Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Martin Græsdal

Self-Calibration of Stereo Vision for Autonomous Ferry

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke
Co-supervisor: Annette Stahl, Øystein K. Helgesen
May 2021

NTNU
Kunnskap for en bedre verden

Martin Græsdal

Self-Calibration of Stereo Vision for Autonomous Ferry

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke
Co-supervisor: Annette Stahl, Øystein K. Helgesen
May 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis explores the possibilities of implementing a self-calibration algorithm for a stereo vision system to be used on an autonomous ferry. The algorithm aims to replace traditional offline calibration methods which are dependent on a calibration rig of known dimensions to function, by being able to perform the calibration on data gather by the cameras during normal operation. An auto-calibrating algorithm would lessen the need for maintenance of the stereo system.

A calibration method is discussed, implemented and evaluated in this report. This method tries to estimate the calibration parameters by minimizing the reprojection error of feature matches between three images. Epipolar and trilinear constraints are introduced to the problem to guide the optimizer towards the solution. An Extended Kalman Filter is used as optimiser.

The trilinear constraints needs three-view matches to be computed. A three-point matcher is proposed. This matcher finds common matches between the two stereo images and one of the images in the next stereo image pair. Different types of feature methods are discussed throughout the report. SIFT, SURF and ORB are evaluated and tested as descriptors. FLANN and brute force are considered as matching methods.

The auto-calibrating algorithm was tested on real data captured in the Trondheim channel during the project period. Results for testing revealed that the algorithm did not manage to reproduce the calibration done using traditional methods, and some of the weakness using EKF as an optimizer on non-linear problems was discussed.

Sammendrag

Denne rapporten ser på muligheten til å implementere en selvkalibrerende algoritme for et stereo vision system som skal tas i bruk på en autonom ferje. Målet med algoritmen er å erstatte tradisjonelle offline kalibereringsmetoder, som er avhengig av en kalibreringsrig av en kjent størrelse for å fungere, ved å være i stand til å kalibrere stereo kameraene kun ved hjelp av data som blir samlet av kameraene under normal drift. En selvkalibrerende algoritme vil gjøre at stereo systemet trenge mindre vedlikehold.

En kalibreringsmetode blir diskutert, implementeret og vurdert i denne rapporten. Denne metoden prøver å estimere kaliberingsparametrene ved å minimere reprojeksjonsfeilen som oppstår mellom matchende features i tre bilder. To typer constraints er lagt til optimaliseringsproblemet for å hjelpe optimeringsmetoden med å finne riktige parametre, nemlig epipolar og trelinear constraints. Et Extended Kalman Filter er brukt for å optimalisere problemet.

Den trelineare constrainten trenger feature matcher fra tre forskjellige bilder. En trepunktsmatcher er foreslått i denne rapporten. Den finner matcher mellom bildene fra stereoparet og et av bildene fra det neste stereoparet som blir prosessert. Forskjellige featuremetoder are diskutert i rapporten. SIFT, SURF og ORB er vurdert som feature descriptors. FLANN og "brute force matcher" er vurdert som machingmetode.

Autokalibreringsalgoritmen ble testet på ekte data som ble samlet inn i Trondheimskanalen i løpet av prosjektperioden. Resultatene fra testing av algoritmen viste at algoritmen ikke klarte å gjenskape de samme parameterene som en tradisjonell kalibreringsmetode produserte. Noen av svakhetene ved bruk EKF som en optimaliserer blir diskutert.

Preface

This thesis concludes my 2-year Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). I want to thank my main supervisor Edmund Førland Brekke for his invaluable support and council throughout the project period. I would also like to thank Annette Stahl and Øystein Kaarstad Helgesen for being co-supervisors. Their input and help were very appreciated. My main motivation for working with a stereo vision system was the course TTK4255 - Robotic Vision which I took in the spring of 2020, and where Annette Stahl was the lecturer. The unleashed potential that lays in cameras as sensor is intriguing, and to take part in the development in the field of computer vision has been very exciting.

The projection is written as part of the research project Autoferry at NTNU [28]. One week during the course of the project was set aside for gathering data sets for testing using the ferry milliAmpere. The data gathering was done in collaboration Thomas Hellum, Martin Gerhardsen and Kristian Auestad. The collaboration was prosperous, even though everyone needed different data for their projects. I would like to thank Egil Eide for trusting us in lending milliAmpere. Also, thank you to Glenn Angell and Terje Hauge at the workshop of ITK for building the weather casings for the cameras and for lending us a car during the experiment week.

Lastly, I would give a big thank you to my co-student Kristian Auestad. We have been working on the same stereo system. He has been a solid sparring partner throughout the project.

Martin Græsdal

Trondheim, 31.05.2021

Contents

Chapter 1

Introduction

Autoferry is a project at NTNU which are developing autonomous passenger ferries which can navigate in urban waters. The prototype ferry milliAmpere has been available for testing for several years already, and now the second ferry milliAmpere2 is soon to be launched [28]. For any autonomous vehicle, situational awareness is extremely important. When there is no operator to observe the environment and make navigational decisions based what he sees, a computer must do that work instead. milliAmpere uses a lot of different sensor to detect dangers in its surroundings already, but a stereo vision system has not yet been implemented on the ferry.

Computer vision is the art of transforming an environment into digital data that a computer can interpret using cameras as sensors. With a stereo camera system an accurate depth estimation is possible since the system captures two images of the same since simultaneously. By knowing the configuration of the two cameras, it is possible to estimate the 3D position of a viewed point in relation to the cameras.

The accuracy of the stereo system is heavily dependent on the calibration of the cameras. There are at least six intrinsic parameters per camera and six extrinsic parameters between each camera to be determined to create a good camera model which can transform 2D pixel points into 3D world points. Estimation of these parameters are done through calibration. A lot of the calibration methods today are based on observing a calibration rig of known dimension such as a checkerboard. These methods require the stereo camera system, and effectively the vehicle utilizing the system, to be put out of service whenever a re-calibration is needed. In order to make the ferry more independent, a method of performing an online calibration without the need of a calibration rig is desired.

This report aims at developing an auto-calibration algorithm for a stereo vision system. It is a

continuation of the research done by the author in his final year specialization project in 2020 [16]. The stereo system has been developed by L. Theimann and T. Olsen during their master project [44]. Since the system is to be implemented on milliAmpere, the algorithm needs to compatible with the current ROS-system implemented on the ferry. Different types of feature methods need to be evaluated in order find the most fitting ones for this application. This thesis also discusses the possibilities of performing the calibration on the data gather from the stereo system during normal operation.

1.1 Report outline

Chapter 2 introduces the theory that the calibration algorithm is based on. In chapter 3 the software and hardware of the stereo system is presented. A week of the project period was spent at sea, gathering data set for the algorithm to be tested on. The data gathering is described in chapter 4. Details of the auto-calibrator is elaborated in chapter 5, including a description of image processing and implementation choices made for the algorithm. Chapter 6 presents the results of the implementation followed by a discussion about the findings in chapter 7. The thesis is concluded in chapter 8.

Chapter 2

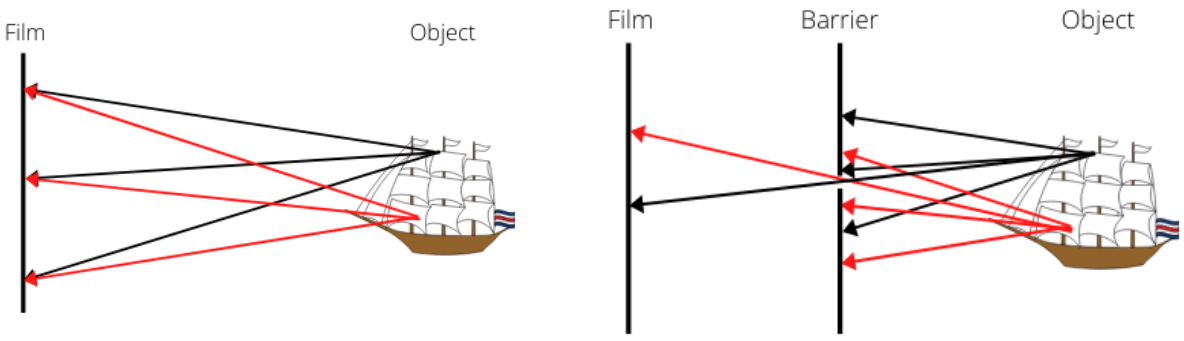
Theory

This chapter presents the most important theoretical topics for the rest of the report. Since this thesis is a continuation of the authors specialization projection, some of the sections in this chapter is based on theory chapter in the project report [16]. Especially section 2.1 and section 2.2 is largely based on the project.

2.1 Camera Model

A camera model is used to translate 3D points in a scene to corresponding 2D points in the image plane. The pinhole camera model is based on the simple principle of the pinhole camera. Such a camera is defined as a closed box with a tiny hole in which light is let through. The light hits a photosensitive surface, often referred to as a film, in which the image is captured. An object in the real world will reflect light in every direction. The small hole filters the light, making sure that the light emitted from a point in the scene will only enter the box from one direction [42].

The image plane is defined on the photosensitive surface. Because of the directed light, the observed scene will appear inverted in the image plane. In order to reduce exposure time and still have a focused image, real cameras permits light rays through a wider opening than a pinhole, utilizing a series of lenses to focus the light through a single point called the centre of projection, which acts as the pinhole in the model. Distance from the centre of projection to the image plan is called the focal length f. The orthogonal line from the image plane which passes through the projection centre is the optical axis, and the point where this line originates on the image plane is the principal point. To simplify visualization of the image plane and its geometry, it is com-



(a) Light beams from the scene. (b) Filtering by the pinhole.

Figure 2.1: Principle of pinhole camera.
Illustrations courtesy of Kristian Auestad.

mon practice to create a virtual image plane in front of the camera. That way their image will no longer appear inverted, and there is no need to rotate the image. The focal length is used to determine where this plane is to be placed. Hence forward, when referring to the image plane, it is the virtual image plan in front of the camera that is discussed.

Origin of the *image coordinate system* is at the principal point with the Z-axis coinciding with the optical axis, pointing towards the scene. X-axis is parallel with the horizontal line, and Y is pointing downwards. *The pixel coordinate frame* has it origin in the top left corner of the image. This system is two-dimensional with the X-axis coinciding with the columns, and Y-axis with the rows, of pixels in the image. A *camera coordinate system* is defined with origin in the projection center and follows the same orientation as the image coordinate system. The position and orientation of the camera in the world frame are defined by the camera coordinate frame.

To determine the representation of a 3D point on the image plane, a line from the 3D point to the optical centre can be drawn, and at the point where this line intersects the image plane will be its 2D correspondence. The mathematical translation from 3D point (X, Y, Z) to the image coordinates (x, y) can be expressed as:

$$x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z} \tag{2.1}$$

Representing the 3D point in the pixel coordinate frame, requires considering the translation of origin from camera to pixel coordinates. When converting from 3D to a 2D plane from a single viewpoint the scale of the scene is lost. This is due to a effect called forced perspective. The camera cannot tell if an object is small and 1 meter away, or big and at a 10 meter distance. The

transformation is thus defined up to scale, and a scaling factor must be added. In homogeneous coordinates this can be written as:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.2}$$

Where $(p_x,\ p_y)$ is the principal point. This transformation assumes that the pixels are perfect squares. This is not always the case, especially not in digital cameras using charge-coupled devices (CCD) [20]. To compensate for the unequal scaling effect different pixel sizes can create, a factor in x and y direction are multiplied. The factors m_x and m_y are defined as pixel per unit distance. Additionally, a cross term between X and Y called the skew term s are added. Usually this term is zero, but is used in the special case where the image axes are not perpendicular.

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_x f & s & m_x p_x & 0 \\ 0 & m_y f & m_y p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.3}$$

This model represents a 3D point in relation to the camera. If the camera is to move or there are multiple cameras in the system, it is desired to relate the 3D point to the world coordinate frame. In order to fix that, the pose of the camera in relation to the world frame is added to the model [42].

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} m_x f & s & m_x p_x & 0 \\ 0 & m_y f & m_y p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2.4}$$

2.1.1 Model Parameters

The pinhole model gives rise to a lot of parameters to be determined in order for the model to be a valid approximation of the true camera. These parameters are often divided into intrinsics and extrinsics.

Intrinsic Parameters

Intrinsics are the parameters that describes the inside of the camera. They are collected in the calibration matrix:

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & x_o \\ 0 & f_y & y_o \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Where} \quad \begin{matrix} f_x = m_x f \\ f_y = m_y f \\ x_o = m_x p_x \\ y_o = m_y p_y \end{matrix} \tag{2.5}$$

Since the skew parameter s commonly is zero, each camera usually have four intrinsic parameters to be estimated.

Extrinsic Parameters

The extrinsics are the relative position of the camera. It contains the rotation and translation in relation to a given coordinate frame. If the system contains multiple cameras, every camera can refer to a common origin in world. In cases where there are no natural point to relate, the position of one of the camera can be set as origin, and all other determine its relative position from the reference camera.

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R}_R & \boldsymbol{t}_r \\ \boldsymbol{0}_{1x3} & 1 \end{bmatrix} \quad \text{where} \quad \begin{matrix} \boldsymbol{R}_r = \boldsymbol{R}_z(\theta)\boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\psi) \\ \boldsymbol{t}_r = \begin{bmatrix} x & y & z \end{bmatrix}^\top \end{matrix} \tag{2.6}$$

The rotational matrix uses Euler angles in the sequence of roll-pitch-yaw [10]. Each rotation has its own angle and the translation contains three values, resulting in 6 extrinsic parameters to be estimated per camera.

Distortion

One of the weaknesses of the pinhole model, is that it assumes a perfectly planar image plane. In most cameras this is not the case because of the introductions of lenses [42]. There are two types of lens distorting effects: Radial and tangential distortion.

Radial distortion leads to straight lines in the image appearing bent. The effect is more apparent

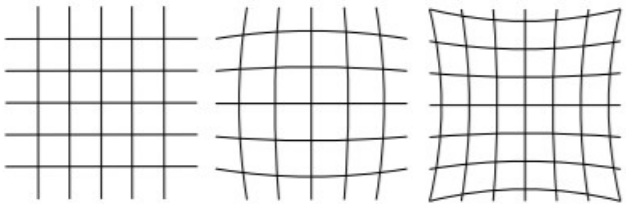


Figure 2.2: Radial distortion [17].

along the edges and in the corners of image. Figure 2.2 illustrates this effect in an image. A non-planar film is the source for this effect. Center of radiation are typically in the principle point. With (x_0, y_0) as the principal point and (x, y) the measured, point correction for radial distortion can be modelled [47]:

$$\hat{x} = x_0 + \bar{x}(1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + ...) \tag{2.7}$$

$$\hat{y} = y_0 + \bar{y}(1 + K_1 r^2 + K_2 r^4 + K_3 r^6 + ...) \tag{2.8}$$

where

$$\bar{x} = (x - x_0), \quad \bar{y} = (y - y_0), \quad r^2 = \bar{x}^2 + \bar{y}^2$$

The distortion is approximated by a Taylor series where $K_1, K_2, K_3, ...$ are coefficients that needs to be estimated. The first terms are often the biggest contributors, and therefore the latter terms are often dropped.

Tangential distortion appears when the image plane and lens are not vertically aligned. The mathematical model of the tangential distortion is [43]

$$\hat{x} = x_0 + p_1(r^2 + 2\bar{x}^2) + 2p_2\bar{x}\bar{y} \tag{2.9}$$

$$\hat{y} = y_0 + 2p_1\bar{x}\bar{y} + p_2(r^2 + 2\bar{y}^2)) \tag{2.10}$$

where

$$\bar{x} = (x - x_0), \quad \bar{y} = (y - y_0), \quad r = \sqrt{\bar{x}^2 + \bar{y}^2}.$$

P_1 and P_2 are the distortion coefficient. Usually the tangential distortion is so small that they are not taken into account in the model.

2.1.2 Calibration

Estimation of the camera parameters are done via calibration algorithms. Calibration methods are divided into two categories: classical methods and self-calibration. Classical calibration methods rely on a calibration rig enabling some information about the scene observed by the cameras. Self-calibrating or auto-calibrating algorithms utilizes prior knowledge about the calibration of the cameras and matching point features to estimate the parameters [20]. While some of the classical methods are well renowned, the self-calibration methods need to be tailored to the specific appliance.

Zhang's Method

Zhang's method is a commonly used calibration method. It is a hybrid between classical and self-calibration. The only requirement for the method to work, is that the cameras observe a planar pattern that are being shifted around in the scene. A checkerboard is often used for this purpose. Either the camera or the pattern can be moved, in order to get different orientations of the pattern. Features are being used to track the pattern. Linear transformation of the pattern is then used to get an initial estimation of the camera parameters. The parameters are refined further by using a Levenberg-Marquardt algorithm to reducing the reprojection error [50].

One of the weaknesses with this method is that the pattern has to cover big parts of the images. The calibration should also be performed at the distance of which the cameras are to observe objects. If the cameras are to operate over long distances, the pattern must be very large. This makes Zhang's method less usable in real world applications.

2.2 Optimization

A lot of the calibration methods boils done to a nonlinear optimization problem. The aim is to find a set of variables that minimizes an object function. When these functions are nonlinear, finding this set is difficult. A common approach to solving these problems are by utilizing iterative methods. In this section some of these methods will be presented.

2.2.1 Steepest Decent

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + a_k \nabla_k \tag{2.11}$$

The steepest decent method is one of the simplest optimization methods, and sets the basis for a lot of other iterating optimization methods. It is a line-search method, which means that the algorithm computes a direction for every iteration in which the function should search for a more optimal solution. For every iteration, the gradient (∇_k) of the function with respect to the parameters is calculated. The parameters are then updated with a new value along the gradient, where the step length a_k determines how much the parameters should be changed[27]. The method has a good convergence rate if the function is simple, but it may struggle a bit more if it becomes more complicated. As a way of improving upon this method different strategies of choosing better search direction is proposed [13].

2.2.2 Gauss-Newton

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(\boldsymbol{x}) \tag{2.12}$$

The Gauss-Newton is a nonlinear least-squares problem method which is an supplement on the Newton method, and allows for an efficient implementation on these. The objective function consists of several residual functions $\boldsymbol{r}$, that should be minimized. In the Newton method the hessian of the system, as well as the jacobian, are calculated in order to choose a search direction. Especially calculating the hessian is computationally heavy if the function is complicated and there are a lot of residuals to consider. The hessian is therefore approximated by the jacobian squared. Equation (2.14) is used to calculate the search direction $\boldsymbol{p}_k$ [27].

$$\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{p}_k = -\nabla f(\boldsymbol{x}_k) \tag{2.13}$$

$$\boldsymbol{J}_k^\top \boldsymbol{J}_k \boldsymbol{p}_k = -\boldsymbol{J}_k^\top \boldsymbol{r}_k \tag{2.14}$$

Since

$$\nabla^2 f(\boldsymbol{x}_k) \approx \boldsymbol{J}_k^\top \boldsymbol{J}_k \tag{2.15}$$

The Gauss-Newton method has a much faster convergence rate than the steepest decent method for moderate sized problems [13].

2.2.3 Levenberg–Marquardt

Levenberg-Marquardt is a modification of the Gauss-Newton. This method can both adjust the search direction and the step length. While utilizing the equation from Gauss-Newton eq. (2.14), a damping factor λ are added to adjust the search [27].

$$(\boldsymbol{J}_k^\top \boldsymbol{J}_k + \lambda \boldsymbol{I})\boldsymbol{p}_k = -\boldsymbol{J}_k^\top \boldsymbol{r}_k \tag{2.16}$$

The damping factor are initialized at a high value. For every iteration, the effect of the step $\boldsymbol{p}_k$ on the state are tested on the residuals. If the step does not lead to a reduction in residuals, the damping factor are increased. But if the step was successful, the state is updated, and the damping factor are decreased. This leads to a flexible optimization. When the damping factor is big it will dominate the hessian approximation term, and the search direction is similar to the steepest decent. With a small damping factor, it is comparable with the Gauss-Newton. That way the Levenberg-Marquardt get the safety of convergence from the steepest decent, and the speed from the Gauss-Newton [13].

2.3 Features

When working with direct methods in computer vision, one of the most important aspects is to have solid features to work with. In this section the most relevant and commonly used feature descriptors are presented.

2.3.1 Harris Corner

The Harris corner detection is a method first presented in 1988 by Chris Harris and Mike Stephens [19]. Their method identifies edges and corners in an image by looking at the intensity. For each pixel in the image a window W are selected. By shifting the window slightly around the pixel, a Sum of Squared Difference (SSD) energy function are created.

$$E_{SSD}(u,v) = \sum_{u,v \in W} \big(I(x+u, y+v) - I(x,y)\big)^2 \tag{2.17}$$

By some algebraic manipulations this function can be approximated using a Taylor series.

$$E_{SSD}(u,v) \approx [u\,v] \underbrace{\begin{bmatrix} \sum_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} u \\ v \end{bmatrix} \tag{2.18}$$

By observing how the energy function changes as the window moves, the structure around the pixel is exposed. If there are no changes to the energy, the pixel is located on a flat intensity structure. If there are changes when shifting in some, but not all, direction the point is on an edge. A corner is detected if the intensity changes when moving in every direction. In order to translate this into a mathematical formula, Harris came up with this response function [19]:

$$R = \det(\boldsymbol{A}) - \kappa \operatorname{trace}^2(\boldsymbol{A}) \tag{2.19}$$

where

$$\det(\boldsymbol{A}) = \alpha\beta, \;\; \operatorname{trace}(\boldsymbol{A}) = \alpha + \beta \tag{2.20}$$

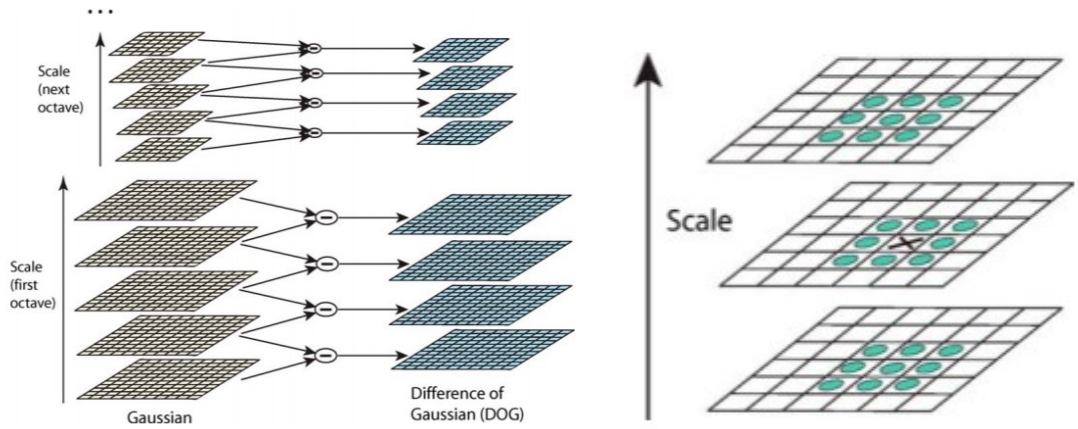
α and β are the eigenvalues of the $\boldsymbol{A}$. κ is scalar value which needs to be chosen, and typically lies between $[0.04, 0.15]$ [18]. A high positive R-value indicates a corner, while high negative values indicate edges. If R is a small number, it is considered a flat area.

The Harris corner method is computationally lightweight and fast. One of the weaknesses with this method is that it is scale dependent, making it unsuitable if the scene is non static. A feature on a object which is to be tracked, might not be detected in the next frame if the object has moved closer or further away from the camera.

2.3.2 Scale Invariant Feature Transform

The Scale Invariant Feature Transform (SIFT) is a feature descriptor that are invariant to scale. It was developed by Davis Lowe in 2004, and it is renowned for its robustness. Scale invariancy is achieved by searching for stable features over all possible scales in the image by utilizing a continuous scale function called scale space [22]. Using SIFT might come at a price though, as the algorithm is know the be quite computationally heavy [39].

The feature extraction is done through four steps. First the algorithm searches over all scales and image locations to find candidates that might be local maximas and minimas, using Difference



(a) Creation of Difference of Gaussian (DoG).

(b) Comparing candidates with neighbors.

Figure 2.3: Feature search in SIFT.
Illustrations courtesy of David Lowe [22].

of Gaussians (DoG). A candidate is detected by comparing it with its eight neighbouring pixels on the same scale and 9 neighbours on the scale above and below. If the candidates' value is lower or higher than its neighbour it is a local extrema. The position of the candidate along with the scale are saved. The first step is executed over several layers of sampling. The image is resampled and smoothed by a factor of 2 for every layer [22].

In the second step the algorithm evaluates all the candidates found in step one and reject unsatisfactory key-points. Candidates that are rejected are those with low contrast or located along edges which can lead to uncertainties. The low contrast rejection is performed by using a Taylor series approximation to the scale space function evaluated at the candidate. A threshold value is selected, and candidates that score below that are rejected. Edges are rejected in similar fashion as in the Harris corner method with the eigenvalues of a hessian matrix.

The next step is to assign an orientation to the key points based on local image gradient direction. Its orientation is determined through a histogram where every sample point around the key point gives a weighted vote for a direction. The direction with the most votes is selected as the orientation for the key point. If there are several directions that are within 80% of the highest peak, duplicate key points are created with same location but with different orientation. This make SIFT invariant to rotation of the images [22].

The final step is to create a solid descriptor of the key points. Using a similar technique as the previous step, the local gradient of the neighbourhood is now considered. The 16x16 neighbouring points are divided into 4x4 sub regions. Every sub region votes in a histogram with 8 bins. The results from these histograms are used as the descriptor. This creates a descriptor

with 4x4x8 = 126 dimensions. The final feature contains pixel location, orientation and the descriptor.

Considering the computational complexity SIFT is not well suited for real time applications. What is gained by SIFT is a robust descriptor which is invariant to scale, rotation and translation. SIFT was previously patented, but are now free for all to use [5].

2.3.3 Speeded Up Robust Features

In 2006 Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool presented a feature detector with lots of the same attributes as the SIFT. They named their detector Speeded Up Robust Features (SURF), as the method outperformed most of the state-of-art descriptors of their time on speed. SURF is invariant to depth and rotation, and it is robust to noise.

SURF uses a box filter to approximate the determinant of Hessian. A way to speed up the filter is to evaluate the integral image instead of the real image. Every point $\boldsymbol{x} = (x, y)$ in the original image gets assign a value of the sum of the intensities of every point in the rectangle above itself, with origin in the top left corner, shown in eq. (2.21). With the image converted to a integral image, the calculation of finding the intensity inside the box filter reduced to simple addition and subtraction.

$$I_{\Sigma}(\mathbf{x}) = \sum_{i=0}^{i \leqslant x} \sum_{j=0}^{j \leqslant y} I(i, j) \tag{2.21}$$

Like SIFT, SURF uses scale space to become invariant to scale. One of the benefits of the box filters is that resampling of the image is no longer necessary. Instead, the different scales can be evaluated by changing the size of the box filter. This saves a lot of time, and it is part of the reason why SURF is faster than SIFT. The interest point is localized using a non maximum suppression in a 3x3x3 neighbourhood [2].

In order to make the SURF descriptor invariant to rotation, a direction is assigned to the features. This is achieved by calculating the Haar wavelet response in x and y direction within a circle of radius of $6s$ around the key point, where s is the scale in which the feature was found [2]. The direction is then found by scanning the circle for the sector with the highest sum of responses. A sector size of $\frac{\pi}{3}$ is chosen. The dominant sector is chosen as the features direction.

To describe the features, a square frame of size $20s$ is selected. The frame is aligned with the feature direction and split into 16 subsections. Within each subsection 5x5 regularly spaced

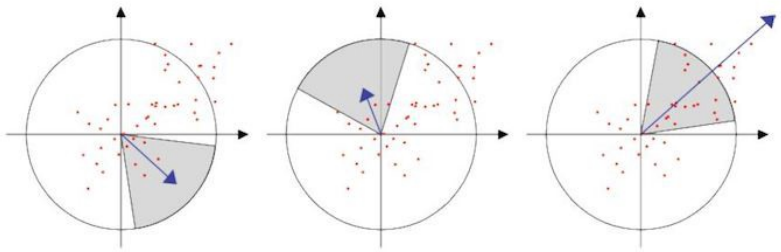


Figure 2.4: Scanning for dominant direction [45].

points are selected and the Haar wavelet response is calculated in x and y direction. The sum of response as well as the sum of absolute values of the responses x and y direction are saved for each subsection. The result is a 64 dimensional descriptor for the feature point [2]

SURF is faster than SIFT, and less susceptible to noise, making it more suitable for online implementations. With the smaller descriptor, a speed advantage when working with the features is gained, but some of precision might be lost. Rublee argues that the feature direction of SURF is poorly approximated [39]. SURF is still protected under a non-commercial license [1].

2.3.4 Orientation FAST Rotation BRIEF

Orientation FAST Rotation BRIEF (ORB) was developed by OpenCV labs to create a solid feature extractor that was faster than SIFT. OpenCV uses open-source code, so their method was never licensed. The method is based on two other descriptors, FAST and BRIEF [39].

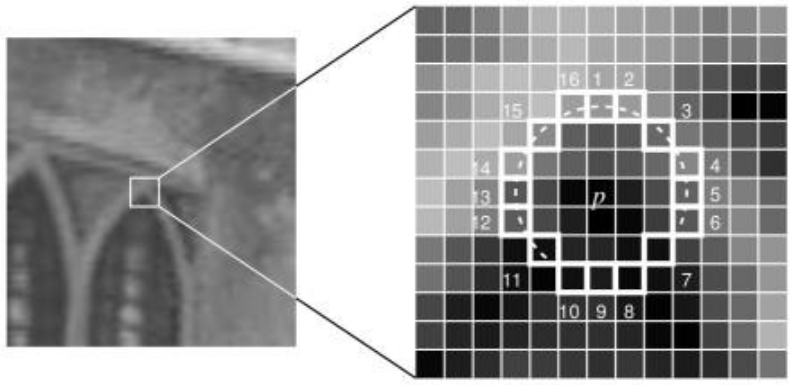


Figure 2.5: FAST key point finder [33].

Key point candidates are found using FAST. A circle is considered around a pixel. For the center pixel to be a corner, a set number of consecutive pixels along the rim of the circle needs to be ether brighter or darker than a threshold value of the center pixel. Every candidate is then

valuated using a Harris corner measure, and only the best corner candidates are saved. Since FAST does not produce multi-scale features, FAST is run on every layer of a scale pyramid of the image. A direction is given to every feature by finding the intensity centroid with the assumption that a corners intensity is offset from its center. The key point is give the direction of the vector from corner center to the intensity centroid [39]. The features are described using an improved version of the BRIEF descriptor. BRIEF describes the smoothed patch of image around a feature by performing a binary test of intensity of the surrounding pixels. The result is a binary vector of 256 elements that describes the area around the key point. The many problem with BRIEF is that its not invariant to rotation of the scene. This is counteracted by steering the patch using the direction found by FAST [39].

The result is robust feature detector which has a lot of the same properties as the SIFT, that requires a lot less computational power [34]. Because of its efficiency and the fact that is has been open for use, makes this descriptor very common in a lot of computer vision systems. The state-of-art visual SLAM method ORB-SLAM is based on this descriptor [6].

2.4 Feature matching

When features are found in one image, it is often desirable to locate the same feature in another image. This is done through feature matching. The most common method for matching is Brute force and FLANN. RANSAC is a method of filter out bad matches.

2.4.1 Brute Force

The Brute-Force matcher (BF) is a simple matcher, where every key point in on image is attempted paired with every key point in the other image. A distance between the key points is calculated, and the closest one is selected as its match. Different distance calculations are used depending on which feature descriptor is being used. For SIFT the L2 norm is the most optimal, and for ORB the hamming distance is favoured [23]. Since every feature gets a match, no matter how bad, it is important to have a threshold of an acceptable distance to discard the worst matches.

BF can be computationally heavy if there are a lot of features in the scene. The reason why this method is used is because there is minimal risk of overlooking any good matches.

2.4.2 FLANN

FLANN is an abbreviation of Fast Library for Approximating Nearest Neighbours. The matcher is a collection of two different algorithms that solves the nearest neighbour problem by applying ether randomized kd-trees or hierarchical k-means trees to search for solutions [24]. The FLANN solver will choose the best algorithm depending on which data set is provided by the user, meaning that the user does not need an in-depth knowledge of how the methods operates. The result is an easy-to-use method that is much faster than BF on big data sets. Distance between features are, like with BF, the quality measurement for the match [24].

The main drawback with working with search trees is the fact that even though a key point gets a match, there it can not guarantee that the best available match is selected. This might result in more outliers or mismatches than BF.

2.4.3 RANSAC

Whichever method is used to match features in two images, there will always occur some mismatches. These mismatches are often referred to as outliers, they may cause inaccuracy for algorithms which is relying on the matches. It is thus desirable to filter out the outliers. One of the most common methods for this operation is called RAndom SAmple Consensus (RANSAC).

RANSAC tries to find the model that best fits a set of data. This is done by selecting a random minimal subset of the data and calculate the model according to this subset. To evaluate the quality of the estimated mode, the data points in the full set which lies within a distance threshold of the model is identified. These data points are referred to as the inliers. If the number of inliers is satisfactory, the model is re-estimated using only the inliers. But if the model didn't result in enough inliers, a new subset of data is selected, and a new model is calculated. These steps could be repeated multiple times, until an acceptable model is found. It is common to set a finite number of trials for the algorithm. If none of the models resulted in the preferred amount of inliers, the model with highest amount of inliers is selects. Its set of inliers are used to re-estimate the model [20].

A simple way of visualizing RANSAC is to consider a set of 2D points, where the goal is to find the line that best represents all of the points. Two random points are selected, which forms a line (solid line in 2.6). All the points are checked to if they lay within the thresholds (dotted lines in 2.6). When points a and b are selected a total of 10 points are inliers, but if c and d are selected no other points falls inside the threshold.

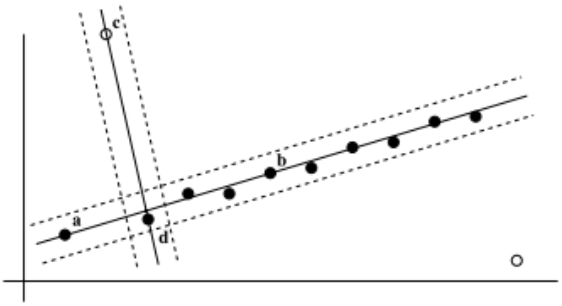


Figure 2.6: Line fitting using RANSAC
Courtesy of Hartley and Zisserman [20].

For feature matching between two images, the simplest model to use is to find the 2D homography which describes the projection transformation taking a point in the one image to its match in the second image. Only 4 point correspondences are needed to find the homography [20]. By using RANSAC to find the homography, the resulting inlier subset will only contain solid matches.

2.5 Reprojection error

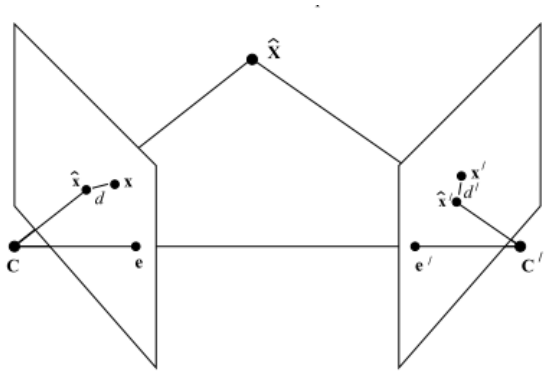


Figure 2.7: Reprojection error
Courtesy of Hartley and Zisserman [20].

Reprojection error is a measure that can be used to quantify the quality of the stereo calibration. The pinhole camera model equation (eq. (2.4)) transforming a 3D world coordinate $\boldsymbol{X}$ to a 2D image point $\boldsymbol{x}$ can be represented as $\boldsymbol{x} = \boldsymbol{\pi}(\boldsymbol{X})$ Consider two cameras observing a common scene. A feature match between the two images are found, key points denoted as $\boldsymbol{x}$ and $\boldsymbol{x}'$. The reprojection error of the match can then be found by first project the feature point in the second image to a 3D point using the second cameras model, then transform this 3D point to a image point in the first image frame. The distance between the reprojected point $\hat{\boldsymbol{x}}$ and the

image point found in the match $\boldsymbol{x}$ is the reprojection error d [20].

$$d = \boldsymbol{x} - \hat{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{\pi}^{-1}\left[\boldsymbol{\pi}'(\boldsymbol{x}')\right] \tag{2.22}$$

If the camera models were perfect the reprojection error would be zero. Since the model are only an approximate of the real camera it is not realistic to hope for no reprojection error. Most algorithms utilizing the reprojection error therefor only discussed methods of reducing the reprojection error, and not eliminating it.

2.6 Multiple View Geometry

The main advantage of using multiple cameras is that a scene is observed from multiple angles at the same time-instance. When the baseline between the cameras is known, the geometric relationships between them can be utilized to acquire knowledge about the world coordinates.

2.6.1 Two-View Geometry

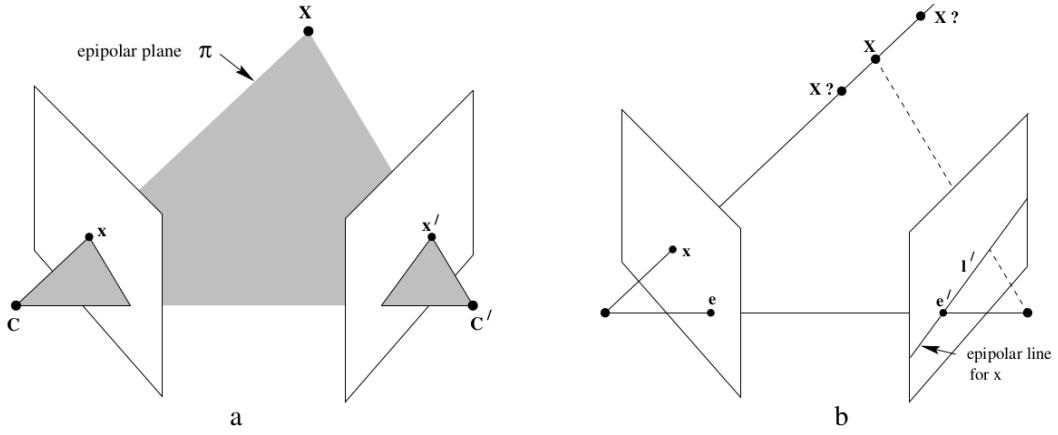


Figure 2.8: Epipolar geometry
Courtesy of Hartley and Zisserman [20].

When two cameras are observing the same landmark $\boldsymbol{X}$, rays to the camera centres $\boldsymbol{C}$ and $\boldsymbol{C}'$, can be created. The rays will intersect the image plane at $\boldsymbol{x}$ and $\boldsymbol{x}'$. When drawing a line from $\boldsymbol{C}$ to $\boldsymbol{C}'$ which intersects the image planes at the epipoles $\boldsymbol{e}$ and $\boldsymbol{e}'$ respectively, a plane is created. This plane is referred to as the epipolar plane. This geometrical relationship is captured by the

fundamental matrix $\boldsymbol{F}$ [20].

$$\boldsymbol{x'}^{\top}\boldsymbol{F}\boldsymbol{x} = 0 \tag{2.23}$$

A special case of the fundamental matrix called the essential matrix $\boldsymbol{E}$. It relates the fundamental matrix to camera matrices $\boldsymbol{K}$ and $\boldsymbol{K'}$. The essential matrix is defined in terms of normalized image coordinates, meaning that the effect of the know calibration matrices $\boldsymbol{K}$ and $\boldsymbol{K'}$ are removed for the points [20].

$$\hat{\boldsymbol{x}}'^{\top}\boldsymbol{E}\hat{\boldsymbol{x}} = 0 \text{ where } \hat{\boldsymbol{x}} = \boldsymbol{K}^{-1}\boldsymbol{x} \tag{2.24}$$

The essential matrix can also be defined using the translation and rotation of the two cameras. If one camera has zero rotation and is located at origin and the other camera to have rotation matrix R and translation vector t, the essential matrix can be defined as:

$$\boldsymbol{E} = [\boldsymbol{t}]_x \boldsymbol{R} \tag{2.25}$$

Where $[.]_x$ is the skew-symmetric matrix operator [8]. Putting all the definitions together the fundamental matrix F can be express according to the intrinsic and extrinsic parameters of the system.

$$\boldsymbol{F} = \boldsymbol{K'}^{-\top}[\boldsymbol{t}]_x \boldsymbol{R}\boldsymbol{K}^{-1} \tag{2.26}$$

Epipolar geometry is often used to simplify and strengthen the search for matches between multiple cameras. If a key point is observed in one image, and the epipoles $\boldsymbol{e}$ and $\boldsymbol{e'}$ is known, their relationship together with $\boldsymbol{x}$ can be used to draw a line in the second image. The corresponding $\boldsymbol{x'}$ must then be located on this epipolar line, hence reducing the matching problem quite drastically.

2.6.2 Three-View Geometry

In a three-view scenario a lot of new geometric relationships appears. These relationships can be utilized in calibration. The trifocal tensor is the three-view equivalent to the fundamental matrix in two-view geometry. The trifocal tensor contains three 3x3 matrices, with a total of 27 elements. One of these elements are a common scaling factor, leaving 26 independent elements

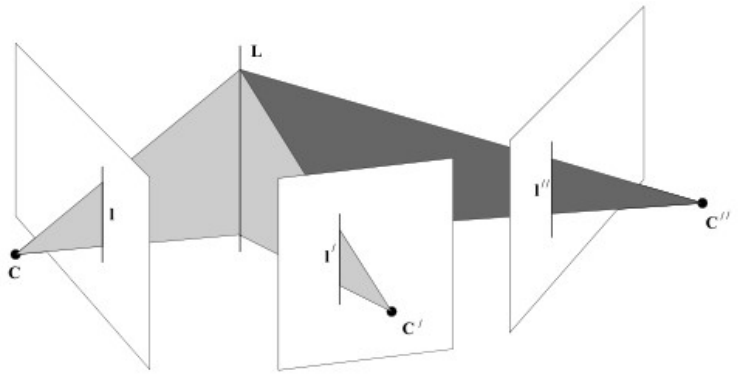


Figure 2.9: Three-view geometry
Courtesy of Hartley and Zisserman [20].

[20].

Given the projection matrices of the three cameras observing a common world point, each element in the trifocal tensor can be calculated. Consider

$$\boldsymbol{P}_1 = \boldsymbol{K}_1 \boldsymbol{R}_1 [\boldsymbol{I}, -\boldsymbol{t}_1] \tag{2.27}$$

$$\boldsymbol{P}_2 = \boldsymbol{K}_2 \boldsymbol{R}_2 [\boldsymbol{I}, -\boldsymbol{t}_2] \tag{2.28}$$

$$\boldsymbol{P}_3 = \boldsymbol{K}_3 \boldsymbol{R}_3 [\boldsymbol{I}, -\boldsymbol{t}_3] \tag{2.29}$$

the l-q-rth elements in the trifocal tensor can be found by

$$\mathbf{T}_l^{qr} = (-1)^{l+1} \det \begin{bmatrix} \sim \mathbf{P_1}^l \\ \mathbf{P_2}^q \\ \mathbf{P_3}^r \end{bmatrix} \tag{2.30}$$

where $\sim \boldsymbol{P}_\mathbf{1}^l$ represents every row in $\boldsymbol{P}_1$ except row l. $\boldsymbol{P}_\mathbf{2}^q$ and $\boldsymbol{P}_\mathbf{3}^r$ is the qth and rth row in $\boldsymbol{P}_2$ and $\boldsymbol{P}_3$ [8].

The trifocal tensor holds lots of the same properties as the fundamental matrix. It can be used to transfer points from matches in two images to a matching point in the third image. With the trifocal tensor both points and line correspondences can be transferred [20].

Chapter 3

Setup

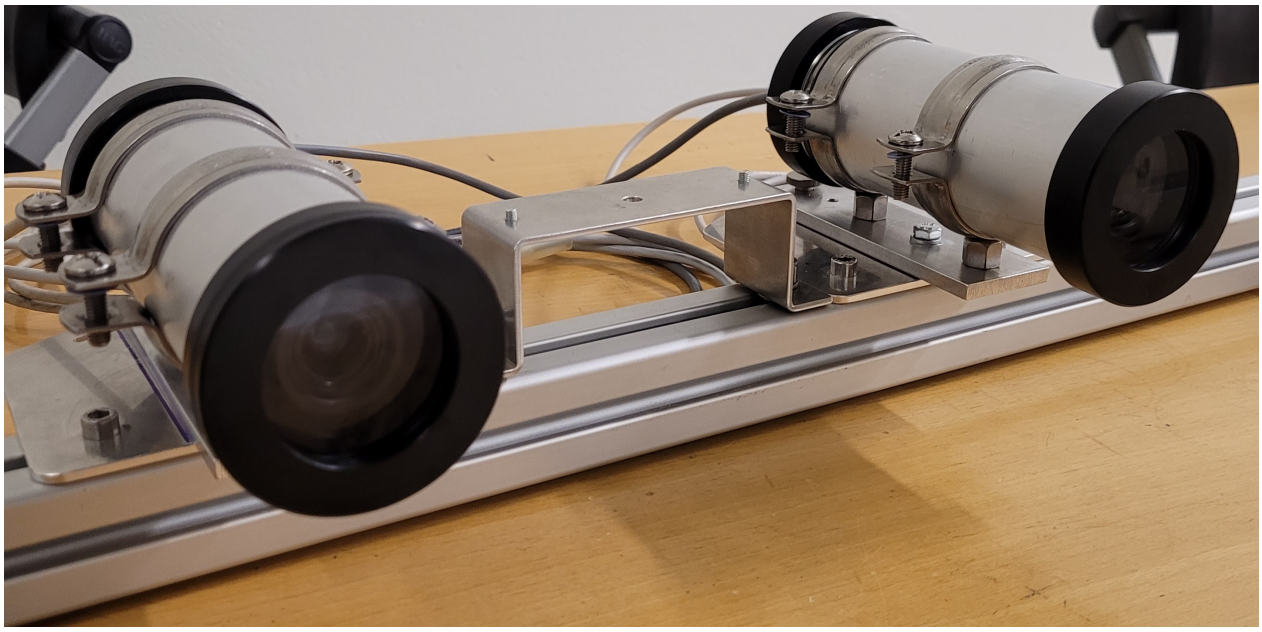


Figure 3.1: The stereo rig.

The stereo rig which the calibration algorithm are designed for are presented in this chapter. The rig consists of two cameras mounted on an aluminium beam. The cameras are mounted on adjustable brackets, creating a flexible and mobile stereo system. The images are captured ether by saving them directly on to the hard disk or by streaming over ROS and then save the ROS bag.

3.1 Hardware

The cameras are provided by FLIR and the optic lenses are from Edmund Optics. Wiring is done according to documentation provided from FLIR, and the cameras communicate over a GigE interface.

3.1.1 Camera

(a) FLIR Blackfly S GigE.

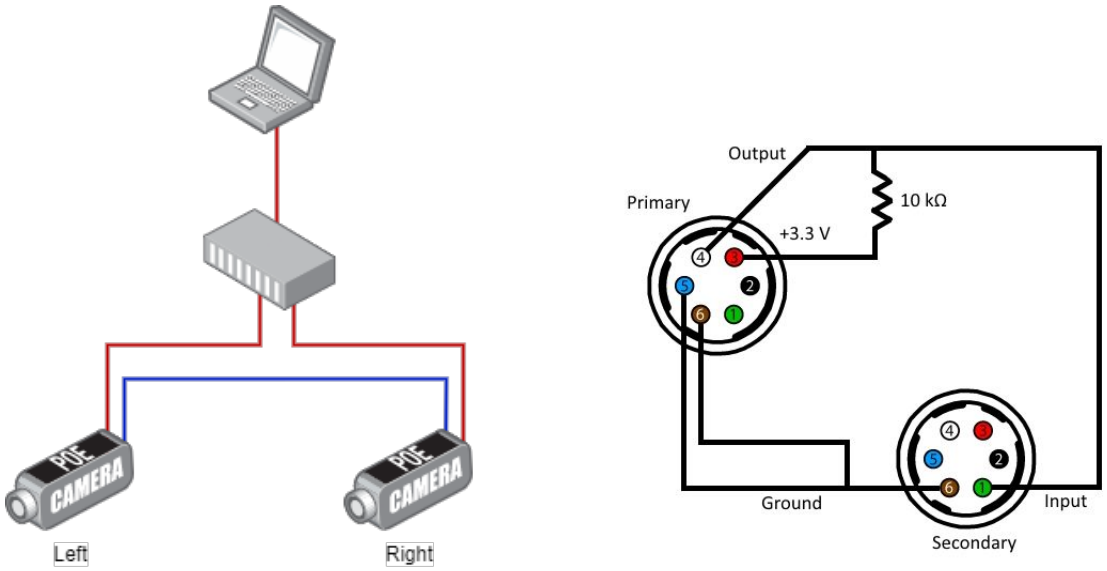
(b) Edmund Optics C Series.

Figure 3.2: Components in stereo system.

The cameras are of model Blackfly S GigE from FLIR. This model has a global shutter, making it less susceptible to fast moving objects in the scene since the whole image is captured in one instance. A Sony IMX264 camera sensor is fitted in the camera, which utilizes CMOS with 2/3" format and has a resolution of 2448x2048. Since the cameras support Power over Ethernet (PoE), the wiring becomes less complicated. The cameras uses the GigE interface to communicate [12].

On both cameras there are fitted a lens from Edmund Optics. The lenses are from their C Series with a fixed focal length of 8.5 mm. Combined with the 2/3" sensor in the FLIR-cameras, the cameras has a horizontal field of view of 59.2°[9].

3.1.2 System Connections



(a) System overview

(b) GPIO wiring

Figure 3.3: Components in stereo system

The two cameras communicate with the computer through a PoE router, which also provides the cameras with power. Both the captured images from the cameras and the trigger command for the cameras are sent over these Ethernet cables (coloured red in fig. 3.3a). The trigger command from the computer is only sent to the master camera which in this system is selected as the left camera. In order to synchronize image capturing in both cameras, the master camera sends a trigger command to the right camera over a General Purpose IO (GPIO) cable (coloured blue in fig. 3.3a). Some interconnections and resistance in the circuit between the cameras are necessary as shown in fig. 3.3b. These were added by customising the cable where the interconnections were done inside a splicing of the cable.

3.1.3 Stereo Rig

How the cameras are mounted in relation to each other has a great impact on how well the stereo system will perform. The choice of setup were based on the work done by Theimann and Olsen, who mounted the cameras on a 1.7 meter long beam[44]. To decrease the uncertainty field, and to be able to detect object over greater distances, a large baseline is preferred. Since the system is to be used on a ferry, there are physical limitations as to how long it can be. The cameras are rotated inwards by 1° to increase the overlapping Field of View (FoV) and reduce the blind

spot in front of the cameras. The resulting optimal rotation and translation vectors from the left camera to the right camera are chosen as:

$$\boldsymbol{R} = \begin{bmatrix} \theta = 0 & \phi = 0 & \psi = -2° \end{bmatrix} \tag{3.1}$$

$$\boldsymbol{t} = \begin{bmatrix} 1.7m & 0 & 0 \end{bmatrix}^\top \tag{3.2}$$

Where θ, ϕ and ψ represents roll, pitch and yaw. The translation results in a fixation point at 50 meters, with a blind spot of 1.6 meters in front of the cameras. At a distance of 50 meters, the camera has a horizontal FoV of 50 meters [44]. While this is the ideal setup, ensuring that the rotation is exactly 1° is near impossible. Just the slightest error will impact the accuracy of the system. To compensate for the mounting inaccuracy, the translation matrices need to be calculated in the calibration.

3.1.4 Weatherproofing

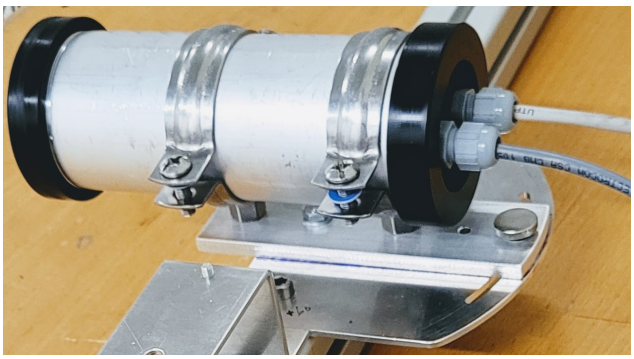


Figure 3.4: The back end of the capsule with the cable glands.

The operating temperature range for the cameras are from 0°C to 50°C, and they are not waterproof [12]. This makes them not well suited for use outdoors use in the Nordic conditions of Trondheim. During the project there was built two metal capsules for the cameras to protect them from the environment. The capsule openings are fitted with O-rings and the cables fitted through cable glands to make it as waterproof as possible. In front of the cameras lenses there are mounted a window of polycarbonate. Testing was performed to make sure that the polycarbonate did not affect the image quality. During testing it was found that cameras produced some heat under operation and that the capsules captured a lot of the heat. This leads to believe that the stereo rig could be used in cold weather, but not be mounted permanently outside during the winter months without the cameras running. For a permanent installation, a better solution for isolation and cooling is still needed.

3.2 Software

When implementing new software to the stereo system, it is important to remember that it has to be compatible with the onboard computer of milliAmpere. The computer on milliAmpere runs Ubuntu 16.05 and the modules communicates over a ROS system.

3.2.1 ROS

ROS is an open source Robot Operating System used to simplify communications between different modules in a complex system. The way it works is simple; every module in the network is operating independently. If they gather information that might be useful to other modules, that information is broadcast over the network under a relevant topic. Modules that are interested in utilizing the information, subscribes to the given topic. The messages that are sent over the network are standardized, meaning that the modules can run different languages with ease. Since the modules works independently, they can be taken in and out of operation without having to shut down the entire network [36].

The OBC on milliAmpere is running on ROS. While in operation, the ROS-system is configured to log data from certain topics. Every message is timestamped and written to a file type called ROS bags, which makes it possible to look at the history of the data gathered. The bags can be played back at a later stage, mirroring the message flow in real time as the broadcast unfolded at the time of recording.

The ROS community holds a lot of open-source packages for different types of hardware and applications, including ready-made nodes for mono and stereo calibration for cameras [29, 30].

3.2.2 Spinnaker SDK Camera Driver

One of the packages available in the community is a driver to operate the FLIR-cameras. They are found on Github[26], and implemented in C++. The thorough guide in the Readme-file in the repository was followed in order to setup the code for the cameras to run. One thing to note is that for the cameras to be recognized by computer, the user first has to change the Ethernet adapters IP-address on the network adapter to match the IP-addresses of the cameras, making sure every component is on the same subnet. The only way of reading the IP-address of the cameras is through a software from FLIR called Spinnview. Spinview can be downloaded on FLIR's web site [11]. It is recommended running Spinnview on a Windows computer, as the

software appeared quite prone to bugs and frozen screens and proved sub optimal to work with. If other changes to the camera settings is to be done, it is recommended to do those straight in the code of the camera driver.

While working with the stereo rig, it was desired to capture colour images. This proved to be harder than anticipated. Both the driver and cameras support colour images, but when trying to run it, the driver crashed after only a few captured images. A lot of time was spent trying find optimal settings to make the driver work, changing the frames per second (fps) of camera capture, buffer size and packet size to no avail. Some runs the cameras misbehaved even when grey-scale images were capture, where some of the images suddenly was dropped. If to many images was dropped in succession the driver crashed. This led to a theory that cameras buffer might fill up over time leading to crashes. Attempts were made where the cameras where properly flushed before starting the camera acquisition, but still no luck. Going forward an idea might be to build a new capture node from scratch, using the spinnaker driver as a blueprint. Eventually after a lot of trial and error, the cameras ran consistently at 20 fps capturing grey-scale images, which was sufficient for experiments run in this report.

3.2.3 OpenCV

Many useful functions in computer vision are available through the open-source library OpenCV. It is natively written in C++, but a lot of the functionality are also available in Python, Java and MATLAB [32]. Since the newly expiration of patent, the SIFT descriptor is only included in the newest update of OpenCV. SURF is only available for non-commercial usage and are only available in the extra functions included in the opencv_contrib package. The algorithm in this report is written in C++ with OpenCV version 4.5.2 including the contrib extensions. Installation guides can be found on OpenCV's website.

Chapter 4

Experiments

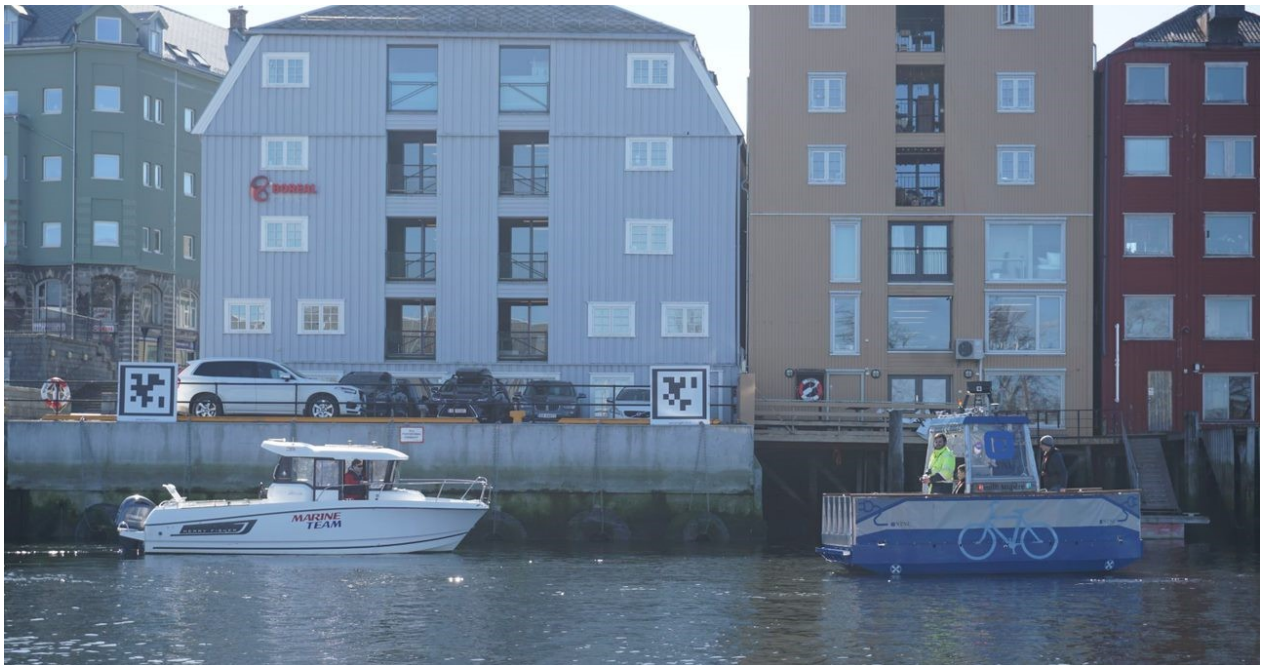
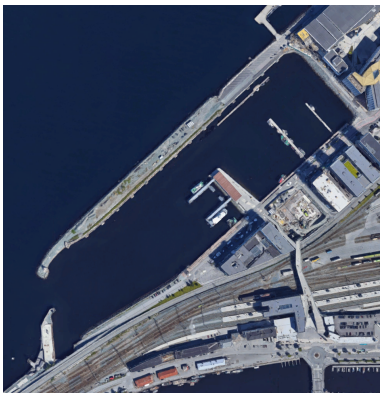


Figure 4.1: The ferry milliAmpere to the right and the leisure boat Havfruen to the left.

During week 15 data was gathered on milliAmpere in the channel of Trondheim. The experiments were a collaboration together with Kristian Auestad, Martin Gerhardsen and Thomas Hellum. Since everyone were working on different theses there was a great variety in which scenarios each person wanted to perform. In total 23 different scenarios were recorded, where 7 of these involved observing a leisure boat. The aim for the experiments was to gather solid data sets so that systems could be tested on realistic data. All data was gathered in ROS bags.

4.1 Scenarios

(a) Trondheim channel.



(b) Brattørbassenget.

Figure 4.2: Locations for data gathering.

Since data was gather for four different master theses, the scenarios can be split into four categories; SLAM, Object tracking, estimation of position according to know locations on shore and auto-calibration.

SLAM

These scenarios are generally the ones that has the longest run times. The goal of these scenarios was to have data sets where the ferry crossed the same locations multiple times in order to test if a SLAM algorithm was able to detect loop closers. Scenarios varied from crossing the channel multiple times to traversing the whole length of the channel. Two of the scenarios was performed sailing some loops outside the channel in Brattørbassenget.

Object Detection

For these scenarios, position data for the target boat is available. These scenarios mainly contain runs where milliAmpere moves ether very slowly or standing still while observing Havfruen at different distances. Head on passing and overtakes are also part of these scenarios. Two of the scenarios was performed at open sea, in order to have as little noise in the scene as possible.

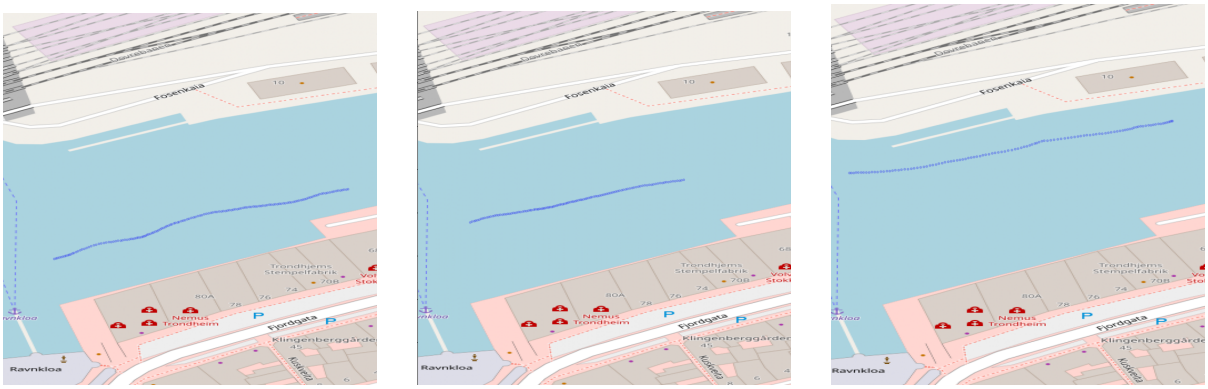
Estimation of Position

At one of the quays along the channel two 1.6x1.6 meter April tags was mounted 13.4 meters apart. The GPS-location of these tags were logged using the same system as on Havfruen. The estimation of position scenarios involved sailing around in front of these tags these tags, making sure that the 360°mono cameras in the sensor rig observed the tags at different distances and angles.

Auto-Calibration

Figure 4.3: The tall buildings observed in the auto-calibration runs.

These are the scenarios that are relevant for this thesis. The aim of these scenarios was to create as much structure in the image as possible in an attempt to cover the entire frame with potential feature matches. While observing the face of a row of tall buildings milliAmpere sailed sideways down the channel. Distance to building faces was varied to compare the performance of the auto-calibration at varying degree of diversity in the matches.



(a) Close route (b) Intermediate route (c) Far route

Figure 4.4: Auto-calibration scenario routes.
The building faces are to the south of the channel

4.2 Data Gathered

To create data sets that was useful for both us and to future students it was desired to harvest as much data as possible from the experiments. For each scenario, the stereo rig sampled stereo

Figure 4.5: The sensor rig in black while the stereo rig is provisionally mounted on the roof of the ferry

images, the sensor rig collected 360° mono-view images and infrared images as well as LiDAR-data. The sensors on milliAmpere recorded its GPS-position, IMU-data, heading and throttle usage. OBC contains nodes that processes the GPS and IMU-data, calculating relative position of the ferry in relation to a set point on shore. Since the stereo rig was running and recording on a separate computer, the ROS bags from milliAmperes OBC and the stereo rig had to be merged afterwards. Every message is timestamped, so the synchronization proved unproblematic.

The leisure boat Havfruen was hired to act as a marker for some of the scenarios. Its GPS-position was tracked using an GPS unit from Advanced Navigation. This system was not running on a ROS, so the data had to be saved locally. The raw data was timestamped and save to an .csv file in order for the ROS data and position data from Havfruen to be synchronized.

A list of all topics contained in the ROS bags is found in the appendix B.

4.2.1 Calibration of the Stereo Cameras

To have a reference of the calibration for the stereo cameras, a traditional calibration using a checkerboard was performed. The stereo calibration node found in the ROS community was used for this purpose. This package is built on OpenCV, performing recognition of the checkerboard, and performing both intrinsic and extrinsic calibrations on the cameras. A thorough discussion of the calibration results are found in section 7.2. The calibration data was saved in a note-file and made available along the data gathered.

Chapter 5

Auto-Calibration

Since the goal of the projection of milliAmpere is to create a totally autonomous ferry which will carry people across the channel, it is important that all the systems are reliable. This includes the stereo system that is to be installed permanently on the ferry. One part of being autonomous is minimizing the need for maintenance. Traditional calibration methods is based on observing an object of know dimensions in different angles [50]. These methods require the ferry to be put out of service and operators to manually perform the calibrations. Even though calibration has been done perfectly, when a camera system is operating over longer periods of time, there will always come a time where a re-calibration is necessary. Many elements can factor in making the present calibration inaccurate. Vibrations and wind can affect the extrinsics of stereo setup, temperature change and fluids on the lens can result in changes of the intrinsic. For a vessel to be as autonomous as possible, it is desirable to have a system which calibrates itself with the data gathered during normal operation.

(a) Channel.

(b) City.
Courtesy of Rehder [38].

Figure 5.1: Comparison of environments

A lot of the auto-calibration algorithms discussed in other papers are developed with cars in mind [8, 14, 25, 38, 46]. One big difference in a urban environment compared to a marine en-

vironment is the level of structure in the scene. When moving through a city there are a lot of static scenery to extract solid features from. Passing buildings, parked cars and structure in the street such as road markings is all easily detectable. At sea there are a lot less texture to pin key points on. Even when sailing in an urban marine environment such as the channel of Trondheim the only static scenery can be hundreds of meters away. While it is not recommended, it might be possible to find matches in the structure of the sea for a stereo image pair taken simultaneously, but finding matches between to different time-instances is not possible. The same problem arises in the sky. This results in large parts of the image depleted of matches. H. Wang et al. implemented a self-calibration algorithm which made lack of structure in a marine environment the algorithms biggest strength. They performed the calibration by observing the sea horizon and got very good results [46]. Inside the channel or fjord of Trondheim there are unfortunately no clear view of the horizon. In order to use one of the approaches designed for land vessel, a way of getting more structure in the scene was needed. A solution is to have the ferry sailing sideways down the channel while observing the face of some tall buildings along the shore of the channel.

A robust algorithm which finds both the intrinsic and extrinsic parameters online are desired. Musleh et al. [25] and Wang et al. [46] presented techniques which only estimates the pose of the cameras. Zhang et al. has created a self-calibration method for both Chang'e-3 and Chang'e-4 which are two lunar rovers. Their method is based on feature detection and bundle adjustment. When they do their bundle adjustment, all images has already been processed, which means that their method is not an online method [48, 49]. An online approach to bundle adjustment was presented by Rehder et al. They break up the bundle adjustment problem into smaller executable task, making it more ideal for online applications. Their algorithm retrieves all the calibration parameters up to scale [38]. Both Gopaul et al. and Dang et al. suggests method using three point matches [8, 15]. Gopaul et al. uses three-view scale restraint equations which are searched a solution for using an optimization algorithm [15]. Dang et al. calibrates the cameras by find the parameters that minimizes the reprojection error. They propose three different methods in their paper. The first one is a reduced bundle adjustment approach, the second uses only one epipolar constraint while the last method uses both epipolar constraint and constraints from the trifocal tensor of three-point matches. One thing that is special with their system is the stereo cameras can be rotated, changing the extrinsics of their system. A continuous calibration is needed to keep up with this movement [8].

Dang et al.'s approach using the trifocal tensor as constraints was the chosen method to pursue in this thesis. The idea of not having to deal with a computationally heavy bundle adjustment seemed like a good strategy for an online implementation.

5.1 Image Processing

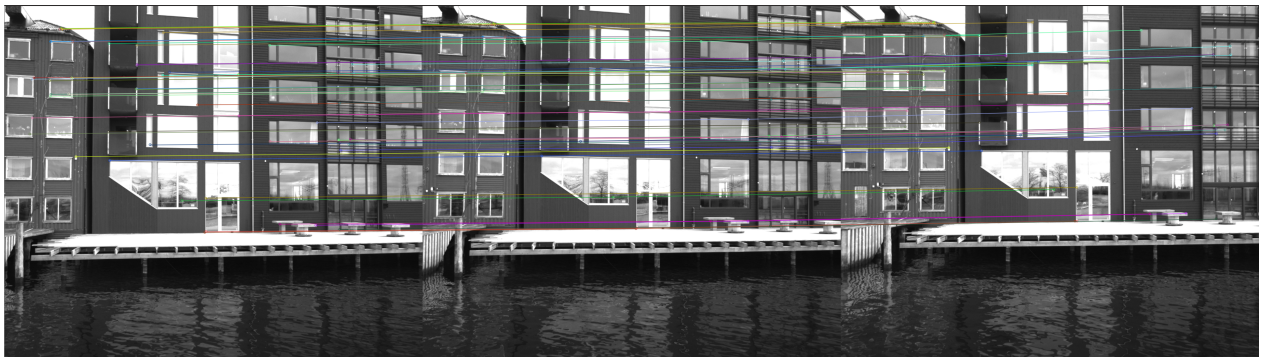


Figure 5.2: Three point matching between left(k+1), left(k) and right(k).

In order to utilize the trilinear constraints suggested by Dang et al.[8], matching has to between three different images. The stereo system only outputs two images for every instance, so one image from the neighbouring epochs is needed to have three images available. Since the left camera was chosen as master camera, with the camera frame originating in lefts position, it was natural to choose two images taken by left camera. Three-point matches was created using one instance of stereo images plus the proceeding left image. All image processing is done using the OpenCV library.

Which feature detector and matching method is used may have a big impact on the performance of the system, both when it comes to precision and run time. The code is built so that it is easy to switch between the different methods. Currently it supports SIFT, SURF and ORB as feature descriptors, and FLANN and Brute force as matchers. SIFT and SURF is run without any restrictions regarding the number of matches they can find, while ORB is capped with a maximum of 5000 key points. The methods sort their key points after confidence level, so if ORB identifies more than 5000 features it will return the 5000 best feature points. Performance of the different methods are discussed in section 7.1.

When a new image pair is detected, the new images are first rectified using the best available intrinsic coefficients. Features from the images is then found using one of the feature descriptors. When key points and their descriptors are extracted from the images, matches can be found. First the matches between the new left and new right images are matched and stored to be used in the next epoch, then the new left key points are matched with the left image of the previous time step. For each matching process, the bad matches are filtered out using RANSAC. In the code this is done using OpenCV's function `findHomography` which has a built in RANSAC option, and outputs a mask which can be used to filter out the outliers. The homography found by the function is not used further in the code. The optimal RANSAC threshold was found to be 15.

At this point a total of three sets of solid matches are available: the stereo images of the current time step, the stereo images in the previous time step and between the left images in two time-steps. The three-point matches are found comparing the matches from the previous time step with the matches between the epochs. A three-point match is identified if same key point in the common image is present in both the matching sets. The common image in this case is the trailing left image.

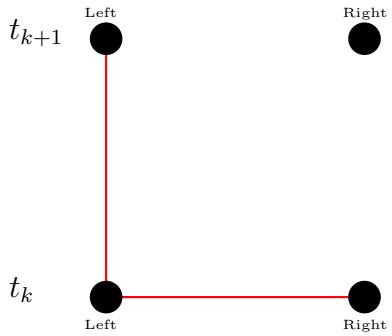


Figure 5.3: Configuration of three point matching

During testing it was found that if the elapsed time between the epochs was too small, the matcher had problem distinguishing the good from the bad matches. The ferry is moving at a slow rating, leading to images between the epochs being almost identical if they are sampled at a high rate. A minimal elapsed time of 1 second was implemented.

5.2 Calibration of Camera Parameters

The target for the auto-calibration method proposed by Dang et al. is to find the intrinsic and extrinsic parameters which minimizes the reprojection error [8]. Given a set S_t of three point matches the cost function that needs to be minimized can be formulated as follows:

$$\sum_{i \in S_t} \|\boldsymbol{d}_L(k)\|^2_{\mathbf{C}_{L,i}(k)} + \|\boldsymbol{d}_R(k)\|^2_{\mathbf{C}_{R,i}(k)} + \|\boldsymbol{d}_L(k+1)\|^2_{\mathbf{C}_{L,i}(k+1)} \tag{5.1}$$

$$\sum_{i \in S_t} \left\|\hat{\mathbf{x}}_{L,i}(k) - \mathbf{x}_{L,i}(k)\right\|^2_{\mathbf{C}_{L,i}(k)} + \left\|\hat{\mathbf{x}}_{R,i}(k) - \mathbf{x}_{R,i}(k)\right\|^2_{\mathbf{C}_{R,i}(k)} \tag{5.2}$$
$$+ \left\|\hat{\mathbf{x}}_{L,i}(k+1) - \mathbf{x}_{L,i}(k+1)\right\|^2_{\mathbf{C}_{L,i}(k+1)}.$$

Where $\boldsymbol{C}$ is the covariance matrix containing the uncertainty of the measurements [8]. A total

of 8 intrinsic, 4 distortion parameters and six extrinsic values needs to be estimated, resulting in a total of 18 free parameters. Finding the optimal solution for the cost function is very complicated. Additionally, since a third images from a different epoch is used, the transform of the third camera also needs to estimated. This brings the total of degrees of freedom up to 24 for the system. To direct the cost function towards the solution a set of constraints are added to the problem. Dang et al. suggest using trilinear constraints formulated from the trifocal tensor [8].

Consider a three point match $\boldsymbol{x}^L - \boldsymbol{x}^R - \boldsymbol{x}^{L+1}$, the trilinear constrain is defined as:

$$g_{qr}\left(\mathbf{T}, \mathbf{x}^L, \mathbf{x}^R, \mathbf{x}^{L+1}\right) = \sum_{l=1}^{3} \mathbf{x}_l^L \left(x_q^R \mathbf{x}_r^{L+1} T_l^{33} - x_r^{L+1} T_l^{q3} - x_q^R T_l^{3r} + T_l^{qr} \right) = 0 \tag{5.3}$$

where the points $\boldsymbol{x}$ are the normalized and on the form $\boldsymbol{x} = \left[x,\ y,\ 1\right]^\top$. $\boldsymbol{x}_2^L$ indicates the y element in $\boldsymbol{x}$. T_l^{qr} is the $l-q-r$'th element in the trifocal tensor:

$$T_l^{qr} = (-1)^{l+1} \det \begin{bmatrix} \sim \mathbf{P}_L^l \\ \mathbf{P}_R^q \\ \mathbf{P}_{L+1}^r \end{bmatrix} \tag{5.4}$$

And the projection matrices are given by

$$\begin{aligned} \boldsymbol{P}_L &= \boldsymbol{K}_L \left[\boldsymbol{I}, \mathbf{0}\right] \\ \boldsymbol{P}_R &= \boldsymbol{K}_R \boldsymbol{R}_R \left[\boldsymbol{I}, -\boldsymbol{t}_R\right] \\ \boldsymbol{P}_{L+1} &= \boldsymbol{K}_L \left[\boldsymbol{R}_{L+1}, \boldsymbol{t}_{L+1}\right]. \end{aligned} \tag{5.5}$$

The variables q and r in eq. (5.3) can be chosen freely between $\{1,2,3\}$, giving rise to 9 different constraints. Four of the constellations are linearly independent and two of them are chosen as the trilinear constraints in the algorithm; $g_{1,1}$ and $g_{1,2}$ [8].

In addition to the two trilinear constraints, the epipolar constraint is used as well. This is a constraint only between the stereo image pair.

$$h_e\left(\mathbf{F}, \mathbf{x}^L, \mathbf{x}^R\right) = \begin{pmatrix} \mathbf{x}^R \\ 1 \end{pmatrix}^T \mathbf{F} \begin{pmatrix} \mathbf{x}^L \\ 1 \end{pmatrix} = 0 \tag{5.6}$$

where

$$\mathbf{F} = \mathbf{K}_R^{-T} \mathbf{R}_R \left[-\mathbf{t}_R\right]_\times \mathbf{K}_L^{-1} \tag{5.7}$$

Now that all the constraints are defined, the full optimization problem can be assembled. Consider a set S_t of three point matches, the algorithm needs minimize this cost function [8]:

$$\begin{aligned} \sum_{i \in S_t} \left\| \hat{\mathbf{x}}_{L,i}(k) - \mathbf{x}_{L,i}(k) \right\|^2_{\mathbf{C}_{L,i}(k)} + \left\| \hat{\mathbf{x}}_{R,i}(k) - \mathbf{x}_{R,i}(k) \right\|^2_{\mathbf{C}_{R,i}(k)} \\ + \left\| \hat{\mathbf{x}}_{L,i}(k+1) - \mathbf{x}_{L,i}(k+1) \right\|^2_{\mathbf{C}_{L,i}(k+1)} \end{aligned} \tag{5.8}$$

s.t.

$$\boldsymbol{h} = \begin{bmatrix} g_{11}\left(\mathbf{T}, \mathbf{x}_{L,i}(k), \mathbf{x}_{R,i}(k), \mathbf{x}_{L,i}(k+1)\right) \\ g_{12}\left(\mathbf{T}, \mathbf{x}_{L,i}(k), \mathbf{x}_{R,i}(k), \mathbf{x}_{L,i}(k+1)\right) \\ h_e\left(\mathbf{F}, \mathbf{x}_{L,i}, \mathbf{x}_{R,i}\right) \end{bmatrix} = \mathbf{0} \tag{5.9}$$

for all $i \in S_t$.

The cost function is very non-linear considering that three squared terms are added to the equation for every match. Together with the three multilinear constraints per match and the 24 degrees of freedom of the parameters, the problem optimization problem is extremely hard to solve. Since the problem is non-linear there will be several locally optimal solutions, but in the calibration algorithm, only the global optimum is interesting. All other calibration configurations will result in bad performance of the cameras.

Good initial values of the parameters will enhance the chances of finding the global solution. When the system is to be put into service, it will be reasonable to assume that a previous set of calibration parameters will be available. Changes to the parameters during normal operation will be moderate, making the old parameters a good starting for the new calibration. If bigger changes to the system is performed such as moving and rotating the cameras or changing the focus of lenses, a traditional calibration using a checkerboard will be recommended. The auto-calibration algorithm can then be used to refine the parameters afterwards.

5.2.1 Model Simplifications

Ideally the algorithm should be able to estimate all the parameters, but since optimization problem is so complex it is necessary to simplify the problem where it is possible. One simplification has already been done when the origin of the camera frame was chosen as the origin of the left camera. This halved the extrinsic parameters. The algorithm assumes rectified images, so it is not able to estimate the distortion coefficients. Distortion coefficients from the checkerboard calibration is used to undistort the images.

Dang et al. did a thorough error sensitivity analysis in their paper. They argue that the image centre p_x and p_y should be omitted in the auto-calibration method because of the strong correlation between the error from images centre offset and error in the extrinsic angles [8]. The baseline translation $\boldsymbol{t}_r$ is also left out because the scale in the image is not observable without knowledge of the viewed scene. Dang et al.'s analysis shows that small errors in the baseline doesn't have a big effect on the performance [8]. The image centres for the cameras are kept constant with the value for the checkerboard calibration. Baseline between the cameras can easily found by measuring the distance them. Finally the pixels in the camera are assumed to be square resulting in the focal length parameters fx and fy being identical

All of these simplifications reduce the degrees of freedom of the parameters a lot. The parameter list has gone from 28 to 11 elements. The parameters left to be estimated are as follows:

- $\boldsymbol{t}_{ep}$ translation of left camera between epochs.
- $\boldsymbol{\omega}_{ep}$ rotation of left camera between epochs.
- $\boldsymbol{\omega}_R$ extrinsic angles of right camera.
- f_L focal length of left camera.
- f_R focal length of right camera.

5.2.2 Extended Kalman Filter

There are a lot of methods to solve an optimization problem. One of these methods is using a Kalman filter. Kalman filters a well-known technique for estimating the state of a system. The filter will by design try to minimize the error between the true state and the estimated stated [41]. A Kalman filter take in to account that there may be noise in the system that it is trying to model [37].

The filter goes through three steps for every iteration of the filter. First step is a prediction step where the state and covariance matrix are predicted using the previous state and newest con-

trol inputs. The next step is the innovation step where a Kalman gain is calculated based on the measurements. In the last step, state and covariance matrix are updated using the Kalman gain. Since the optimization problem in this algorithm is non-linear, an Extended Kalman Filter (EKF) is needed to find the solution. EKF differs from normal Kalman Filter since the non-linear system is linearized about the current state estimate and measurement.

To set up the EKF a state vector and a state transition function needs to be defined. State vector was defined as

$$\boldsymbol{z} = \left[\boldsymbol{t}_{ep},\, \boldsymbol{\omega}_{ep},\, \theta_R,\, \phi_R,\, \psi_R,\, f_L,\, f_R\right]^\top. \tag{5.10}$$

The state transition function models how the state is expected to behave between the time steps. Focal length and extrinsic angles are expected to be static, so now change is expected under normal operation, except the addition of some noise variable n. milliAmpere is logging its position and heading. By utilizing the navigation data as a control input $\boldsymbol{u}$, estimating the movement of the ferry between the time step is avoided. How the navigation data is handled is described in section 5.3. The system is only interest in the cameras relative position between each time step and not their true position in the world frame, the old state is discarded, and the control inputs used directly as the new state. Noise variables n are added to the epoch transform as well. The resulting state transition function is defined as

$$\boldsymbol{z}(k+1) = \boldsymbol{f}(\boldsymbol{z}(k), \boldsymbol{u}(k)) \tag{5.11}$$

$$\begin{bmatrix} \boldsymbol{t}_{ep}(k+1) \\ \boldsymbol{\omega}_{ep}(k+1) \\ \theta_R(k+1) \\ \phi_R(k+1) \\ \psi_R(k+1) \\ f_L(k+1) \\ f_R(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \theta_R(k) \\ \psi_R(k) \\ \phi_R(k) \\ f_L(k) \\ f_R(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{I}_3 & 0 \\ 0 & \boldsymbol{I}_3 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_{t_{ep}}(k) \\ \boldsymbol{u}_{\omega_{ep}}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{n}_{t_{ep}} \\ \boldsymbol{n}_{\omega_{ep}} \\ n_{\omega_R} \\ n_{\omega_R} \\ n_{\omega_R} \\ n_f \\ n_f \end{bmatrix}. \tag{5.12}$$

Predict

Prediction is calculate using the *a posteriori* state $\boldsymbol{z}^+(k)$ and the corresponding $\boldsymbol{P}^+(k)$ from the previous time step. The *a priori* is given as

$$\begin{aligned}\mathbf{z}^-(k+1) &= \mathbf{f}\left(\mathbf{z}^+(k), \mathbf{u}(k)\right) \\ \mathbf{P}^-(k+1) &= \mathbf{F}\mathbf{P}^+(k)\mathbf{F}^T + \mathbf{L}\mathbf{Q}_c(k)\mathbf{L}^T + \mathbf{Q}_s(k)\end{aligned} \tag{5.13}$$

where

$$\mathbf{F} = \partial\mathbf{f}/\,\partial\mathbf{z}|_{\mathbf{z}^+(k),\mathbf{u}(k)} \text{ and } \mathbf{L} = \partial\mathbf{f}/\,\partial\mathbf{u}|_{\mathbf{z}^+(k),\mathbf{u}(k)} \tag{5.14}$$

$\boldsymbol{Q}_c(k)$ and $\boldsymbol{Q}_s(k)$ are the covariance matrices for the white noise independent Gaussian random variables in the control input and state, respectively [8]. They are modelled as

$$\boldsymbol{Q}_c = \begin{bmatrix} n_{t_{ep}} & 0 & 0 & 0 & 0 & 0 \\ 0 & n_{t_{ep}} & 0 & 0 & 0 & 0 \\ 0 & 0 & n_{t_{ep}} & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{\omega_{ep}} & 0 & 0 \\ 0 & 0 & 0 & 0 & n_{\omega_{ep}} & 0 \\ 0 & 0 & 0 & 0 & 0 & n_{\omega_{ep}} \end{bmatrix} \tag{5.15}$$

$$\boldsymbol{Q}_s = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & n_{\omega_R} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{\omega_R} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{\omega_R} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_f & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_f \end{bmatrix} \tag{5.16}$$

Innovate

The innovation step starts by linearizing the constraints $\boldsymbol{h}$ about the operating point $(\hat{\boldsymbol{x}}, \boldsymbol{z}^-)$. This is done by partial derivative the constraints first over the state and then over the measurements.

$$\mathbf{A} = \partial \mathbf{h} / \partial \mathbf{z}|_{\hat{\mathbf{x}},\mathbf{z}^-}, \quad \mathbf{B} = \partial \mathbf{h} / \partial \mathbf{x}|_{\hat{\mathbf{x}},\mathbf{z}^-} \tag{5.17}$$

Especially derivation over the state can be very cumbersome as this involves derivation of eq. (5.5). Petersen and Pedersen describes how to derivate a determinant [35]. The trifocal constraints derivated by f_L is given as

$$\frac{\partial \det \boldsymbol{h}}{\partial f_L} = \operatorname{trace}\left(\operatorname{adj}(\boldsymbol{h}) \frac{\partial \boldsymbol{h}}{\partial f_L}\right). \tag{5.18}$$

When $\boldsymbol{A}$ and $\boldsymbol{B}$ are calculated, the Kalman gain can be found.

$$\mathbf{S} = \left(\mathbf{A}\mathbf{P}^-\mathbf{A}^\top + \mathbf{B}\mathbf{R}\mathbf{B}^\top\right)^{-1} \tag{5.19}$$

$$\mathbf{K} = \mathbf{P}^-\mathbf{A}^\top\mathbf{S} \tag{5.20}$$

Where $\boldsymbol{R}$ is the covariance matrix of the white noise in the measurements. The size of the $\boldsymbol{R}$ matrix depends on the number of matches and given by

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{m1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R}_{m2} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{R}_{m3} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{R}_{mi} \end{bmatrix}, \; \boldsymbol{R}_{mi} = \begin{bmatrix} n_{mx} & 0 & 0 & 0 & 0 & 0 \\ 0 & n_{my} & 0 & 0 & 0 & 0 \\ 0 & 0 & n_{mx} & 0 & 0 & 0 \\ 0 & 0 & 0 & n_{my} & 0 & 0 \\ 0 & 0 & 0 & 0 & n_{mx} & 0 \\ 0 & 0 & 0 & 0 & 0 & n_{my} \end{bmatrix}. \tag{5.21}$$

Where n_{mx} and n_{my} are the measurement noises.

Update

Now that the Kalman gain is calculated, it is time to update the state vector and covariance matrix. The *a posteriori* state and covariance are given as

$$\mathbf{z}^+ = \mathbf{z}^- - \mathbf{K}\mathbf{h}(\hat{\mathbf{x}}, \mathbf{z}^-) \tag{5.22}$$

$$\mathbf{P}^+ = (\mathbf{I} - \mathbf{K}\mathbf{A}) \cdot \mathbf{P}^- \tag{5.23}$$

The *a posteriori* matrices $\boldsymbol{z}^+$ and $\boldsymbol{P}^+$ are used as the starting point for the next iteration of the EKF and represents the current best estimation of the state.

5.2.3 Initial Values

In order for the EKF to be stable and not diverge during estimation of complicated states, the initial values of the *a posteriori* matrices are very important. The initial value for the state is chosen close to the result form the checkerboard calibration. $\boldsymbol{t}_{ep}$ and $\boldsymbol{\omega}_{ep}$ set to the first control input.

$$\boldsymbol{z}_0^+ = \left[\boldsymbol{t}_{ep} = \boldsymbol{u}_{t_{ep},0},\ \boldsymbol{\omega}_{ep} = \boldsymbol{u}_{\omega_{ep},0},\ \theta_R = 0,\ \phi_R = 0,\ \psi_R = -4°,\ f_L = 1230,\ f_R = 1230\right]^\top \tag{5.24}$$

For the covariance matrix the initial value is usually chosen with the variance σ^2 of every state variable along the diagonal [41]. This represents the confidence the EKF should have in the initial state.

$$\boldsymbol{P}_0^+ = \begin{bmatrix} \sigma^2_{t_{x,ep}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{t_{y,ep}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{t_{z,ep}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\theta_{ep}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\phi_{ep}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{\psi_{ep}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{\theta_R} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{\phi_R} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{\psi_R} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{f_L} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma^2_{f_R} \end{bmatrix} \tag{5.25}$$

5.3 ROS Node

Since all the data is organized in ROS bags, and the system is to be implemented on the ROS system on the OBC of milliAmpere in the future, it was decided that the auto-calibration algorithm should be implemented directly as a ROS node.

The data that is essential for the auto-calibration algorithm is the stereo images, as well as the position of the ferry at the time when the images was taken. Left and right images are broadcast at the topics `/camera_array/left/image_raw` and `/camera_array/right/image_raw`. These two topics will always be synchronize due to the configuration explained in section 3.1.2. Processed navigation data is available through the topic `/navigation/nav_estimate`. An approximate time synchronizer, which is part of the ROS interface [31], is equipped to make sure that the three topics are in sync. The navigation topic is updated about 20 times for every stereo image, so the time difference between the stereo images and the latest navigation update is only 20 milliseconds.

The navigation topic contains messages which describes milliAmperes North-East-Down (NED) position and heading in reference to a known landmark. Heading is described in quotations and are transformed to Euler angles following these equations [4]:

$$\boldsymbol{q} = \left[q_w,\ q_x,\ q_y,\ q_z\right]^\top \tag{5.26}$$

$$\begin{aligned}\theta &= \tan^{-1}\left(2\frac{q_w q_z + q_x q_y}{1-2\left(q_y^2+q_z^2\right)}\right)\\ \phi &= \sin^{-1}(2(q_w q_y - q_x q_z))\\ \psi &= \tan^{-1}\left(2\frac{q_w q_x + q_y q_z}{1-2\left(q_x^2+q_y^2\right)}\right)\end{aligned} \tag{5.27}$$

The calibration algorithm is only interested in the relative movement between two time-steps, so the coordinate of the landmark is not relevant. The camera frame is not coinciding with the orientation of NED leading to the desire of describing the heading and movement in relation to the camera frame instead. The camera frame is defined as the z-axis pointing inwards in the scene, y-axis pointing down and x-axis being horizontal. A camera looking in the x direction with respect to NED can be described in camera frame as

$$\boldsymbol{X}^{camframe} = \boldsymbol{R}_Z(-90°)\boldsymbol{R}_Y(-90°)\boldsymbol{X}^{NED} \tag{5.28}$$

To describe the position of left camera, the heading of the ferry needs to be added to point.

$$\boldsymbol{R}(\omega)\boldsymbol{X}^{camleft} = \boldsymbol{X}^{camframe} \tag{5.29}$$

$$\boldsymbol{X}^{camleft} = \boldsymbol{R}(\omega)^\top \boldsymbol{X}^{camframe} \tag{5.30}$$

Where

$$\boldsymbol{R}(\omega) = \boldsymbol{R}_Z(\theta)\boldsymbol{R}_X(\phi)\boldsymbol{R}_Y(\psi) \tag{5.31}$$

When the position of the ferry is described in the frame of the left camera, the movement of the camera between the time steps is a simple operation of subtracting the start position from the finish position. Since the heading is described in roll, pitch and of the vessel, no further action is needed to transform them into the camera frame.

5.4 Algorithm Overview

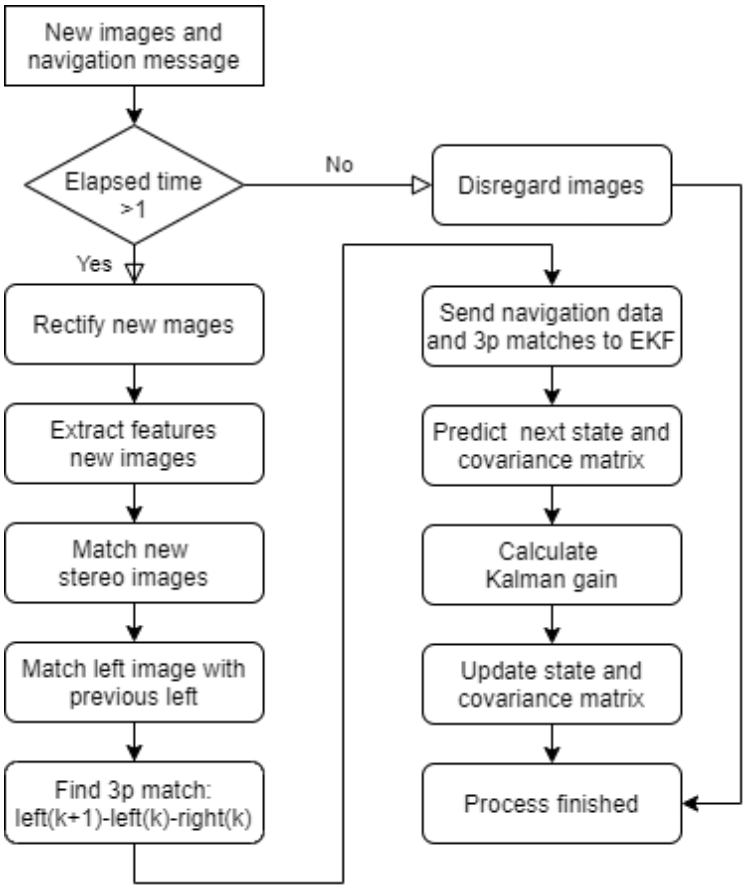


Figure 5.4: Overview of the auto-calibration algorithm.

The algorithm can be summarized by the flowchart in fig. 5.4. The ROS node is constantly listening for image and navigation messages. A call-back function is triggered when messages are present. Images are first processed, before the three-point matches and navigation data is sent to the EKF. The EKF goes through the predict, innovation and update state.

Chapter 6

Results

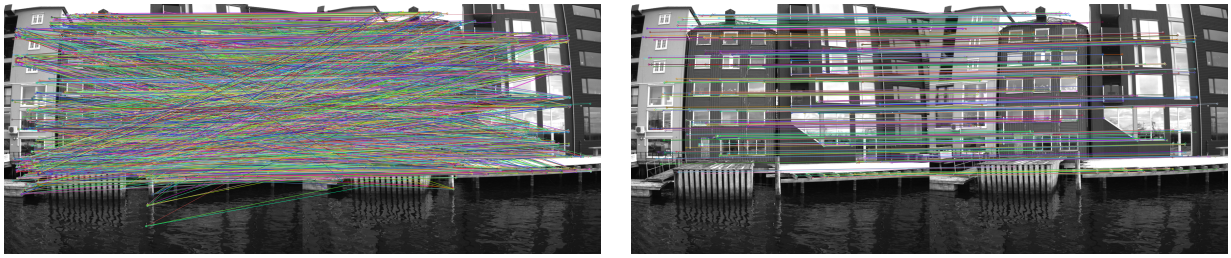
The auto-calibration algorithm was tested on three different scenarios. They are all very similar, where the cameras are aimed at some tall buildings in the channel but the distance to the buildings are varied. In the first part of this chapter the results from the matching process is introduced. Then follows an overview of the auto-calibrations performance on the different runs. Finally, the run time of the algorithm is presented.

6.1 Matching

Three of the feature descriptors and the two matching methods presented in section 2.3 were tested on each of the three output scenarios. For each run, one instance of the three-point matching is shown as well as a table of the average number of matches found for the two instances of two-point matchings and the-three point matcher using every configurations of matchers and descriptors.

6.1.1 RANSAC

These two images demonstrate the performance of RANSAC in one instance of two-point matching



(a) Before. (b) After.

Figure 6.1: Result of RANSAC in close run using ORB as descriptor

6.1.2 Close Run

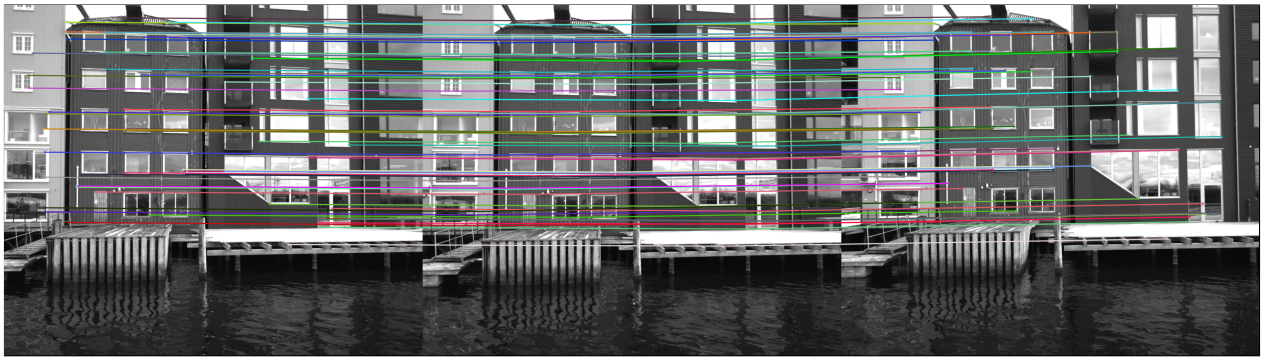


Figure 6.2: Three point matching in the close run

Feature method	Stereo matches	Epoch matches	3p matches	Runtime[ms]
SIFT + FLANN	303.440	293.250	47.267	498.052
SIFT + Bf	304.573	289.043	46.316	476.082
SURF + FLANN	818.042	608.740	102.672	648.009
SURF + Bf	814.052	595.983	100.250	788.158
Orb + FLANN	444.949	432.692	63.598	239.641
Orb + Bf	452.077	384.410	57.564	191.625

Table 6.1: Average number of matches per time step during the close run.

6.1.3 Intermediate Run

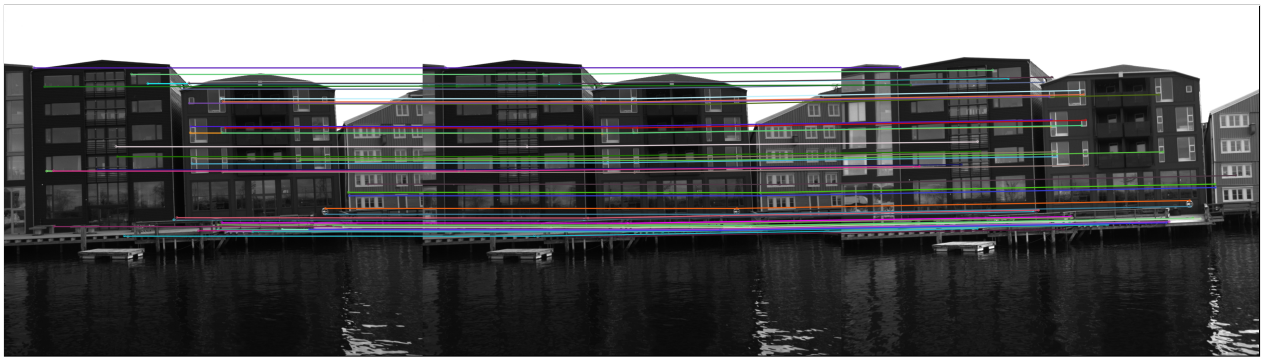


Figure 6.3: Three point matching in the intermediate run

Feature method	Stereo matches	Epoch matches	3p matches	Runtime[ms]
SIFT + FLANN	205.447	167.681	25.035	429.745
SIFT + Bf	201.942	164.115	23.942	420.517
SURF + FLANN	493.700	274.593	39.607	498.007
SURF + Bf	485.563	263.465	37.715	492.558
Orb + FLANN	461.329	344.151	46.212	227.473
Orb + Bf	436.370	265.000	34.103	182.442

Table 6.2: Average number of matches per time step during the intermediate run.

6.1.4 Far Run

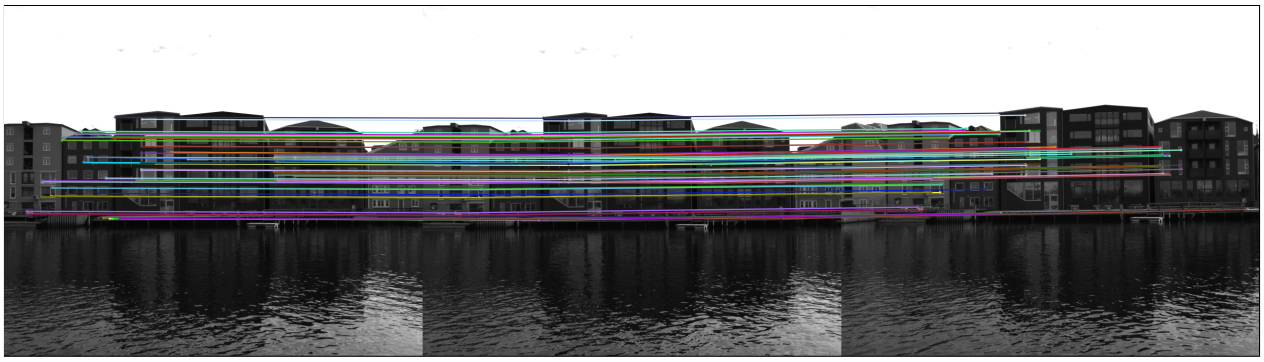


Figure 6.4: Three point matching in the far run.

Feature method	Stereo matches	Epoch matches	3p matches	Runtime[ms]
SIFT + FLANN	409.707	519.293	163.983	547.919
SIFT + Bf	423.543	525.517	169.577	722.847
SURF + FLANN	692.026	663.759	178.293	528.837
SURF + Bf	692.735	658.735	175.282	554.907
Orb + FLANN	411.655	398.422	98.670	238.274
Orb + Bf	398.716	410.216	99.138	207.762

Table 6.3: Average number of matches per time step during the far run.

6.2 Calibration

After analysing of the feature matching, it was decided to test the auto-calibrator using only SIFT and ORB as descriptor and FLANN as matcher. The three scenarios are tested using the two descriptors. For every run, the final state is shown. Then the development of the deviation from the ground truth calibration is presented. The 3D reconstruction error compared to the ground truth is plotted before a table displaying the average error over the scenario is displayed.

3D reconstruction is calculated using this formula

$$\epsilon_{rel} = \frac{\|\hat{\mathbf{X}} - \mathbf{X}\|}{\|\mathbf{X}\|}. \tag{6.1}$$

Where $\hat{\mathbf{X}}$ is the 3D representation of the 2D points from triangulation using the estimated calibration parameters. $\mathbf{X}$ is the triangulated point using the parameters from the ground truth [8].

6.2.1 Ground Truth

To have something to compare the calibration results with, a ground truth calibration was performed using a checkerboard. The method is based on Zhang's method, and the calibration was performed using an existing ROS package available online [30]. The intrinsics and extrinsics are presented below as well as the reprojection error from the images used to perform the ground truth calibration.

$$\boldsymbol{K}_L = \begin{bmatrix} 1235.921 & 0 & 618.280 \\ 0 & 1235.131 & 534.628 \\ 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{K}_R = \begin{bmatrix} 1235.838 & 0 & 640.793 \\ 0 & 1236.346 & 529.334 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

$$\boldsymbol{R}_R\left(\theta = -0.013,\ \phi = 0.011,\ \psi = -0.101\right)\ [rad] \tag{6.3}$$

$$\boldsymbol{t}_R = \begin{bmatrix} 1741.601mm & -9.331mm & 91.0434mm \end{bmatrix}^\top \tag{6.4}$$

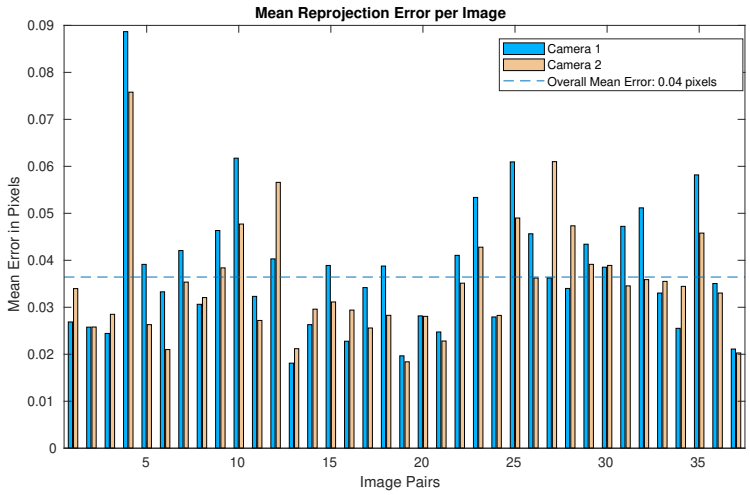


Figure 6.5: Reprojection error from checkerboard calibration.

6.2.2 Initial Values

The initial values of the EKF used to produce the results are listed below. The values were unchanged between the different scenarios.

$$\boldsymbol{z}_0^+ = \left[\boldsymbol{t}_{ep} = \boldsymbol{u}_{t_{ep},0},\ \boldsymbol{\omega}_{ep} = \boldsymbol{u}_{\omega_{ep},0},\ \theta_R = 0,\ \phi_R = 0,\ \psi_R = -4°,\ f_L = 1230,\ f_R = 1230\right]^\top \tag{6.5}$$

Noise	Value
$n_{t_{ep}}$	10^{-7}
$n_{\omega_{ep}}$	10^{-7}
n_{ω_R}	10^{-8}
n_{f_L}	10^{-8}
n_{f_R}	10^{-8}
n_{mx}	10^{-4}
n_{my}	10^{-4}

Table 6.4: Initial noise values.

σ	Value
$\sigma_{t_{ep}}$	0.1
$\sigma_{\omega_{ep}}$	0.05°
σ_{θ_R}	0.5°
σ_{ϕ_R}	0.5°
σ_{ψ_R}	2°
σ_{f_L}	7
σ_{f_R}	7

Table 6.5: Initial standard deviation values.

6.2.3 Close Run with SIFT as Descriptor

θ	ϕ	ψ	f_L	f_R
-0.029	0.014	0.168	1227.596	1229.280

Table 6.6: Final values for calibration parameters after the close-SIFT scenario.

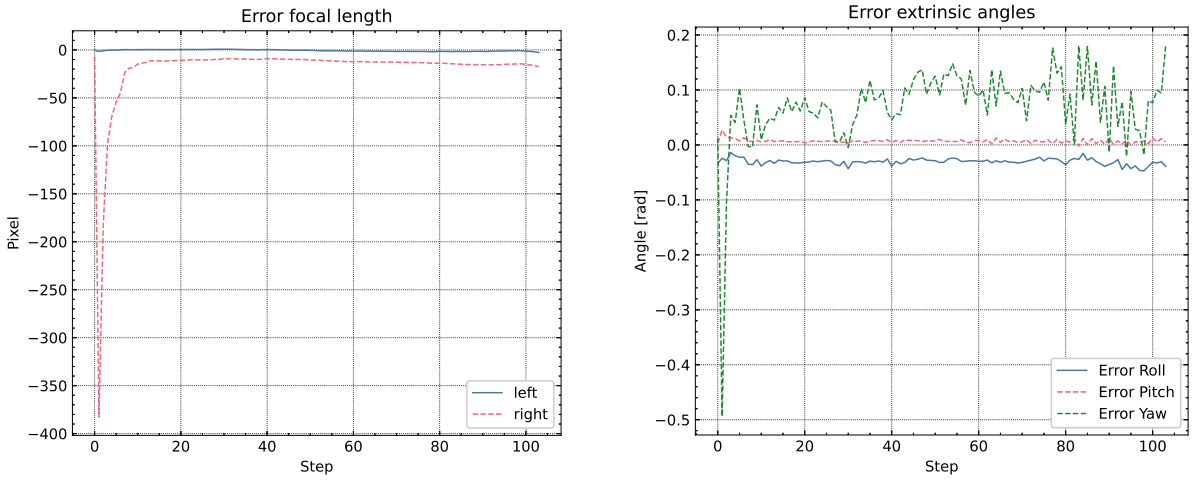


Figure 6.6: Errors in state estimated in the close-SIFT scenario.

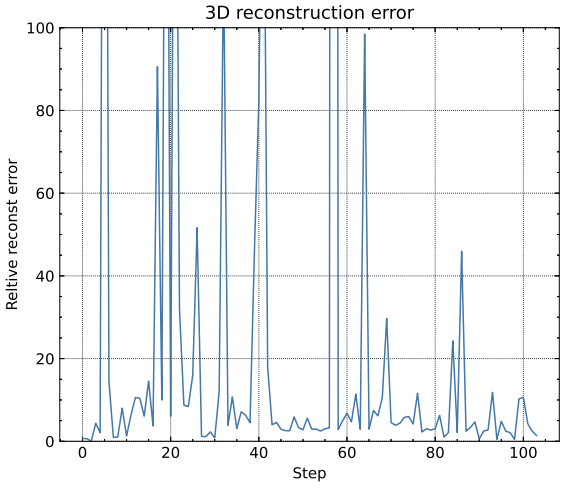


Figure 6.7: 3D reconstruction error for close-SIFT scenario.

Variable	Average Error
θ	-0.031
ϕ	0.007
ψ	0.072
f_L	-0.591
f_R	-19.679
3D rec	29.979

Table 6.7: Average errors from close-SIFT scenario.

6.2.4 Close Run with ORB as Descriptor

θ	ϕ	ψ	f_L	f_R
-0.050	0.009	0.354	1218.965	1223.615

Table 6.8: Final values for calibration parameters after the close-ORB scenario.

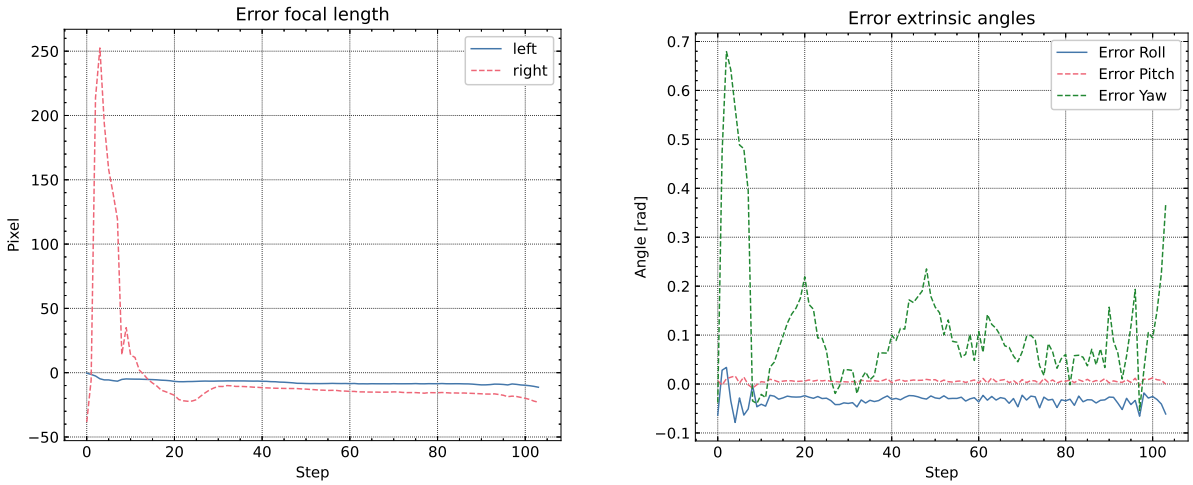


Figure 6.8: Errors in state estimated in the close-ORB scenario.

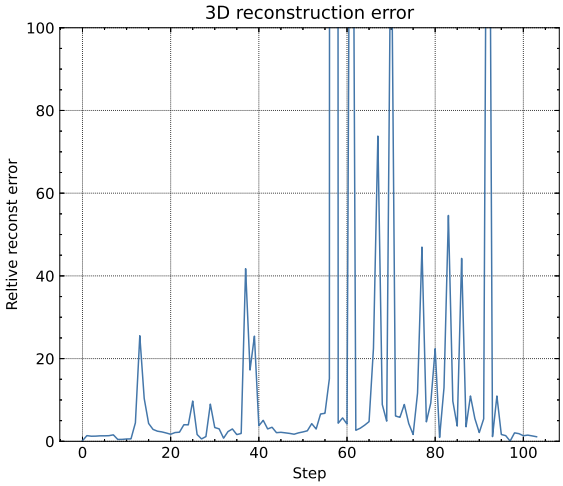


Figure 6.9: 3D reconstruction error for close-ORB scenario.

Variable	Average Error
θ	-0.032
ϕ	0.006
ψ	0.112
f_L	-7.511
f_R	-2.244
3D rec	39.591

Table 6.9: Average errors from close-ORB scenario.

6.2.5 Intermediate Run with SIFT as Descriptor

θ	ϕ	ψ	f_L	f_R
-0.034	0.010	-0.001	1225.580	1235.830

Table 6.10: Final values for calibration parameters after the intermediate-SIFT scenario.

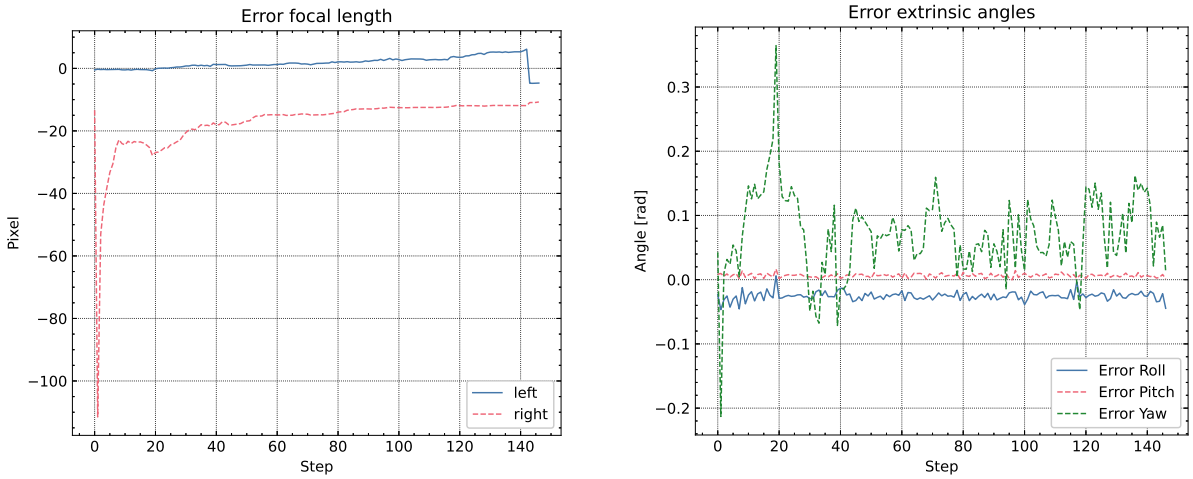


Figure 6.10: Errors in state estimated in the intermediate-SIFT scenario.

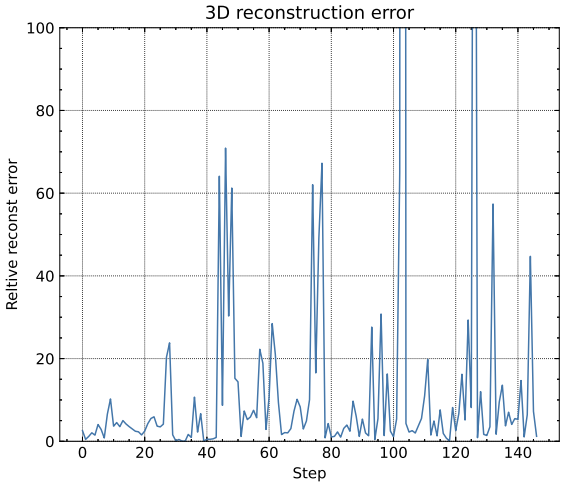


Figure 6.11: 3D reconstruction error for intermediate-SIFT scenario.

Variable	Average Error
θ	-0.025
ϕ	0.006
ψ	0.072
f_L	1.737
f_R	-17.207
3D rec	19.703

Table 6.11: Average errors from intermediate-SIFT scenario.

6.2.6 Intermediate Run with ORB as Descriptor

θ	ϕ	ψ	f_L	f_R
-0.020	0.014	0.087	1236.096	1237.344

Table 6.12: Final values for calibration parameters after the intermediate-ORB scenario.

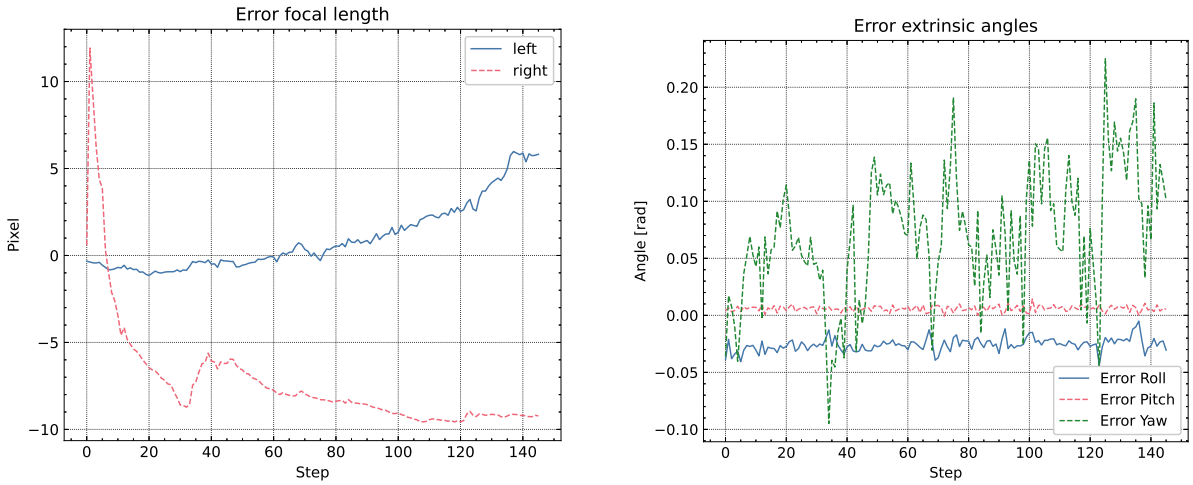


Figure 6.12: Errors in state estimated in the intermediate-ORB scenario.

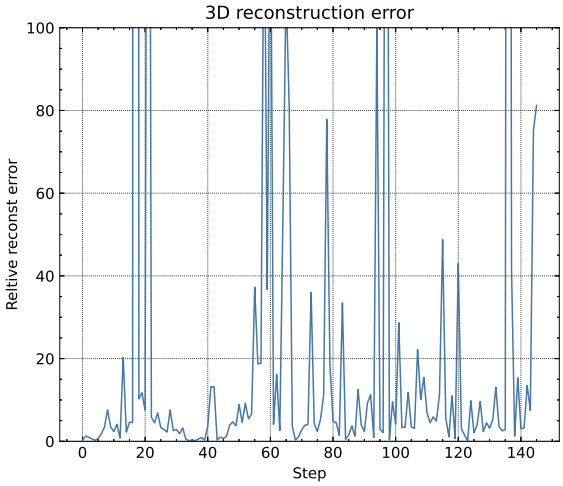


Figure 6.13: 3D reconstruction error for intermediate-ORB scenario.

Variable	Average Error
θ	-0.0256
ϕ	0.006
ψ	0.071
f_L	0.980
f_R	-7.271
3D rec	44.870

Table 6.13: Average errors from intermediate-ORB scenario.

6.2.7 Far Run with SIFT as Descriptor

θ	ϕ	ψ	f_L	f_R
0.010	0.011	-0.483	1298.135	1615.022

Table 6.14: Final values for calibration parameters after the far-SIFT scenario.

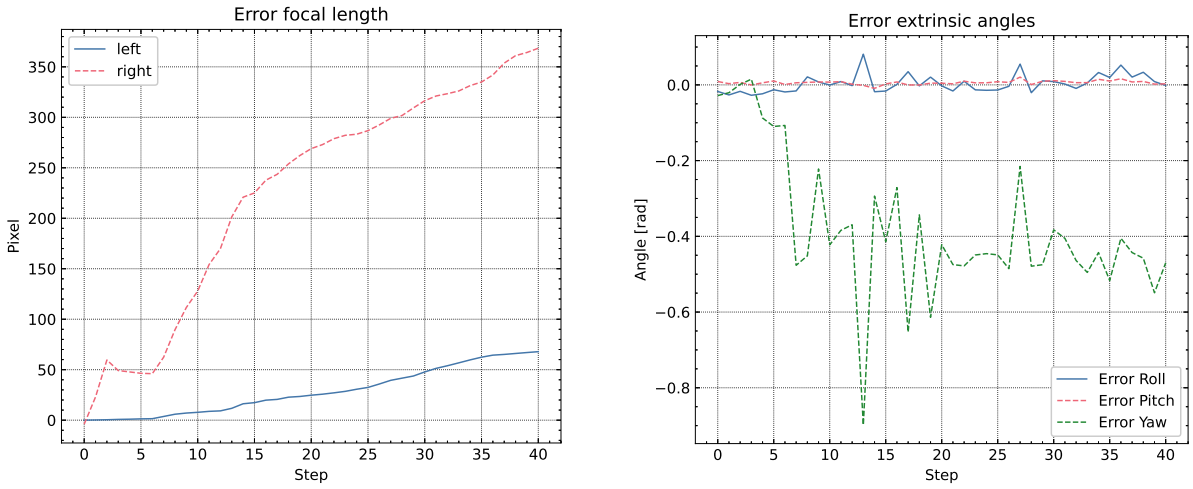


Figure 6.14: Errors in state estimated in the far-SIFT scenario.

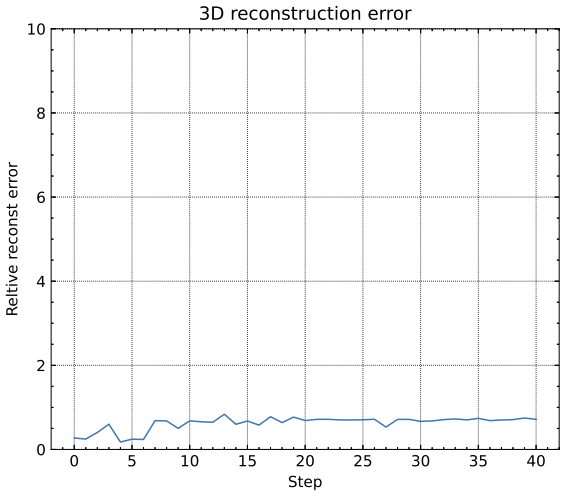


Figure 6.15: 3D reconstruction error for far-SIFT scenario.

Variable	Average Error
θ	0.003
ϕ	0.006
ψ	-0.379
f_L	28.574
f_R	225.53
3D rec	0.624

Table 6.15: Average errors from far-SIFT scenario.

6.2.8 Far Run with ORB as Descriptor

θ	ϕ	ψ	f_L	f_R
-0.096	0.010	1.408	1062.854	925.707

Table 6.16: Final values for calibration parameters after the far-ORB scenario.

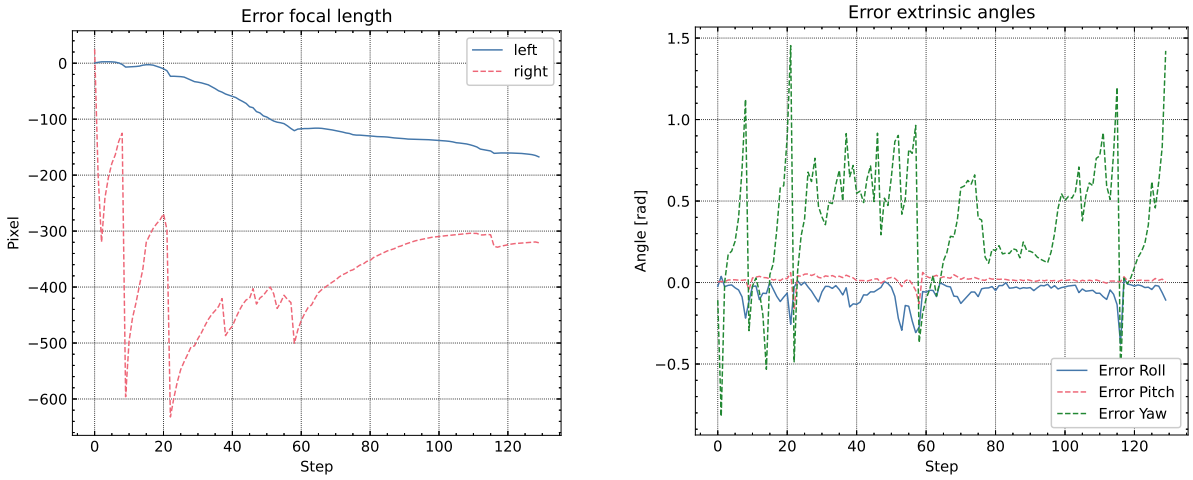


Figure 6.16: Errors in state estimated in the far-ORB scenario.

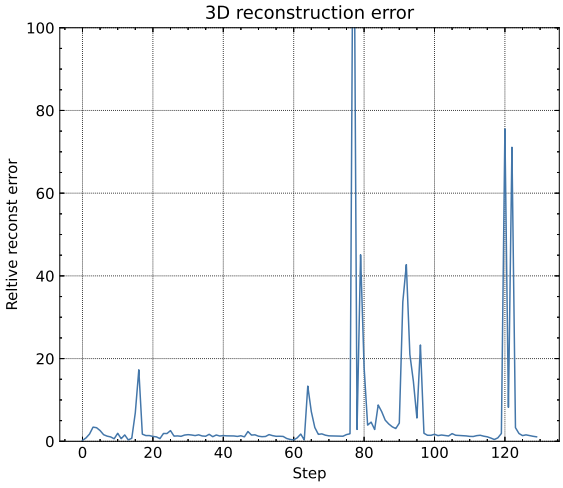


Figure 6.17: 3D reconstruction error for far-ORB scenario.

Variable	Average Error
θ	-0.063
ϕ	0.017
ψ	0.378
f_L	-94.418
f_R	-368.199
3D rec	5.776

Table 6.17: Average errors from far-ORB scenario.

6.3 Runtime

Below is the runtime of the different scenarios listed. The timing was performed using the C++ library chrono [7].

Feature method	Matching time [ms]	EKF time [ms]	Total time [ms]
Close SIFT	498.052	188.230	686.282
Close Orb	239.641	353.751	593.392
Intermediate SIFT	429.745	63.301	493.047
Intermediate Orb	227.473	164.338	391.811
Far SIFT	547.919	2409.520	2957.439
Far Orb	238.274	846.290	1084.564

Table 6.18: Runtimes of the auto-calibration algorithm.

Chapter 7

Discussion

At first glance of the results presented in chapter 6, it is clear to see that self-calibrating algorithm didn't perform at a satisfactory level. The results varied very during all the scenarios and none of them managed to recreate values close to the ground truth. In this chapter some observations from the results are discussed.

7.1 Feature Matching Performance

Figure 6.1 shows how well the RANSAC algorithm worked in order to remove outliers from the matches. The chaos from all the matches before RANSAC is sorted out. An indication that the matches are correct is observing that every match line is fairly parallel. Since the images are distorted some deviation from perfectly parallel lines are expected, but a mismatch would be easily visible. The RANSAC used in the algorithm was strict, reducing the number of matches found two-point matches quite drastically accepting only down to 10% of all the matches.

During all three scenarios a decent number of three-point matches was detected no matter which feature methods was chosen. The biggest difference between the three scenarios is how high distribution of the matches there are over the images. Figure 6.4 shows that all the matches are found between a small height interval in the image in the far run. It is impossible to find matches in the sea and sky, resulting in a big blind spot for the cameras in this scenario. When the ferry moves closer to the buildings, fig. 6.2 and fig. 6.3 shows that the feature coverage becomes better and better. But even in the close run there are areas of the image where no matches are found especially in the lower part of the images. A good distribution of the features over the entire image is important to pick up on all the attributes of the camera. Distortion effect is al-

ways biggest away from the images centre, so to be able to estimate the distortion parameters it is important to have key points in the extremities of the picture. There are techniques of distributing the matches uniformly over the entire image, such as the grid method presented by Bergmann et al. [3].

There was no real difference in number of matches found when using FLANN over brute force. This is partly due to the strict RANSAC filtering. Bf turned out to be slightly faster than FLANN in some of the scenarios, which was a bit surprising. This may be due to relatively low number of features found in the images [24]. Since there were no real performance differences between the matching methods, it was decided to only use the more sophisticated FLANN methods when testing the full system.

The choice of feature descriptor had a big impact on the number of matches detected, and the algorithms runtime. SURF delivered the most amount of matches in both the close and far run, but also had the highest calculation time, with the runtime being almost three times that of ORB in the close run. ORB was the fastest of the three, while outscoring SIFT on matches in both the close and intermediate scenario. SIFT delivered close to the same number of matches as SURF in the far scenario, but the least amount in the two other runs. Its runtime in the close and intermediate time was lower then SURF's. Due to the runtime of SURF on the close and intermediate scenarios, the decision of dropping this descriptor in further testing was made. Considering the solid amount of matches found by ORB and its low runtime, it was arguably the best descriptor of the three for the self-calibrating algorithm.

7.2 Auto-Calibration Accuracy

The performance of the auto-calibration algorithm was evaluated by comparing it to an offline calibration. A reprojection error for all the images from the offline calibration was calculated and is presented in fig. 6.5. The overall mean error is of only 0.04 pixels which are pretty low. Hirschmüller and Gehrig suggests that an error below 0.25 pixels is an acceptable accuracy [21]. It was therefore concluded that the offline calibration was reliable.

As is visible from the final states from all the six different runs, the auto-calibration algorithm failed to replicate the values from the reference even though the initial state (eq. (6.5)) was set very close to the solution. The most notable error was for the extrinsic angles. Three out of the six runs resulted in positive ψ-angles, suggesting that the right camera is pointing in the opposite direction. By looking at the development of the ψ error over time, it is not possible to conclude that the parameter is converging to a value in any of the runs. The final value of ψ thus

depend heavily on when the algorithm is terminating, which is a very bad estimation. The very varying ψ resulted in very unstable 3D reconstruction errors in all the runs except for the far scenario with SIFT. This scenario has quite a few less Kalman steps then the others because of the high calculation time of the EKF. The high runtime is most likely caused by the high number of matches found during this run. A max number of matches should have been implemented in order to prevent the algorithm from exceeding more than a second of runtime per iteration.

7.3 EKF as an Optimizer

Since the accuracy of the auto-calibration algorithm was so inadequate it is fitting to question if an Extended Kalman Filter is the best optimizer for the proposed problem. The optimization problem is very complicated, with a very non-linear cost function and where every constraint is multi linear.

Dang et al. discussed in their paper the importance of a good initial guess for their algorithm to succeed, underlining the complexity of the optimization problem. They demonstrated that this method worked on their system [8]. Even with a very close initial state the EKF still failed on the stereo system in this thesis. A lot of hours went into trying to tune the noise and variance in order to produce the results presented in section 6.2. The EKF was found to be very unstable and it didn't take much deviation from the values in section 6.2.2 before the state diverged to infinity.

Such complicated optimization problems as this may demand a more sophisticated method. One of the biggest weaknesses of the EKF is the linearization of the non-linearities in the system around the current state and measurements. If this approximation is inaccurate, the optimization will fail. Skoglund et al. argued that EKF is just a special case of a Gauss-Newton optimizer, where the step length is kept constant. They presented a moderation to the EKF which included line search to find the optimal step length. They also discuss the Levenberg-Marquardt-EKF [40]. All of these methods make for a more dynamic optimization, and could deal better with the oscillations of the ψ-angle in the model. An implementation of one their suggested methods would likely therefor yielded more successful results for the plane EKF method managed.

Chapter 8

Conclusion

An auto-calibration algorithm was proposed in this report. It was built as a ROS-node which can easily be transferred to milliAmperes OBC. The algorithm has the infrastructure to support multiple feature descriptors and matching methods, making for a flexible system. A calibrator based on reduction of the reprojection error was implemented. Reprojection error minimizing is an optimization problem, and it was tried solved using an Extended Kalman Filter. Epipolar and trifocal constraints was added to the problem to ease the calculation complexity.

In order to utilize the trifocal constraint, the system needed three-point matches. The third view was found using the image from the left camera of the neighbouring epoch. A simple three-point match finder was introduced where first two-view matches was found between the common current left image and the two other images. A three-point match could then be found by a linear search of the two matching lists to find common key points in the common image. RANSAC was used in the two-view matching to eliminate outliers. There was not much difference in performance between the two matchers FLANN and brute force in this application. ORB proved to be the fastest descriptor and delivered a solid amount matches. Considering that image processing is a big part of the total runtime of the algorithm, it was argued that Orb is the best descriptor choice for the auto-calibration algorithm.

From the calibration results it was obvious that auto-calibration algorithm did not perform at the desired level. The EKF optimizer was very unstable, and the initial values in the filter proved to be hard to tune. A lot of camera model simplifications were done in an effort to reduce the problem complexity, but still the estimations were incorrect. This led to the conclusion that the EKF was not an optimal solver for the complex optimization. Alternative optimizers that can be explored in the future was discussed in section 7.3.

Even though the algorithm failed to deliver good results, this report will be a good starting point for anyone trying to create a self-calibrating stereo vision system in the future. The code developed in this project has all the framework for ROS interface and feature matching in place. The report discusses methods of securing enough structure in images in a marine environment, and all the data gathered at sea will be made available for future projects.

Bibliography

[1] Herbert Bay. *SURF website*. 2008. URL: https://people.ee.ethz.ch/~surf/index.html.

[2] Herbert Bay et al. "Speeded-Up Robust Features (SURF)". In: *Computer Vision and Image Understanding* 110.3 (2008), pp. 346–359. ISSN: 10773142. DOI: 10.1016/j.cviu.2007.09.014.

[3] Paul Bergmann, Rui Wang, and Daniel Cremers. "Online Photometric Calibration of Auto Exposure Video for Realtime Visual Odometry and SLAM". In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 627–634. ISSN: 23773766. DOI: 10.1109/LRA.2017.2777002.

[4] José Luis Blanco-Claraco. "A tutorial on SE(3) transformation parameterizations and on-manifold optimization". In: 3 (2021). URL: http://arxiv.org/abs/2103.15980.

[5] Univesity of British Colombia. *The SIFT Keypoint Detector*. 2020. URL: https://www.cs.ubc.ca/~lowe/keypoints/.

[6] Carlos Campos Martínez et al. "ORB-SLAM3: An accurate Open-source library for visual, Visual-inertial and Multi-map SLAM". In: *arXiv* (2020), pp. 1–15.

[7] CPPreference. *Date and time utilities*. URL: https://en.cppreference.com/w/cpp/chrono.

[8] Thao Dang, Christian Hoffmann, and Christopher Stiller. "Continuous stereo self-calibration by camera parameter tracking". In: *IEEE Transactions on Image Processing* 18.7 (2009), pp. 1536–1550. ISSN: 10577149. DOI: 10.1109/TIP.2009.2017824.

[9] Edmund Optics. *8mm UC Series Fixed Focal Length Lens*. 1992. URL: https://www.edmundoptics.com/p/85mm-c-series-fixed-focal-length-lens/14947/.

[10] O Egeland and J T Gravdahl. *Modeling and Simulation for Automatic Control*. JANUARY 2002. 2002. ISBN: 9788292356012. URL: https://books.google.com/books?id=oK0VAAAACAAJ.

[11] FLIR Systems Inc. *Spinnaker SDK*. 2021. URL: https://www.flir.com/products/spinnaker-sdk/.

[12] FLIR Systens inc. *Blackfly S GigE*. URL: https://www.flir.com/products/blackfly-s-gige/?model=BFS-PGE-50S5C-C.

[13] Henri P. Gavin. "The Levenburg-Marqurdt Algorithm For Nonlinear Least Squares Curve-Fitting Problems". In: *Duke University* (2019), pp. 1–19. URL: http://people.duke.edu/~hpgavin/ce281/lm.pdf.

[14] Nilesh S Gopaul. "Optimal image-aide intertial navigation". In: *A dissertation submitted to the faculty of graduate studies in partial fulfillment of the requirement for the degree of doctor of philosophy graduate* August (2018).

[15] Nilesh S Gopaul, Jianguo Wang, and Baoxin Hu. "Camera auto-calibration in GPS/INS/stereo camera integrated kinematic positioning and navigation system". In: *The Journal of Global Positioning Systems* 14.1 (2016), p. 3. ISSN: 1446-3164. DOI: 10.1186/s41445-016-0003-7. URL: https://doi.org/10.1186/s41445-016-0003-7.

[16] Martin Græsdal. "Self-calibration of stereo vision for autonomous ferry". In: (2020).

[17] Banglei Guan, Yang Shang, and Qifeng Yu. "Planar self-calibration for stereo cameras with radial distortion". In: *Applied Optics* 56.33 (2017), p. 9257. ISSN: 1559-128X. DOI: 10.1364/ao.56.009257.

[18] L. Gueguen and M. Pesaresi. "Multi scale Harris corner detector based on Differential Morphological Decomposition". In: *Pattern Recognition Letters* 32.14 (2011), pp. 1714–1719. ISSN: 01678655. DOI: 10.1016/j.patrec.2011.07.021. URL: http://dx.doi.org/10.1016/j.patrec.2011.07.021.

[19] Chris Harris and Mike Stephens. "A combied corner and edge detector". In: *Jahrbücher für wissenschaftliche Botanik* 69 (1988), pp. 762–818. ISSN: 09639292.

[20] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd editio. Cambridge University Press, 2004.

[21] Heiko Hirschmüller and Stefan Gehrig. "Stereo matching in the presence of sub-pixel calibration errors". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009* 2009 IEEE (2009), pp. 437–444. DOI: 10.1109/CVPRW.2009.5206493.

[22] David G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. ISSN: 09205691. DOI: 10.1023/B:VISI.0000029664.99615.94.

[23] Alexander Mordvintsev and K Abid. *Feature Matching*. 2013. URL: https://docs.opencv.org/4.5.2/dc/dc3/tutorial_py_matcher.html.

[24] Marius Muja and David G. Lowe. "Fast approximate nearest neighbors with automatic algorithm configuration". In: *VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications* 1 (2009), pp. 331–340. DOI: 10.5220/0001787803310340.

[25] Basam Musleh et al. "Pose self-calibration of stereo vision systems for autonomous vehicle applications". In: *Sensors (Switzerland)* 16.9 (2016). ISSN: 14248220. DOI: 10.3390/s16091492.

[26] Neufieldrobotics. *spinnaker_sdk_camera_driver*. URL: https://github.com/neufieldrobotics/spinnaker_sdk_camera_driver.

[27] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006. ISBN: 9780387303031. DOI: 10.1007/978-0-387-40065-5.

[28] NTNU. *Autoferry*. URL: https://www.ntnu.edu/autoferry.

[29] Open Robotics. *How to Calibrate a Monocular Camera*. 2019. URL: http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration.

[30] Open Robotics. *How to Calibrate a Stereo Camera*. 2018. URL: http://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration.

[31] Open Robotics. *Messsage Filter*. 2018. URL: http://wiki.ros.org/message_filters.

[32] OpenCV. *About OpenCV*. URL: https://opencv.org/about/.

[33] OpenCV. *FAST Algorithm for Corner Detection*. URL: https://docs.opencv.org/3.4/df/d0c/tutorial_py_fast.html.

[34] OpenCV. *ORB (Oriented FAST and Rotated BRIEF)*. URL: https://docs.opencv.org/master/d1/d89/tutorial_py_orb.html.

[35] Kaare Brandt Petersen and Michael Syskind Pedersen. "The Matrix Cookbook". In: 8 (2012), pp. 1–30. ISSN: 18662617. DOI: 10.1007/978-3-662-45664-4{_}1.

[36] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *Computer Science Department, Stanford University* (2009).

[37] J A Ramos. "A Kalman-tracking filter apporach to nonlinear programming". In: 19.11 (1990), pp. 63–74.

[38] Eike Rehder et al. "Online stereo camera calibration from scratch". In: *IEEE Intelligent Vehicles Symposium, Proceedings* Iv (2017), pp. 1694–1699. DOI: 10.1109/IVS.2017.7995952.

[39] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[40] Martin A. Skoglund, Gustaf Hendeby, and Daniel Axehill. "Extended Kalman filter modifications based on an optimization view point". In: *2015 18th International Conference on Information Fusion, Fusion 2015* (2015), pp. 1856–1861.

[41] Robert F. Stengel. *Optimal State Estimation Kalman, Hinf, and Nonlinear Approaches*. 1994, pp. 299–419. ISBN: 9780471708582. URL: https://books.google.it/books?hl=zh-CN&lr=&id=UiMVoP_7TZkC&oi=fnd&pg=PR3&dq=Optimal+State+Estimation+Kalman&ots=L0Li8GHiCn&sig=HwTX5Tcv0E8JHyAZHmkWYeIu69c#v=onepage&q=Optimal%20State%20Estimation%20Kalman&f=false.

[42] Peter Sturm. "Pinhole Camera Model BT - Computer Vision: A Reference Guide". In: ed. by Katsushi Ikeuchi. Boston, MA: Springer US, 2014, pp. 610–613. ISBN: 978-0-387-31439-6. DOI: 10.1007/978-0-387-31439-6{_}472. URL: https://doi.org/10.1007/978-0-387-31439-6_472.

[43] Zhongwei Tang et al. "A Precision Analysis of Camera Distortion Models". In: *IEEE Transactions on Image Processing* 26.6 (2017), pp. 2694–2704. ISSN: 10577149. DOI: 10.1109/TIP.2017.2686001.

[44] Line Charlotte Kristoffersen Theimann and Trine Ødegård Olsen. "Stereo vision for autonomous ferry". In: *Master's Thesis* (2020).

[45] Deepanshu Tyagi. *Introduction to SURF (Speeded-Up Robust Features)*. 2019. URL: https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e.

[46] Qi Wang, Qin Zhang, and Francisco Rovira-MÁS. *Auto-calibration method to determine camera pose for stereovision-based off-road vehicle navigation*. 2010. DOI: 10.2525/ecb.48.59.

[47] Christian Wöhler. *3D Computer Vision*. Vol. 4. 1. 2016, pp. 64–75. ISBN: 9781447141495. DOI: https://doi.org/10.1007/978-1-4471-4150-1.

[48] Shuo Zhang et al. "Self calibration of the stereo vision system of the Chang'e-3 lunar rover based on the bundle block adjustment". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 128 (2017), pp. 287–297. ISSN: 09242716. DOI: 10.1016/j.isprsjprs.2017.04.004. URL: http://dx.doi.org/10.1016/j.isprsjprs.2017.04.004.

[49] Shuo Zhang et al. "Self-calibration of the stereo vision system of the chang'e-4 lunar rover based on the points and lines combined adjustment". In: *Photogrammetric Engineering and Remote Sensing* 86.3 (2020), pp. 169–176. ISSN: 00991112. DOI: 10.14358/PERS.86.3.169.

[50] Zhengyou Zhang. *A Flexible New Technique for Camera Calibration*. Tech. rep. 1998.

Appendix A

Acronyms

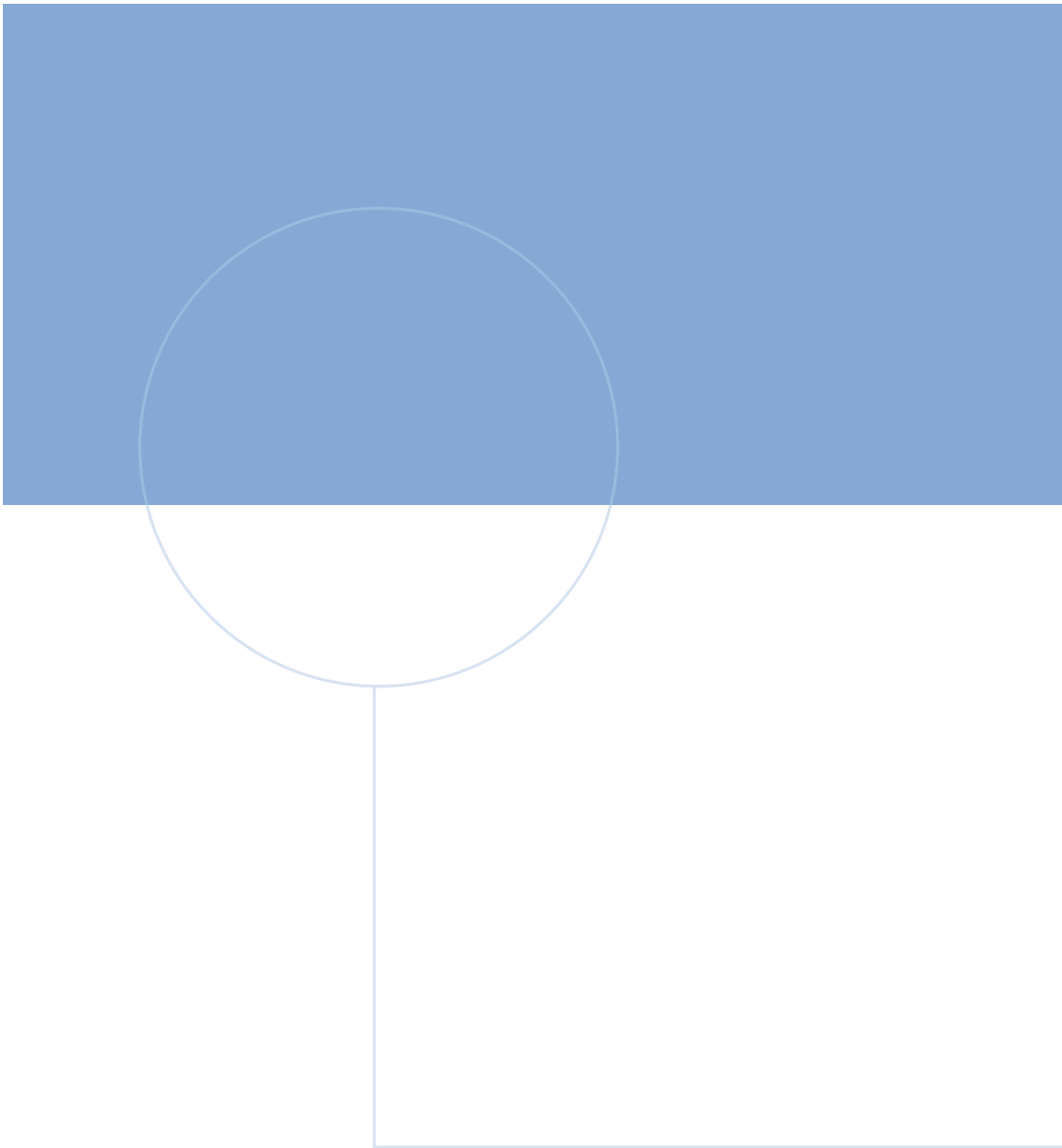
BF	Brute Force. Matching method.
BRIEF	Binary Robust Independent Elementary Features
CCD	Charge-Coupled Device. Sensor in digital imaging
DCM	Direction Cosine Matrix
DoG	Difference of Gaussian
EKF	Extended Kalman Filter
FAST	Features from Accelerated Segment Test
FLANN	Fast Library for Approximating Nearest Neighbor
FoV	Field of View
GNSS	Global Navigation Satellite Systems
GPIO	General Purpose Input/Output
IMU	Inertial Measurement Unit
LiDAR	Light Detection And Ranging
LM	Levenberg-Marquardt optimizer
NED	North East Down. Standard coordinate frame
OBC	On Board Computer
ORB	Oriented FAST and Rotated BRIEF
PoE	Power over Ethernet
RANSAC	RANdom SAmple Consensus
ROS	Robot Operating System
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SURF	Speeded up robust features
SSD	Sum of Squared Differences

Appendix B

Rostopics from data gathering

Topic name	Message type
/camera_array/left/image_raw	sensor_msgs/Image
/camera_array/right/image_raw	sensor_msgs/Image
/IR/F/image_raw	sensor_msgs/Image
/IR/FL/image_raw	sensor_msgs/Image
/IR/FR/image_raw	sensor_msgs/Image
/IR/RL/image_raw	sensor_msgs/Image
/IR/RR/image_raw	sensor_msgs/Image
/sensor_rig/optical/F/image_raw	sensor_msgs/Image
/sensor_rig/optical/FL/image_raw	sensor_msgs/Image
/sensor_rig/optical/FR/image_raw	sensor_msgs/Image
/sensor_rig/optical/RL/image_raw	sensor_msgs/Image
/sensor_rig/optical/RR/image_raw	sensor_msgs/Image
/camera_array/left/camera_info	sensor_msgs/CameraInfo
/camera_array/right/camera_info	sensor_msgs/CameraInfo
/IR/F/camera_info	sensor_msgs/CameraInfo
/IR/FL/camera_info	sensor_msgs/CameraInfo
/IR/FR/camera_info	sensor_msgs/CameraInfo
/IR/RL/camera_info	sensor_msgs/CameraInfo
/IR/RR/camera_info	sensor_msgs/CameraInfo
/sensor_rig/optical/F/camera_info	sensor_msgs/CameraInfo
/sensor_rig/optical/FL/camera_info	sensor_msgs/CameraInfo
/sensor_rig/optical/FR/camera_info	sensor_msgs/CameraInfo

/sensor_rig/optical/RL/camera_info	sensor_msgs/CameraInfo
/sensor_rig/optical/RR/camera_info	sensor_msgs/CameraInfo
/actuator_ref_1	custom_msgs/ActuatorSetpoints
/actuator_ref_2	custom_msgs/ActuatorSetpoints
/actuators/actuator_1/azimuth_angle	std_msgs/Float64
/actuators/actuator_1/thruster/motor_state	custom_msgs/MotorState
/actuators/actuator_1/thruster/system_setup	actuators/TorqueedoSystemSetup
/actuators/actuator_2/azimuth_angle	std_msgs/Float64
/actuators/actuator_2/thruster/motor_state	custom_msgs/MotorState
/actuators/actuator_2/thruster/system_setup	actuators/TorqueedoSystemSetup
/docking_hatch/distance_aft	docking_hatch_driver/Distance
/docking_hatch/distance_fore	docking_hatch_driver/Distance
/dynamic_positioning/control_action	custom_msgs/ThreeDofForce
/navigation/nav_estimate	custom_msgs/NavigationEstimate
/navigation/pose	geometry_msgs/PoseStamped
/navigation/twist_body	geometry_msgs/TwistStamped
/navigation/twist_ned	geometry_msgs/TwistStamped
/rc_state	custom_msgs/RemoteControlState
/supervisor/mode	std_msgs/String
/tf	tf2_msgs/TFMessage
/vectorVS330/GPGGA	custom_msgs/rawGPSdata
/vectorVS330/NMEA	custom_msgs/rawGPSdata
/vectorVS330/fix	custom_msgs/gnssGGA
/vectorVS330/heading	custom_msgs/gnssHDT
/vectorVS330/nmea_sentence	nmea_msgs/Sentence
/vectorVS330/velocity	custom_msgs/gnssRMC
/velodyne_points	sensor_msgs/PointCloud2
/xsens/imu	sensor_msgs/Imu
/xsens/orientation	custom_msgs/orientationEstimate



NTNU

Kunnskap for en bedre verden