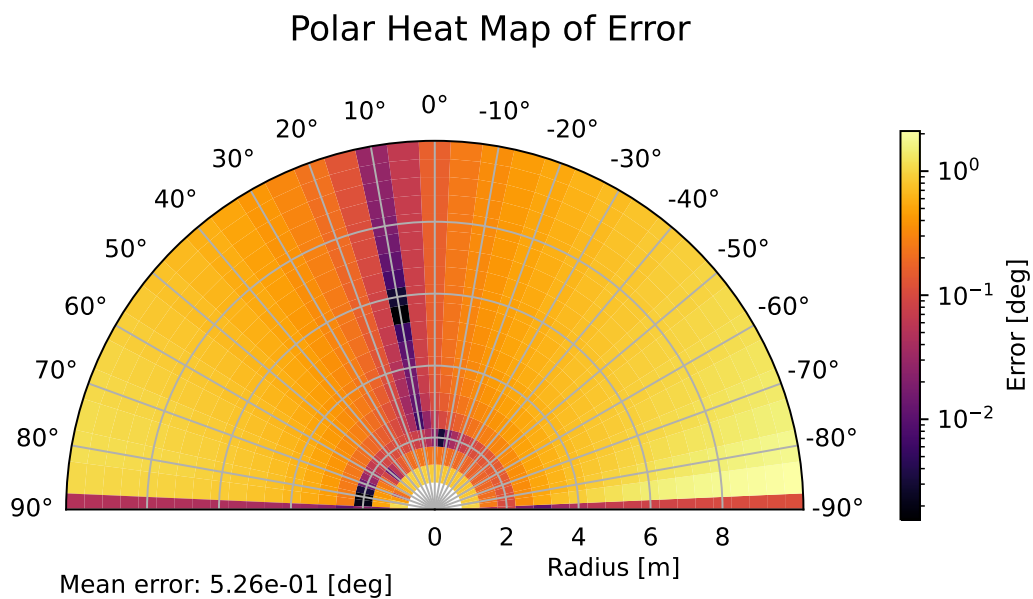


Figure 4.4: Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ for different pan and $r_{\text{meas}} \in [1, 10 \text{ m}]$, for calibration with gradient descent. Parameters are estimated for the standard calibration system. The error is quite uniform except for some close points that seem to extend towards 10° and $\pm 90^\circ$ where the error is lower.

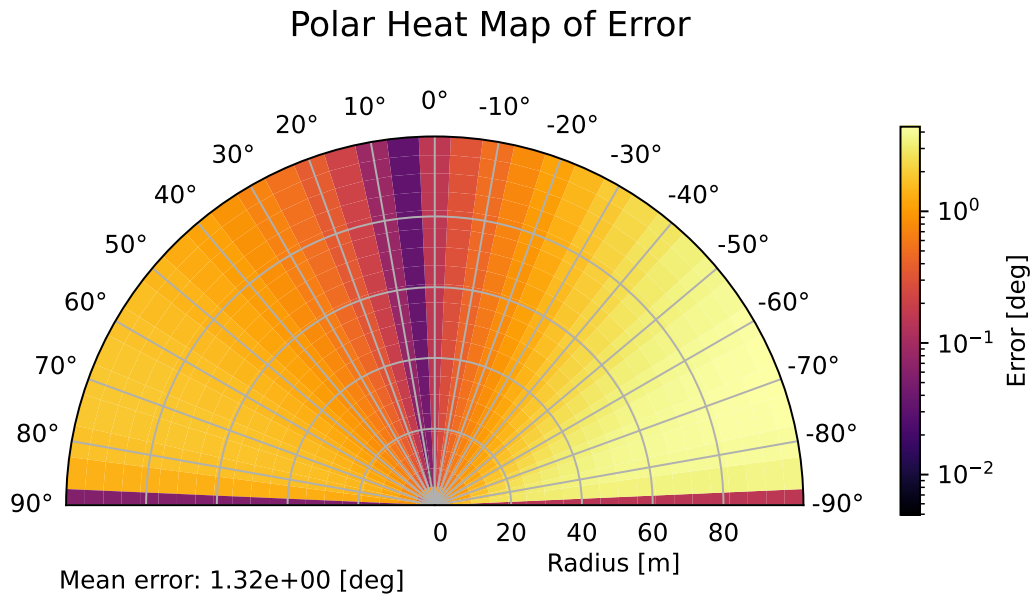


Figure 4.5: Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ for different pan and $r_{\text{meas}} \in [1, 100] \text{ m}$, for calibration with gradient descent. Parameters are estimated for the standard calibration system. A lower error can be seen for angles around 5° and $\pm 90^\circ$.

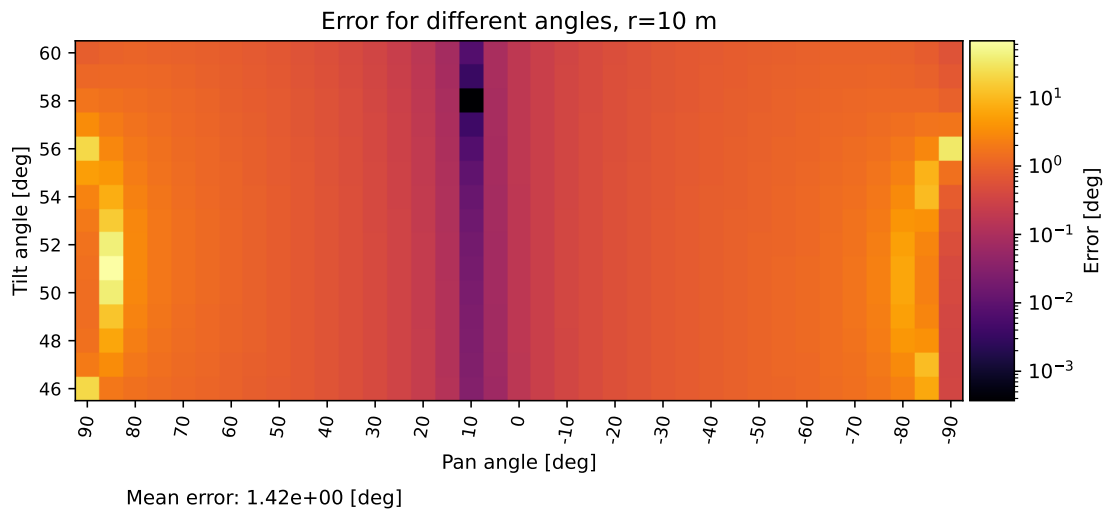


Figure 4.6: Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ for different pan and tilt values when $r_{\text{meas}} = 10 \text{ m}$, for calibration with gradient descent. Parameters are estimated for the standard calibration system. Lower error for pan angles around 10° can be observed while some directions close to $\pm 90^\circ$ pan angle give elevated error.

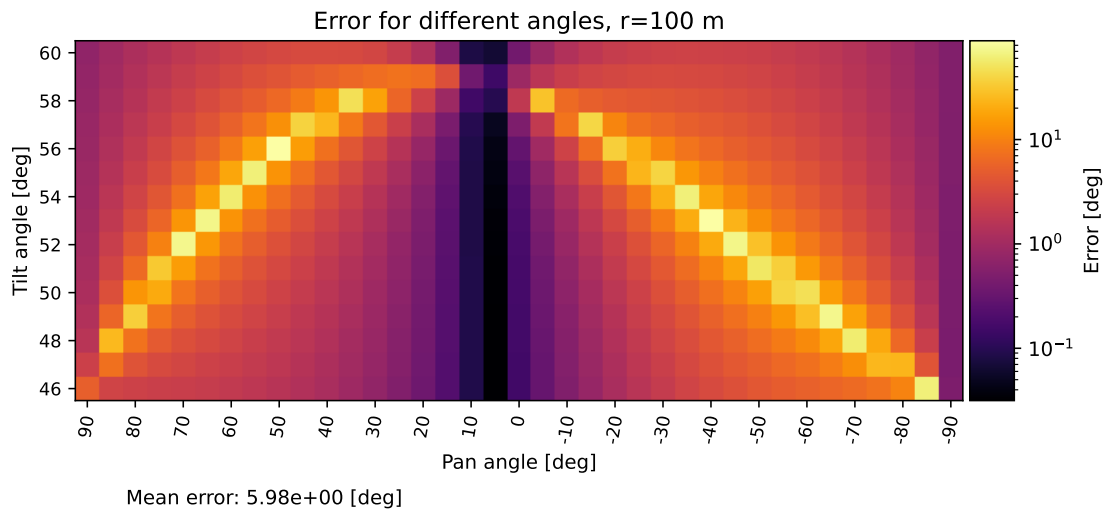


Figure 4.7: Heatmap displaying the approximate angular error $\varphi(\phi_{PTU_0}, \hat{\phi}_{PTU_1} | \theta)$ for different pan and tilt values when $r_{\text{meas}} = 100$ m, for calibration with gradient descent. Parameters are estimated for the standard calibration system. Some combinations of pan and tilt angles give elevated error.

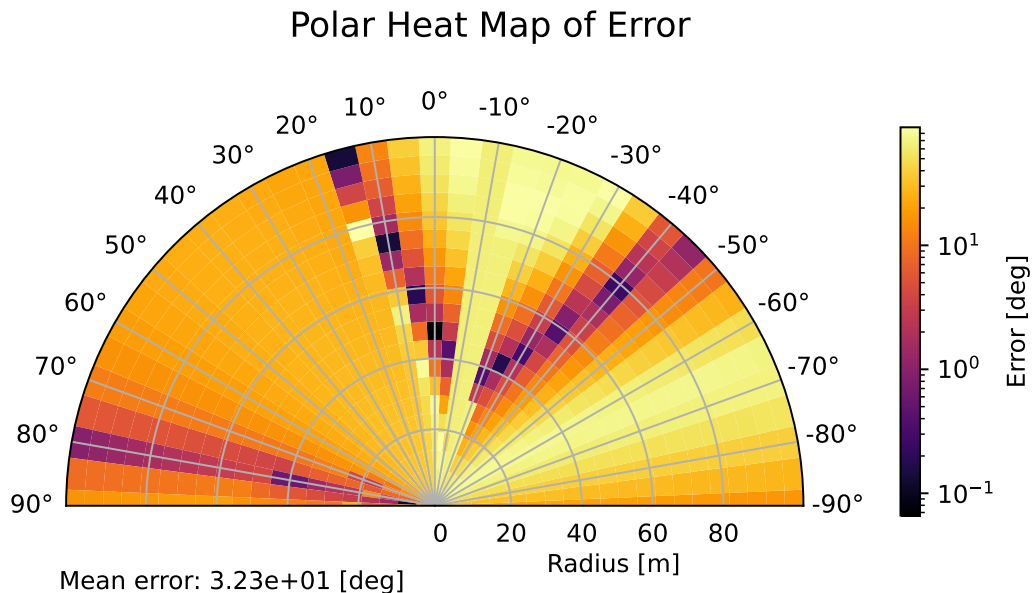


Figure 4.8: Heatmap displaying the approximate angular error $\varphi(\phi_{PTU_0}, \hat{\phi}_{PTU_1} | \theta)$ for different pan and $r_{\text{meas}} \in [1, 100]$ m, for calibration with gradient descent. Parameters are estimated for the first additional case in Table 4.5 with points 100 ± 25 m away. While some sampled positions give a low error, the overall error is substantially higher than the standard calibration case.

4.2.2 Simulated Use Case: SciPy’s Least Squares Optimizer

Test with SciPy’s least squares optimizer were carried out with the increased offset θ_{offset} specified in Table 4.2. Heatmaps for the approximate angular error are shown for when calibrated with the standard calibration system are shown in Figures 4.9, 4.10, 4.11 and 4.12. The mean errors for all calibration systems are shown in Table 4.7.

Table 4.7: Mean approximate angular error for prediction in all tested cases with SciPy least squares.

Calibration scenario			Mean approximate angular errors [deg]			
Radius [m]	Pan [deg]	Tilt [deg]	Pan tilt 10 m	Pan radius 1 – 10 m	Pan tilt 100 m	Pan radius 1 – 100 m
1.5 ± 0.25	$[-67.5, 67.5]$	$[5, 35]$	0.08	0.06	0.11	0.11
100 ± 25	$[-67.5, 67.5]$	$[5, 35]$	3.73	6.44	0.30	1.58
100 ± 25	$[-67.5, 67.5]$	$[0, 1.5]$	2.94	5.37	0.32	1.47
1.5 ± 0.25	$[-67.5, 67.5]$	$[0, 1.5]$	0.19	0.05	0.75	0.11

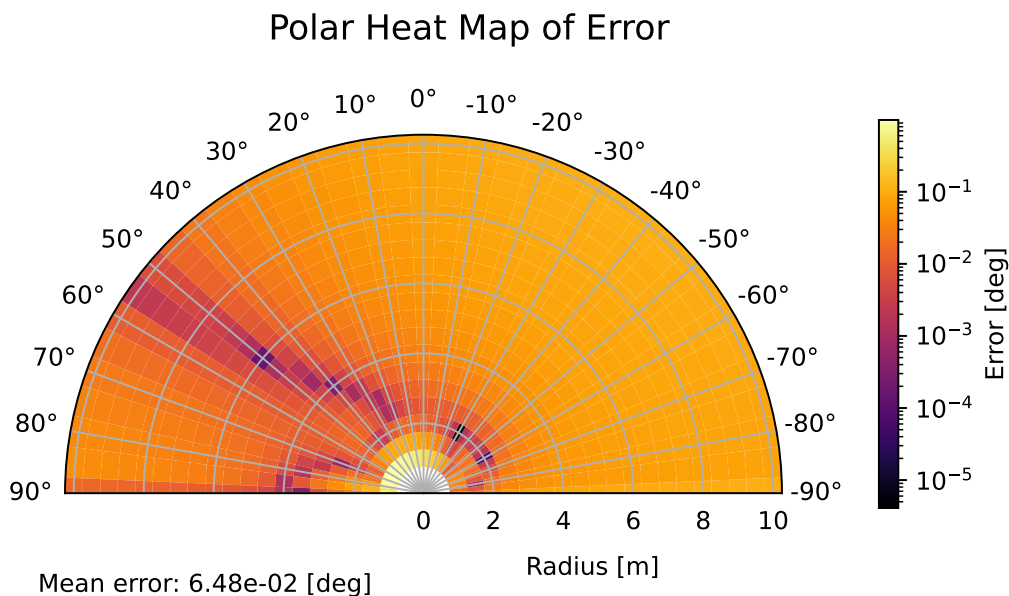


Figure 4.9: Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ for different pan and $r_{\text{meas}} \in [1, 10 \text{ m}]$, for calibration with SciPy least squares. Parameters are estimated for the standard calibration system. The error is quite uniform except for some close points that seem to extend towards 55° where the error is lower.

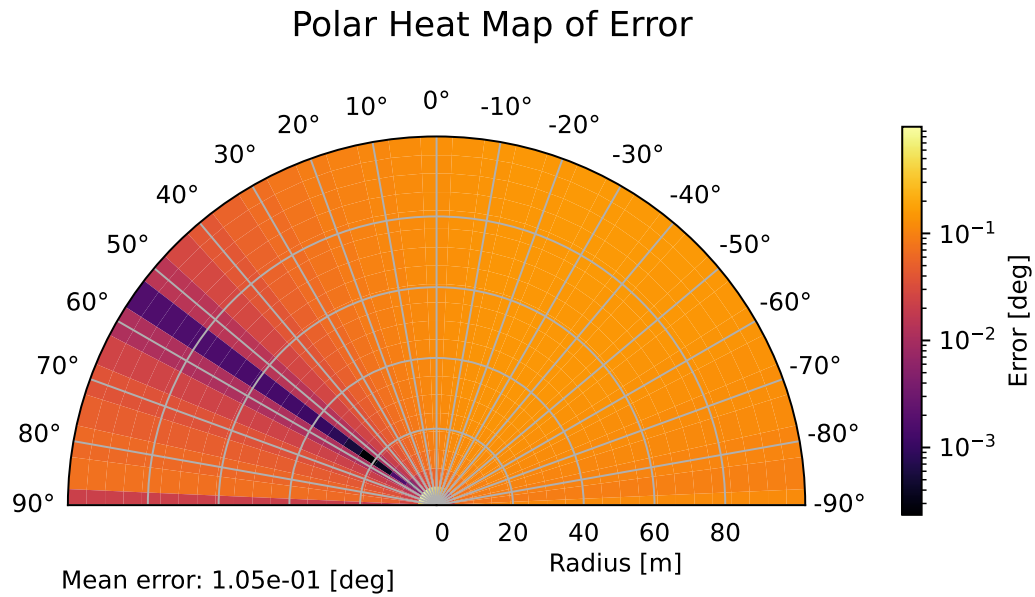


Figure 4.10: Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ for different pan and $r_{\text{meas}} \in [1, 100 \text{ m}]$, for calibration with SciPy least squares. Parameters are estimated for the standard calibration system. A lower error can be seen for angles around 55° .

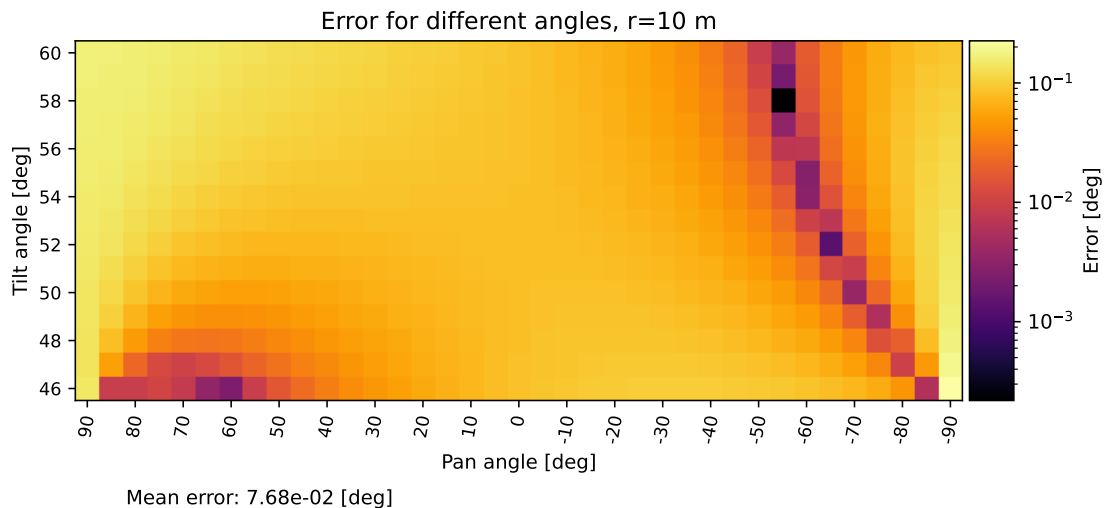


Figure 4.11: Heatmap displaying the approximate angular error $\varphi(\phi_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ for different pan and tilt values when $r_{\text{meas}} = 10 \text{ m}$, for calibration with SciPy least squares. Parameters are estimated for the standard calibration system.

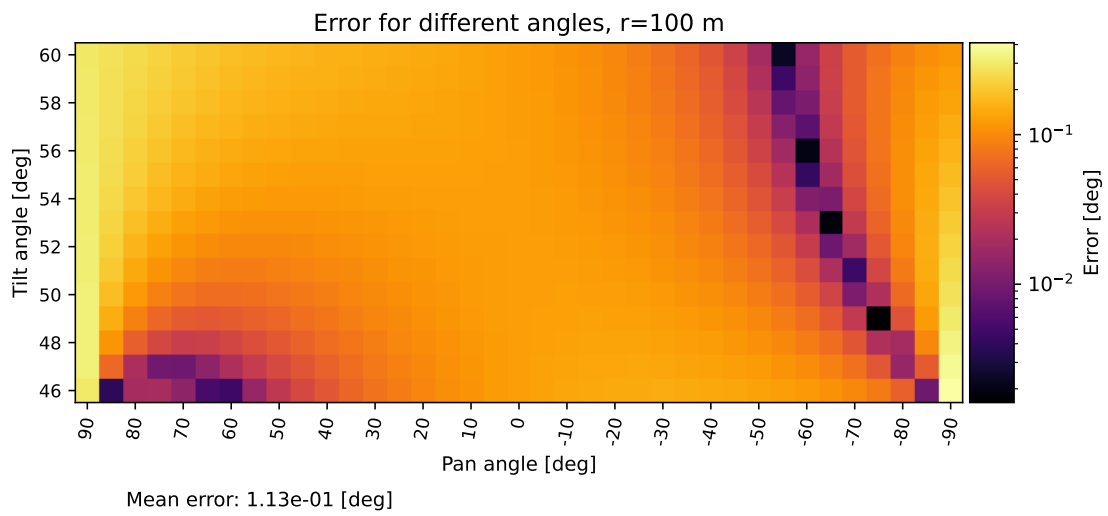


Figure 4.12: Heatmap displaying the approximate angular error $\varphi(\hat{\phi}_{\text{PTU}_0}, \hat{\phi}_{\text{PTU}_1} | \theta)$ for different pan and tilt values when $r_{\text{meas}} = 100$ m, for calibration with SciPy least squares. Parameters are estimated for the standard calibration system.

4.3 Use Case Testing with Real-World System

The test described in Section 3.8.3 was carried out for the system and measurements described in Section 3.11. SciPy’s least squares optimizer was used together with the manually measured system parameters as θ_0 . The recorded measurements were split in to ϕ_{cal} and ϕ_{test} , with 15 of the more distant points selected for the test set. This set contains three different groupings (board positions), color-coded as red (middle), green (left), and blue (right). Predicted and real-world measured ϕ_{PTU_1} are used to calculate the corresponding virtual camera axes, something which is shown in Figure 4.13.

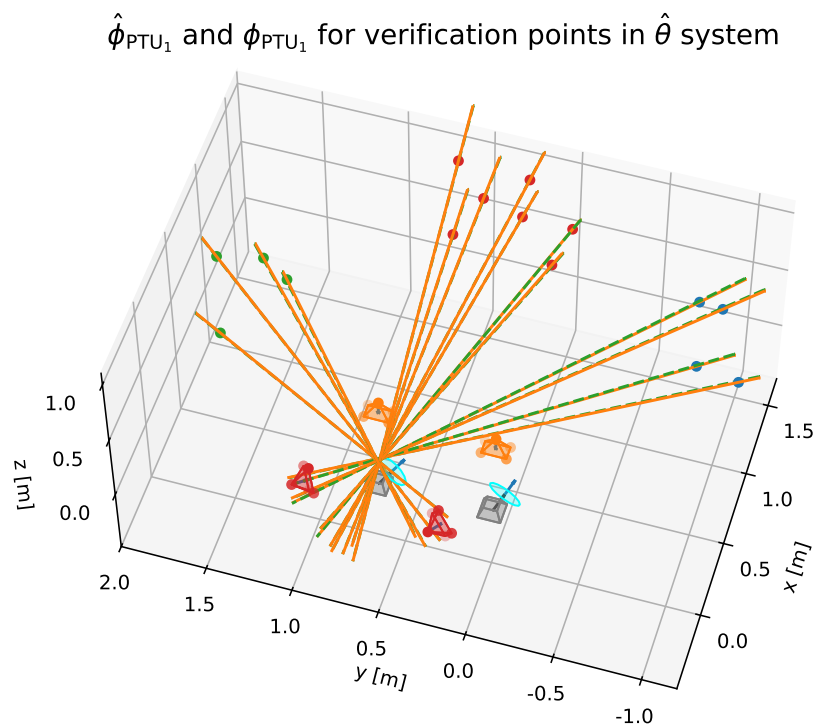


Figure 4.13: Illustration of estimated virtual cameras optical axes for $\hat{\phi}_{\text{PTU}_1}$ (marked in orange) and ϕ_{PTU_1} (marked in green) for real-world verification measurements. It can be seen that the axes are similar for the two cases. This is calculated in the $\hat{\theta}$ system, so it does not exactly represent the real-world optical axes, but they should be similar.

The differences between the predicted and real-world measured PTU mirror angles after calibration are illustrated in Figure 4.14. Higher errors are observed for pan than for tilt but all values are within $\pm 0.3^\circ$.

In addition to the prediction error, other results can be presented for this test scenario. One example is that the mean approximate angular error can still, similar to previous tests, be calculated for the verification points, which is 0.046° . Furthermore, the found parameters $\hat{\theta}$ and the difference to the measured θ_0 are presented in Tables 4.8 and 4.9, respectively. Most differences are relatively small except for roll, yaw and z position values, something which is commented on in Section 5.3.

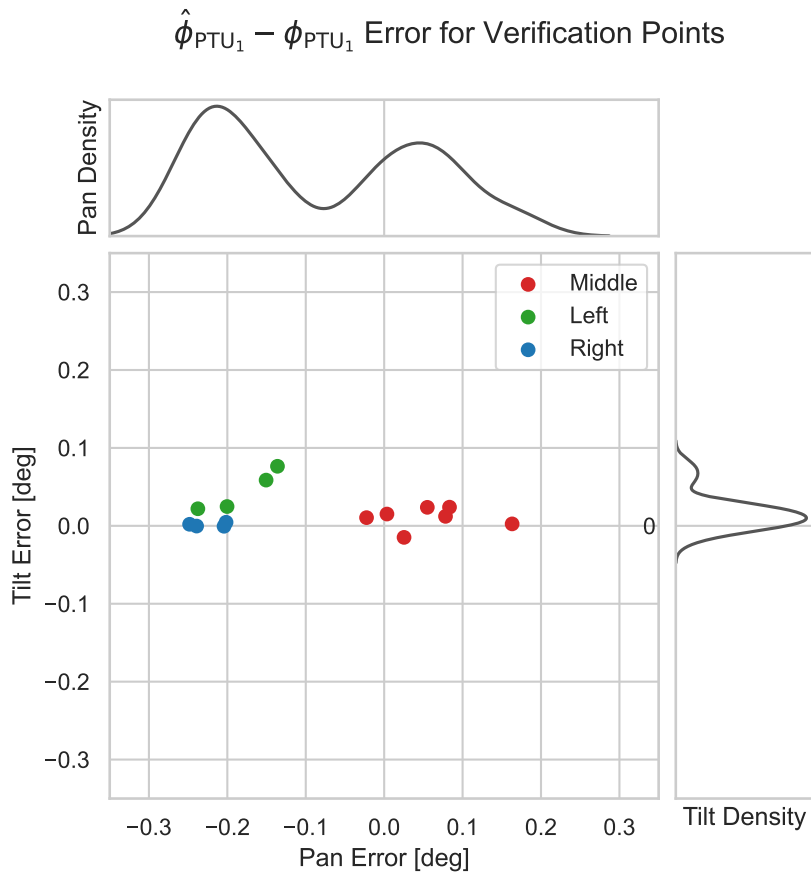


Figure 4.14: Comparison of the predicted $\hat{\phi}_{PTU_1}$ and real-world measured ϕ_{PTU_1} for the pan and tilt axis. Tested verification points are color coded according to the different group positions. Arbitrarily scaled kernel density plots are included to help illustrate the distribution.

Finally, $\hat{\phi}_{PTU_1}$ predictions can also be made using the manually measured θ_0 parameters without performing the parameter estimation. Prediction error for this is shown in Figure 4.15, where larger errors are generally seen.

Table 4.8: Summary of estimated PTU and camera parameters for the real-world system. Positions and radiuses are in millimeters [mm], orientations in degrees [deg].

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
PTU 0	[0, 0, 0]	[0, 0, 0]	[9.2, 109.1]
Camera 0	[36.35, -6.30, 616.8]	[57.65, 89.37, 57.68]	-
PTU 1	[-13.23, 648.3, 4.73]	[11.94, 47.08, -24.87]	[5.5, 108.8]
Camera 1	[58.69, 658.8, 650.0]	[100.4, 89.26, 100.4]	-

Table 4.9: Difference between estimated and measured PTU and camera parameters (Tables 4.8 and 3.1) for the real-world system. Positions and radiuses are in millimeters [mm], orientations in degrees [deg].

Component	$\Delta[x, y, z]$ [mm]	Δ RPY [deg]	$\Delta[r_1, r_2]$ [mm]
PTU 0	[0.00, 0.00, 0.00]	[0.00, 0.00, 0.00]	[2.00, 0.44]
Camera 0	[-3.65, -6.30, 116.80]	[57.65, -0.63, 57.68]	-
PTU 1	[-13.23, -11.71, 4.73]	[0.12, 0.47, -0.25]	[-1.74, 0.09]
Camera 1	[8.69, -1.23, 200.00]	[100.40, -0.74, 100.40]	-

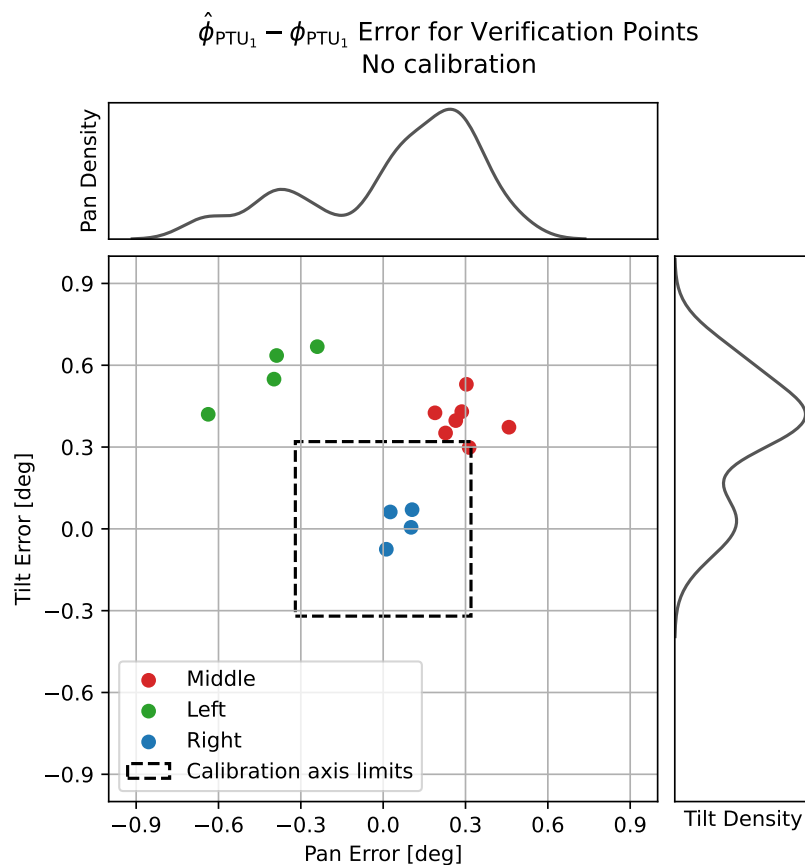


Figure 4.15: Comparison of the predicted $\hat{\phi}_{PTU_1}$ and real-world measured ϕ_{PTU_1} for no calibration. Larger prediction errors are observed than in the calibrated case in Figure 4.14. Note that the axis limits in this plot have been increased.

4.4 Sensitivity to Initial Parameter Estimates

The sensitivity of the parameter estimation process to initial parameter estimates is evaluated by introducing systematic offsets to the position and orientation parameters of Camera 1. This evaluation is conducted separately for both optimization methods used to minimize the cost function. The same simulated calibration and test system described in Section 4.1 is used throughout the parameter estimation process.

Each offset scenario includes perturbations in all parameters for Camera 1, except for the roll angle, which does not influence the solution. This leaves three positional and two rotational parameters to evaluate. For each parameter, three offset values are tested: a positive offset, a negative offset of equal magnitude, and no offset. This results in a total of 243 combinations of initial offsets, each of which is used as a starting point for parameter estimation in the evaluated scenario.

4.4.1 Sensitivity for Gradient Descent Optimization

The sensitivity to initial parameter estimates using the manually implemented gradient descent algorithm to minimize the cost function was evaluated for two different sets of offsets in Camera 1: one with smaller offsets and one with larger offsets. The specific perturbations for these two cases are presented in Tables 4.10 and 4.11, respectively.

The resulting mean angular error (used as a measure of the cost, as described in Section 3.3.1) are computed for all combinations of initial offsets. These results are sorted by error magnitude and presented in Figures 4.16 and 4.17 for the smaller and larger perturbation cases, respectively.

For the case with smaller initial perturbations (Figure 4.16), all offset combinations result in a mean angular error below 1° , and 97.5% of the combinations result in an error below 0.1° . However, for the case with the larger initial perturbations (Figure 4.17), 64.2% of the combinations achieve an error below 1%, and only 7.4% give an error below 0.1%. A summary of these error distributions is provided in Table 4.12.

Table 4.10: Small positional and rotational (RPY) offsets applied to the initial parameter estimates, $\hat{\theta}_0$, for Camera 1. This offset configuration is used in one of the tolerance evaluations of the gradient descent algorithm.

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
Camera 0	[0, 0, 0]	[0, 0, 0]	–
PTU 0	[0, 0, 0]	[0, 0, 0]	[0, 0]
Camera 1	[10, 10, 10]	[0, 2, 2]	–
PTU 1	[0, 0, 0]	[0, 0, 0]	[0, 0]

Table 4.11: Positional and rotational (RPY) offsets applied to the initial parameter estimates, $\hat{\theta}_0$, for Camera 1. This offset configuration is used as the larger offset case for gradient descent and as the smaller offset case for SciPy’s least squares optimizer.

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
Camera 0	[0, 0, 0]	[0, 0, 0]	–
PTU 0	[0, 0, 0]	[0, 0, 0]	[0, 0]
Camera 1	[100, 100, 100]	[0, 5, 5]	–
PTU 1	[0, 0, 0]	[0, 0, 0]	[0, 0]

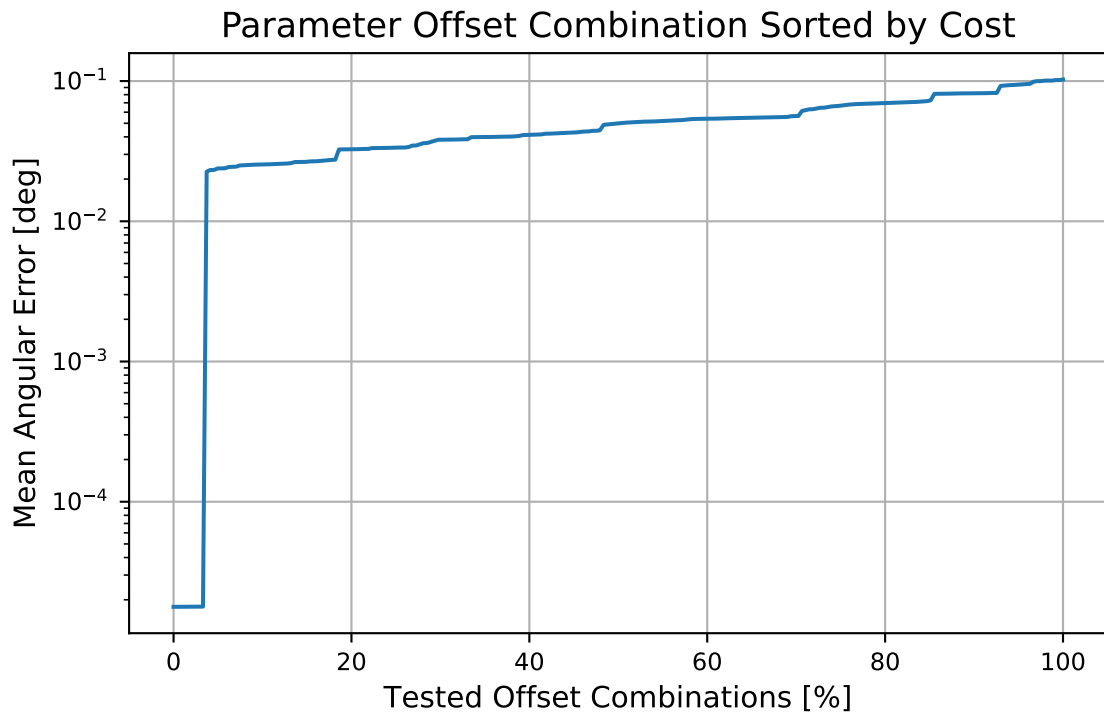


Figure 4.16: Mean angular error after calibration using gradient descent for all combinations of initial offsets specified in Table 4.10, which include 10 mm positional offsets, and 2° pitch and yaw offsets applied to Camera 1. The percentages of the total number of solutions that result in mean angular errors below 0.1° and 1° are reported in Table 4.12 under *Gradient descent, pos:[10,10,10], rot:[0,2,2]*.

Table 4.12: Percentages of mean angular error below 1° and 0.1° for the different initial offset cases for Camera 1 specified in Tables 4.10, 4.11 and 4.13.

Initial offset case	< 1°	< 0.1°
Gradient descent, pos:[10,10,10], rot:[0,2,2]	100 %	97.5 %
Gradient descent, pos:[100,100,100], rot:[0,5,5]	64.2 %	7.4 %
SciPy optimizer, pos:[100,100,100], rot:[0,5,5]	100 %	99.2 %
SciPy optimizer, pos:[150,150,150], rot:[0,10,10]	91.8 %	86.8 %

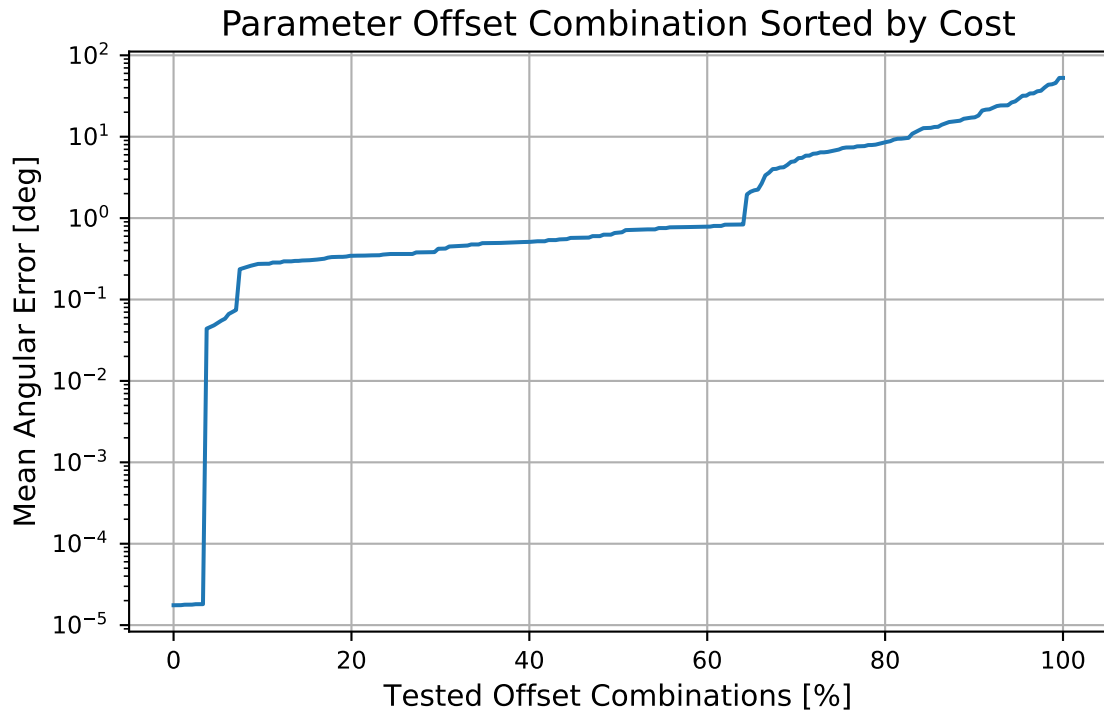


Figure 4.17: Mean angular error after calibration using gradient descent for all combinations of initial offsets specified in Table 4.11, which include 100 mm positional offsets, and 5° pitch and yaw offsets applied to Camera 1. The percentages of the total number of solutions that result in mean angular errors below 0.1° and 1° are reported in Table 4.12 under *Gradient descent, pos:[100,100,100], rot:[0,5,5]*.

4.4.2 Sensitivity for SciPy’s Least Squares Optimizer

Like the evaluation of initial parameter sensitivity using gradient descent, the sensitivity is also evaluated using SciPy’s least squares optimizer for two different initial offset cases: one with smaller offsets and one with larger offsets. The case with the smaller offsets for this optimizer is the same as the larger offsets for the gradient descent optimizer, and can be found in Table 4.11. The larger offset values for the SciPy case are presented in detail in Table 4.13.

The resulting mean angular error is calculated for all combinations of initial offsets, just like in Section 4.4.1, for the two offset cases separately. The results are sorted by error magnitude and are presented in Figures 4.18 and 4.19 for the smaller and the larger perturbation cases, respectively.

The case with smaller offset values (Figure 4.18), results in a mean angular error of less than 0.1° for 99.2% of the initial offset combinations tested, which can be compared to 7.4% in the same offset case for gradient descent. The larger offset values (Figure 4.19) achieve errors below 1° for 91.8% of the offset combinations, and below 0.1° for 86.8% of the combinations. These error distributions are presented, together with the corresponding errors for gradient descent, in Table 4.12.

Table 4.13: Positional and rotational (RPY) offsets applied to the initial parameter estimates, $\hat{\theta}_0$, for Camera 1. This offset configuration is used in evaluation of SciPy’s least squares optimizer with larger offset values than those specified in Table 4.11.

Component	$[x, y, z]$ [mm]	RPY [deg]	$[r_1, r_2]$ [mm]
Camera 0	[0, 0, 0]	[0, 0, 0]	–
PTU 0	[0, 0, 0]	[0, 0, 0]	[0, 0]
Camera 1	[150, 150, 150]	[0, 10, 10]	–
PTU 1	[0, 0, 0]	[0, 0, 0]	[0, 0]

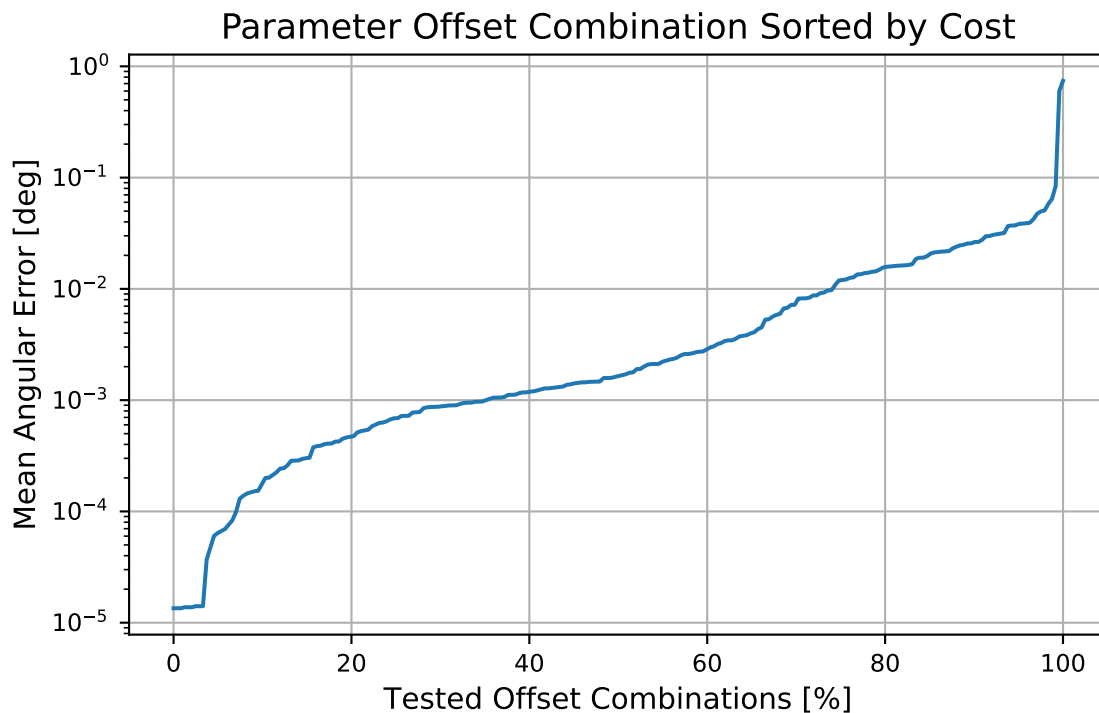


Figure 4.18: Mean angular error after calibration using SciPy’s least squares optimizer for all combinations of initial offsets specified in Table 4.11, which include 100 mm positional offsets, and 5° pitch and yaw offsets applied to Camera 1. The percentages of the total number of solutions that result in mean angular errors below 0.1° and 1° are reported in Table 4.12 under *SciPy optimizer, pos:[100,100,100], rot:[0,5,5]*.

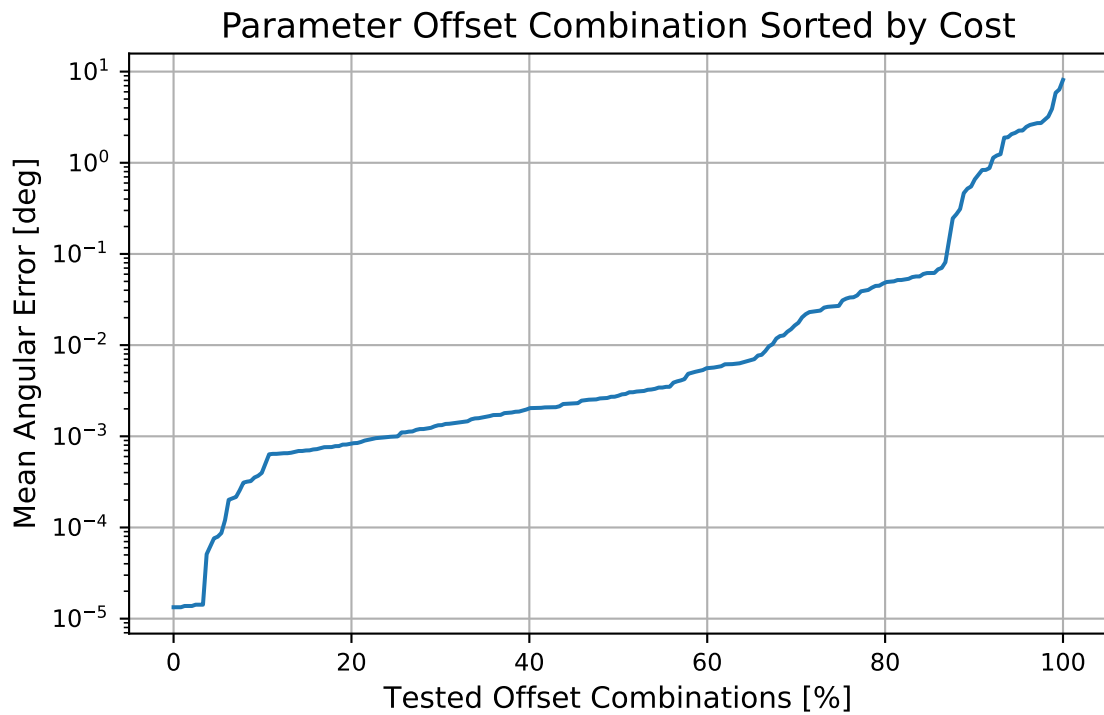


Figure 4.19: Mean angular error after calibration using SciPy’s least squares optimizer for all combinations of initial offsets specified in Table 4.13, which include 150 mm positional offsets, and 10° pitch and yaw offsets applied to Camera 1. The percentages of the total number of solutions that result in mean angular errors below 0.1° and 1° are reported in Table 4.12 under *SciPy optimizer, pos:[150,150,150], rot:[0,10,10]*.

5

Discussion

This chapter discusses the results presented in Chapter 4 and proposes improvements and potential directions for further development of the method.

5.1 Effect of Calibration Point Quantity

The solution accuracy as function of number of calibration points, N , appears to converge when using the manually implemented gradient descent algorithm to minimize the cost function. In contrast, the solution accuracy achieved with SciPy's least squares optimizer continues to improve as N increases over the tested range. However, even though the gradient descent results seem to plateau, the solutions found by the SciPy optimizer results in significantly lower mean angular errors for the same offset scenario. This indicates that the SciPy optimizer produce estimated parameter solutions that better correspond to the true system than the gradient descent algorithm does.

One possible explanation for why the mean angular error for the gradient descent algorithm converges at a relatively high error (approximately 0.2°) could be the parameter step sizes. In this case, the step sizes remain constant throughout the minimization process, and if those step sizes are too large, the algorithm might overstep narrow minima, causing the algorithm to miss regions with lower errors. This issue could potentially be addressed by reducing the step sizes, although this would increase the time required for convergence. An alternative solution would be to implement adaptive step sizes, which may allow the algorithm to locate minima with lower errors more efficiently, without necessarily increasing computation time.

In Section 3.4.1, it is suggested that the amount of calibration points theoretically needed to determine the parameters should correspond to the number of degrees of freedom of the system, which are 18 in this case. This reasoning does not align with the results for the SciPy optimizer shown in Figures 4.2 and 4.3, where additional points beyond the first 18 continue to reduce the cost. A possible explanation is that the optimizer has not found the perfect solution for the parameters, which would have an expected mean angular error of zero. Instead, it gets stuck in local minima. However, the optimizer continues to benefit from the additional data provided by more calibration points, which help it find better, but still local, minima.

For the SciPy least squares optimizer, the effect of number of calibration points was evaluated on two separate sets of offsets to the initial parameters. When the SciPy optimizer is applied to an offset case with larger perturbations, it exhibits a similar behavior in mean angular error reduction with increasing N as in the case with smaller offsets. However, the resulting errors are approximately one order of magnitude larger than those observed for the smaller offset case. While the specific mean angular error values provide some insight, they are specific to the particular simulated calibration and test setup described in Section 3.12 and should not be interpreted as universally representative. The primary contribution of this analysis is the overall trend on how the number of calibration points influences solution accuracy for the two optimization methods, rather than the precise numerical values.

Figures 4.1, 4.2, and 4.3 which show the mean angular error dependence on number of calibration points all experience some error variations for most numbers of calibration points. The result from gradient descent shows that after the median error seems to converge, there still exist sample outliers. If one would evaluate the mean error of the samples for each N instead of the median, these outliers would have a significant impact on the final average. However, if more than five samples would have been evaluated for each N , the effect of the outliers might decrease. The results from SciPy's least squares optimizer likewise show some outlying results, but apart from them seems to have a generally larger variation in mean angular error than the gradient descent errors. However, as mentioned previously, the SciPy optimizer reaches lower errors, even when taking the variations into account. The outlying results observed for both optimization methods propose that the solution occasionally results in unnecessarily large errors. In such cases, rerunning the calibration with slightly adjusted values in one or more initial parameter estimates, θ_0 , may lead to more accurate results.

To summarize, the result shown in Figure 4.1 suggests that the gradient descent requires approximately five or more calibration points before the median of the mean angular error converge for a set of parameter offsets that could be realistic in the real-world setup: one centimeter in the positional parameters and 2° in the rotational parameters. One possible reason for this convergence at a relatively high error can be disadvantageous parameter step sizes, which may cause the algorithm to overshoot better solutions. The error could possibly be reduced by tuning the step sizes or implementing adaptive step sizes. In contrast, Figures 4.2 and 4.3 show that the result from SciPy's least squares optimizer does not exhibit the same converging behavior for the tested range of N as the gradient descent algorithm, as discussed above.

5.2 Use Case Testing in Simulated Environment

Below, the results for gradient descent and SciPy's least squares optimizer in the simulated use case will be discussed.

5.2.1 Simulated Use Case: Gradient Descent

For gradient descent, the mean error for the 10 m ranges in the standard calibration case is similar to, but higher than, the values observed in previous tests. This can be reasonable since the setup (point positions, θ_{true} and θ_0) are the same but the increased error could be explained by the error being evaluated for a different region.

Compared to the previously used standard validation system, described in Table 3.3 where points are distributed $\pm 50^\circ$ horizontally (and from 5° to 35° vertically), the pan angles for ϕ_{PTU_0} now varies between $\pm 90^\circ$. The exact mapping between the mirror angle ϕ_{PTU_0} , and which direction the virtual camera is pointed in, is dependent on the (estimated) camera and PTU pose, but will for this system be approximately $\pm 90^\circ$ horizontally (and approximately from 0° to 30° vertically). From the polar plots (e.g., Figure 4.4) it can be seen that the error is generally higher for angles closer to $\pm 90^\circ$ horizontally which is now included and can increase the calculated mean error compared to previous tests.

Another potential reason is that other radiuses are tested. In Figure 4.4, a decreased error can be seen for $r_{\text{meas}} \approx 2$ m which is close to both the distance of the standard calibration and verification points. This could suggest that the $\hat{\theta}$ found in this case is a local solution that works well for $r_{\text{meas}} \approx 2$ m but does not correctly predict for other ranges. This theory would agree with the further increased observed error for the 100 m measurement cases.

When looking at the three additional calibration cases, very high mean errors are observed. In Figure 4.8 the error for one of these calibration cases is shown. In this figure, a high error ($\gg 10^\circ$) is observed for most sampled positions except for a few. The reason for this could be a combination of gradient descent being unable to find a solution for these point placements, and the given offset for the starting parameter guess. This could be further supported by the additional information that the cost $C(\hat{\theta}|\phi_{\text{cal}})$ for these cases was considerably higher ($\approx 10^3$) which means that the found solution underperformed already during the parameter estimation.

5.2.2 Simulated Use Case: SciPy’s Least Squares Optimizer

For SciPy’s least squares optimizer, lower overall errors are seen for the different cases in Table 4.7, with many of the values below 1° . Compared to the gradient descent result for the 10 m cases, considerably lower errors are observed, even though a larger θ_{offset} has been used (Table 4.2).

Something of interest is that the large increase in errors for the 100 m cases is now not observed, even if this test measures regions not covered by the calibration points. This would be the expected result for if $\hat{\theta} \approx \theta_{\text{true}}$ and highlighting an advantage of having a (correct) model: if the correct parameters are determined, even from a small set of calibration measurements, the system’s behavior can be predicted for all inputs. However, it is important to note that observing this with simulated calibration data does not guarantee the model’s ability to predict real-

world behavior. This is because any real-world effects not captured by the model cannot be detected when the same model is used both to generate data, estimate parameters and make predictions.

An empirical observation is that the errors observed over the pan tilt range for 10 and 100 m (Figures 4.11 and 4.12) show quite similar patterns. This contrasts the behavior for the $\hat{\theta}$ found by gradient descent (Figures 4.6 and 4.7), where different patterns for low and high error areas can be seen for the different distances.

As for the additional calibration cases, calibration on points at 100 ± 25 m radius again resulted in higher mean errors (between 0.3° and 6.44°), but still significantly less than what was observed with gradient descent (between 32.3° and 42.8°). For the 100 m cases, the increase in error is small compared to the 10 m cases. A takeaway from this can be that calibration at long distance works if tests are to be carried out at long distance, while calibration at close range seems to be effective at both. One possible reason for this could be that some model parameters do not have a large effect at long distances. This means that an error in the parameter would not greatly affect the cost during parameter estimation and can be more difficult to estimate, which could lead to the optimization terminating with this error still present. An example of this could be parallax, where at shorter distances the angle to an object can be greatly changed due to displacement, meaning that the object positions would matter less at long distances. Further analysis and tests would need to be carried out to learn more about this behavior.

When comparing the two 10 ± 0.25 m calibration cases the main difference is an increase in error for the pan tilt tests when the tilt range is limited to $[0^\circ, 1.5^\circ]$. The decreased variation in the tilt axis for the calibration set could be thought of as providing the parameter estimation less information about this axis, which could increase the error. The fact that the pan radius errors are largely the same, even though more points are concentrated close to this measurement range, could be an example of the diminishing returns observed in Section 4.1.

In summary, from the results presented, it seems that a wide coverage in pan and tilt, as well as calibrating on points close to the sensor rigs is advantageous. In this test this results in high accuracy for predicted system behavior for regions both close to, but also far from the calibration points, even with a large starting offset for θ_0 . However, further tests with different offsets and system setups are needed to determine whether this is a general behavior of the method.

5.3 Use Case Testing with Real-World System

The results presented in Section 4.3, specifically the low mean approximate angular error and prediction error $\hat{\phi}_{PTU_1} - \phi_{PTU_1}$ in Figure 4.14, generally indicate that the model and found parameters $\hat{\theta}$ seem to capture the systems real-world behavior.

Looking closer at $\hat{\phi}_{\text{PTU}_1} - \phi_{\text{PTU}_1}$ for the calibrated system, it can be seen that the errors are generally larger in pan, especially for the points in the left and right groupings. A possible contributing factor to this can be these points' positions compared to the calibration points, similar to the reasoning in Section 5.2.2 regarding calibration at long or close-range. For the left and right groupings, the required pan angles to observe the points are larger than the angles covered by the calibration measurements. If some parameters have a larger effect for large pan angles, the accuracy of their estimation, and subsequently prediction for large pan angles could be negatively affected. In any case, it should be noted that the prediction errors are within $\pm 0.3^\circ$, which is similar to the expected measurement accuracy of the PTUs and the visual alignment. Therefore, the prediction accuracy for this test is considered satisfactory.

When looking at the parameters, the differences (Table 4.9) between the manually measured parameters and estimated $\hat{\theta}$ are generally low. The values are generally of a similar order of magnitude to what would be considered reasonable measurement errors for the manual measurement performed.

Outliers from the low difference are roll, yaw and z positions of cameras, where large deviations are observed. This can be explained by all of these parameters closely aligning with the cameras optical axes in the real-world setup. As discussed in Section 3.4, rotation and translation along the optical axis has no effect for measurements. This means that these do not impact the cost or predicted error for the system, but also means that these parameter cannot be estimated with the presented method.

As an additional result, the prediction error for the manually measured parameters θ_0 , without any calibration, was also presented in Figure 4.15. These errors were in general larger than for the calibrated system, but less than 1° . This indicated that, depending on the required accuracy, the model can be used for predictions without calibration, but also that the parameter estimation improves the result.

5.4 Sensitivity to Initial Parameter Estimates

The results presented in Section 4.4 show that the SciPy least squares optimizer demonstrates a significantly greater tolerance to variation in initial parameter offsets, as shown in Table 4.12. This supports the conclusion that the SciPy optimizer offers more reliable parameter estimation compared to the manually implemented gradient descent approach.

In these experiments, only the sensitivity to initial offsets in the position and orientation of Camera 1 was evaluated. As a result, it is not possible to draw definite conclusions regarding the tolerance of the estimation method to initial offsets across all system parameters. It is reasonable to speculate that the estimation process would exhibit greater sensitivity to perturbations in the parameters of Camera 1 if initial offsets were simultaneously introduced in other system components. Ex-

tending the current analysis to include all possible combinations of initial parameter offsets, using the same approach as the current, would result in approximately 10^9 initial offset configurations to use as starting points for the parameter estimation. This would be unreasonably computationally heavy, which is why exploring the sensitivity across the full parameter space would require an alternative, more efficient methodology.

An alternative approach to evaluate the method’s tolerance to errors in the initial parameter guess could involve applying offsets to all parameters and then sampling a subset of the offset combinations at random. Calibration would then be performed on that subset. This approach resembles the Monte Carlo method [14], enabling a broader analysis of the parameter space and providing a more accurate assessment of the sensitivity to initial parameter estimates.

5.5 Future Work

This section outlines potential directions for further developing the proposed calibration method. Exploring these areas may help overcome current limitations and increase the method’s applicability to more complex sensor rig systems.

5.5.1 Expand to Systems with More Sensor Rigs

So far, the calibration has been performed using two sensor rigs, but the method may also be applicable to systems with more than two rigs. With the current method, it should be possible to calibrate additional rigs pairwise relative to a specific PTU base. A key advantage of this approach is that it would not require any modifications to the existing method, one simply performs the calibration process an additional time for each new sensor rig that is added to the system. However, a disadvantage of this approach is that information available from sensors other than in the current pair is not utilized during calibration, which may result in suboptimal use of the available data. Major differences involved in developing a method where all sensor rigs are calibrated simultaneously would include a larger set of parameters to estimate, access to more data per calibration point, and the need for a different cost function. This new cost function would need to describe how close multiple optical axes are to intersect, rather than just the closest distance between two axes as in the current cost function. There are likely several ways to define such a function, and the chosen formulation is expected to influence the calibration outcome.

5.5.2 Supplementary Measurement Data

An aspect that might be worth considering is what type of data is collected during calibration measurements. In the current method, each measurement consists of a pair of pan and tilt angles for each mirror, such that the sensors’ optical axes intersect at the same point in 3D space. By collecting additional information per measurement, the space of possible solutions may be reduced. Examples of such data could include distances from the sensors to the calibration point, or the distance

between two sensor rigs. A major challenge in this thesis project is the high number of parameters to estimate. By reducing the number of unknown parameters, the calibration might become easier to solve and the results more accurate. However, the fact that the current method does not require any precise manual measurements of parameters is considered an advantage.

5.5.3 Expand to Complete Camera Calibration

The current method only considers the direction of the optical axis, and does not utilize information from the entire camera image. With this approach, rotation around and translation along the optical axis does not influence the outcome. Extending the method to utilize the full image would allow the measurement point to deviate from the optical axis, but would require knowledge about the intrinsic parameters of the camera. These parameters are used to project 3D point coordinates onto the camera's 2D image plane, and include parameters such as the camera's focal length and optical center. This approach would increase the complexity of the problem, but should be a possible extension to the current method.

When a point is observed at an arbitrary pixel location in the image, its offset from the image center can be converted into an angle using, for example, the pinhole camera model [5]. Once this angle is known, the current model developed in this thesis can still be applied, since the angle offset effectively corresponds to a rotation from the optical axis. Established methods for intrinsic camera calibration can be used to obtain the required parameters. Such methods are available in, for example, the OpenCV library (Open Source Computer Vision Library) [15].

5.5.4 Parameterization

As mentioned in Section 3.5, the chosen parametrization results in more parameters than the number of DOF that affect the measurements, which makes the system over-parameterized. One reason for this is that the parametrization of the camera orientations has three DOF, while only two of them affect measurements. The one that does not impact the result is rotation around the optical axis, as mentioned earlier in this report. Similar to this, camera positions are parametrized using three DOF for the position, even though the translation along the optical axis does not affect measurements.

This discrepancy between the number of DOF that affect measurements and those present in the parametrization can be avoided either by changing the parametrization (reducing the number of DOF), or changing how measurements are performed (which would utilize a greater number of the now existing DOF). An example of the latter is what is described in Section 5.5.3, which can be a relevant solution if cameras are used as sensors. In a setup where the use case, or sensors used, lead to only on-axis measurements, the parametrization can be changed to reduce the number of DOF.

5.5.5 Evaluate Other Optimization Methods

Another potential area of development is to explore other optimization methods, in addition to those that have been tested so far. The optimization methods that have been investigated in this thesis (gradient descent and SciPy's least squares optimizer) are both local optimizers. Since the cost function defined in for this calibration problem contains several local minima, these methods require an initial parameter guess that is quite close to the true parameter values in order for the parameter estimation to find a solution that is close to the true one. By investigating other optimization methods, for example global optimizers, the sensitivity to initial parameter guesses might be reduced.

6

Conclusion

The proposed calibration method, which includes both a model of the system and estimation of its parameters, has shown promising results in both simulated and real-world scenarios. The results indicate that the estimated parameters, when used with the model, can predict the system's behavior with low errors. For example, in a close-range real-world setup, the difference between predicted and measured PTU mirror angles is within $\pm 0.3^\circ$, which is approximately the same as the measurement accuracy of the setup.

A key limitation of the current method is its reliance on an initial parameter guess that is relatively close to the true values in order to achieve accurate parameter estimation. This sensitivity arises from the method used to minimize the cost function, where the optimization algorithm can get trapped in local minima. Alternative algorithms might help mitigate this limitation. However, if an approximate initial guess of the parameters is possible, within a few centimeters or degrees of the true component positions and orientations, the current method appears to perform well.

Bibliography

- [1] S. Hu, Y. Matsumoto, T. Takaki, and I. Ishii, “Monocular stereo measurement using high-speed catadioptric tracking,” *Sensors*, vol. 17, no. 8, p. 1839, Aug. 2017.
- [2] S. M. Kay, *Fundamentals of Signal Processing: Estimation Theory*. vol. 1, Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.
- [3] C. Forssén, “Learning from data,” 2023. [Online]. Available: <https://gitlab.com/cforssen/learningfromdata>, (accessed on: 2025-04-29).
- [4] S. Farrell and S. Lewandowsky, “Basic parameter estimation techniques,” in *Computational Modeling of Cognition and Behavior*. Cambridge University Press, 2018, pp. 47 – 71.
- [5] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2003.
- [6] K. Holmberg, *Optimering: Metoder, modeller och teori för linjära, olinjära och kombinatoriska problem*, 2nd ed. Stockholm, Sweden: Liber AB, 2018.
- [7] I. Gustafsson and K. Holmåker, *Numerisk analys*, 1st ed. Stockholm, Sweden: Liber AB, 2016.
- [8] N. Buduma, N. Buduma, and J. Papa, *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms*, 2nd ed. Sebastopol, CA, USA: O’Reilly Media Inc., 2022.
- [9] J. Schmidt and H. Niemann, “Using quaternions for parametrizing 3-d rotations in unconstrained nonlinear optimization,” 01 2001.
- [10] The SciPy Community. (2025) `scipy.spatial.transform.Rotation` — SciPy v1.11.4 Manual. [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.html>
- [11] ——. (2025) `scipy.optimize.least_squares`. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html

- [12] FLIR Systems, Inc. (2017) FLIR PTU-5 Datasheet. Document No. 17-0891-OEM, Version V2. [Online]. Available: <https://flir.netx.net/file/asset/5650/original/attachment>
- [13] Teledyne FLIR LLC. (2021) FLIR Blackfly S Datasheet. Document No. 21-0687-OEM-BFS-PGE-16S2-LTR, Revision 05/11/21. [Online]. Available: <https://flir.netx.net/file/asset/17228/original>
- [14] G. S. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, 1st ed. Chapel Hill, NC, USA: Springer New York, NY, 1996.
- [15] Open Source Computer Vision Library, “Camera calibration,” 2025. [Online]. Available: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html, (accessed on: 2025-06-15).

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY