NTNU

Self-Calibration of Stereo Vision for Autonomous Ferry

Martin Græsdal

December 2020

TTK4551 - Specialization Project (7.5p)

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Supervisor 1: Edmund Førland Brekke

Supervisor 2: Annette Stahl

Supervisor 3: Øystein Kaarstad Helgesen

Preface

This report was written as part of the specialization project in the second year of my two-year Master's program in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU) in Trondheim. The goal for the 7.5 point subject is to gain experience in a chosen area of study through litterateur search and scientific methods. This project focused on stereo vision and calibration of such a system. The work was carried out during the autumn of 2020.

The project was proposed and supervised by Edmund Førland Brekke, and together with the co-supervisors Annette Stahl and Øystein Kaarstad Helgesen, he provided a lot of help and support for which I would like to thank. The project was part of the research project Autoferry, whose aim is to develop an electric autonomous ferry for urban public transport using exciting new technology. As part of this project I got hands on experience with the ferry "milliAmpere", and I would like to thank Egil Eide for the giving me that opportunity.

As part of this projects some experiments where conducted in to gather data set for the stereo system. This work was carried out alongside Kristian Auestad who specializes in depth estimation using stereo vision, and the experiments presented in chapter 4 of this report.

Trondheim, 23.12.2020

Martin Græsdal

Abstract

Stereo vision is a technique where multiple camera of a known configuration is used to observe a common scene that can be used to deliver situational awareness to robots. In order for a stereo vision system to deliver usable results, the system needs to be well calibrated.

This project studies a way of performing camera calibration without using a calibration rig. The aim is to find a method that can make use of image series and data gathered during normal operation of the autonomous ferry milliAmpere to reduce the need for the ferry to be put out of service in order to perform calibration. An algorithm using 3-view matching and Scale Restrain Equations is proposed, estimating both the extrinsic and intrinsic camera parameters as well as radial distortion using a Levenberg-Marquardt algorithm. The calibration is performed offline using real data from the ferry. Results shows some promising signs, but during testing some problems occurred, meaning that further development of the algorithm is need. Some suggested improvements are presented.

In addition to developing the calibration method, experiments were conducted using milliAmpere. The goal of these experiments was to gather data during different scenarios in order to have realistic data to test methods on in the future. The experiments resulted in more then 4000 stereo image pairs totaling over 8000 images in multiple lighting condition.

Contents

Chapter 1

Background

For a robot to be able to safely move around in an environment, situational awareness is of utmost importance. In autonomy there is no operator to observe and interact with the environment, leaving the computers to do that work. Today the most common sensors used to convert to physical scene into information that the computer can use is radars and LiDAR. They are active sensors that measures the distance to surrounding object, by emitting a a type of signal and measures how long time it takes before the signal returns. While these sensors are thoroughly researched and well renowned, they have some issues. The biggest one might be the price tag which makes the unavailable to some users.

Lately, systems based on cameras have gathered a lot of momentum. They offer the sensors which are a lot cheaper. Previously the computational power required to process the input from cameras have made these systems unpopular. But as the computational resources have increased, the field of computer vision has grown.

For cameras to be able to capture the depth in a scene, the same scene must be viewed from different angles. Stereo vision is a technique where multiple cameras are utilized. The baseline and angle between each camera are known. By using triangulation, 2D points in the image can be transformed to 3D in a world coordinate frame, creating a digital representation of the observed scene.

In order for a stereo vision system to be accurate, a good calibration needs to be performed. There are two types of variables to be calibrated: intrinsic and extrinsic. The intrinsic parameters are individual for each camera and determines how a camera interprets the images. Extrinsic parameters are rotation and translation between the cameras. These parameters are estimated using calibration algorithms.

1.1 Problem Formulation

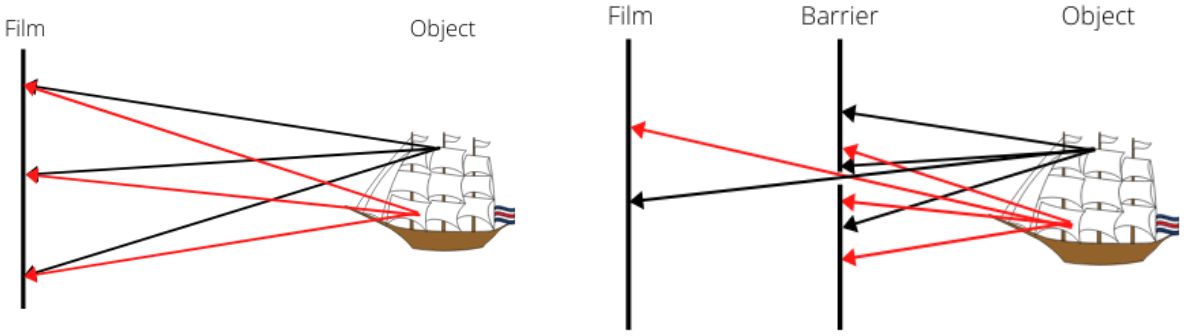
This project aims to implement a robust calibration algorithm for a stereo vision system to be used on an autonomous ferry which is going to operate in the canal of Trondheim. Many calibration algorithms are based on prior knowledge about the scene or movement of the system [30]. These methods requires that the ferry would have to be put out of service every time the cameras needs to recalibrate. Therefor a self-calibrating method that can be performed on data gather during normal operation is desired. This would increase the ferry's operational time. Additionally, the writer aims to gain insight into to the field of computer vision, as well as getting familiar with software and hardware used on milliAmpere and on stereo rig in order to lay a solid foundation for the master project.

Chapter 2

Theory

2.1 Camera Model

The pinhole camera model is used to translate 3D points in the scene to 2D points in the image plane. It is based on the 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 called the film, in which the image is captured. An object in the real world will reflect light in every direction. The small size of the hole will filter the light, making sure that the light emitted from a point in the scene will only enter the box from one direction. [26]



(a) Light beams from the scene

(b) Filtering by the pinhole.

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

The image plane is defined on the photosensitive surface. Because of the directed light, the scene will appear inverted on the image plane. In a real camera the light is permitted through

a wider opening than a pinhole and processed through a series of lenses to reduce exposure time and still have a focused image. The lenses are reduced to 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 that passes through the projection centre is the optical axis, and the point where this originates on the image plane is the principal point. To simply visualization of the image plane and its geometry, it is common 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 the 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 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 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 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) [13]. 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[26].

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

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 are zero, each camera usually have four intrinsic parameters to be estimated.

Extrinsic parameters

The extrinsics are the state 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.

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

The rotational matrix uses Euler angles in the sequence of roll-pitch-yaw[5]. 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 [26]. There are two types of lens distorting effects: Radial and tangential distortion

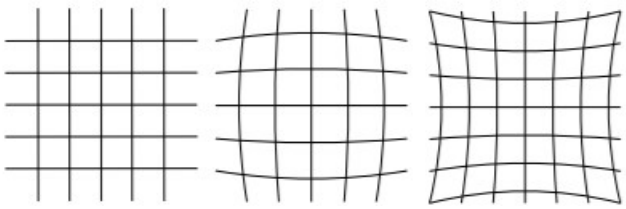


Figure 2.2: Radial distortion [10]

Radial distortion leads to straight lines in the image appearing bent. The effect is more apparent along the edges and in the corners of image. Figure 2.2 illustrates this effect in a 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 [29]:

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

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

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 [27]

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

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

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 are 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 in to 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 [13]. While some of the classical methods are well renowned, the self-calibration methods need to be tailored to the situation of appliance.

Zhang's method

Zhang's method is a commonly used calibration method. It is a hybrid between classical and self-calibration. To 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. Further the parameters are refined by using a Levenberg-Marquardt algorithm to reducing a reprojection error [32].

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 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[20]. The method has a good convergence rate if the function is simple, but it may struggle a bit more if it becomes complicated. As a way of improving upon this method different strategies of choosing better search direction is proposed [7].

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, which allows 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 functions are complicated and there are a lot of residuals to consider. The hessian is therefore approximated by the jacobian squared. Equation (2.16) are used to calculate the search direction $\boldsymbol{p}_k$ [20].

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

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

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 [7].

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. (2.16), a damping factor λ are added to adjust the search [20].

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

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 are big it will dominate the hessian approximation term, and the search direction are 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 of converge from the Gauss-Newton [7].

2.3 Features

In direct methods a set of features in the frames are extracted to give a sparse representation of the scene. There are a lot of different techniques to extract and describe these features. The features lay the foundation for a lot of methods such as object recognition and SLAM. This report uses feature matching in order to calibrate the cameras.

2.3.1 Harris Corner

Harris corner is detection method which looks at the intensity in the image to find edges and corners. Around every pixel in the image a window W are selected. By shifting this this window slightly, 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 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 looking on how the energy function changes as the window moves, an indication of the structure is exposed. If there are no changes, the pixel is 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 in every direction. In order to translate this into a mathematical formula, Harris came up with a response function [12].

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

Where

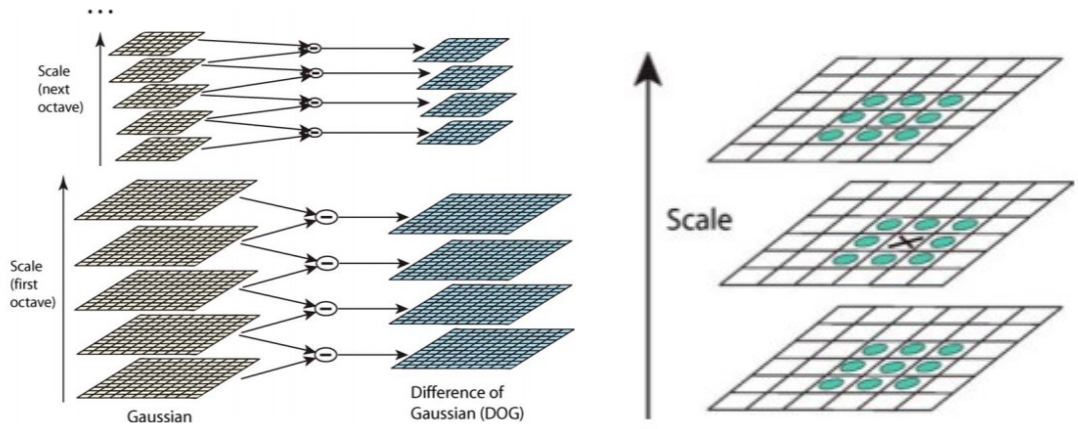
$$\det(\boldsymbol{A}) = \alpha\beta, \;\; \operatorname{trace}(\boldsymbol{A}) = \alpha + \beta \tag{2.20}$$

A high positive R indicates a corner, while high negative values indicate edges. If R is a small number, it is considered a flat area. κ is scalar value which needs to be chosen. It typically lies between $[0.04, 0.15]$. α and β are the eigenvalues of the $\boldsymbol{A}$ [11].

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 [15]. The key expression comes at a price as the algorithm are quite computationally heavy.



(a) Creation of Difference of Gaussian (DoG)

(b) Comparing candidates with neighbors

Figure 2.3: SIFT
Illustrations courtesy of David Lowe [15]

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 in the image. This is done using Difference of Gaussians. A candidate is detected by comparing it with its eight neighbours 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 are executed over several layers of sampling, re-sampling and smoothing the image by a factor of 2 [15].

The second step is to go over all the candidates and reject unsatisfactory key-points. Candidates that are rejected are those with low contrast or are 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 are determined through a histogram where every sample give a weighted vote for a direction. The direction with the most votes is selected as the orientation for the key point. If there are 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 [15].

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 subregions. Every subregion vote 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 [1].

2.3.3 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, utilizing the best part from both. The key points are found using FAST, and then the bad key points are filtered out using Harris corner. The features are described using an improved version of BRIEF, where the orientation of the key point is calculated efficiently [24]. The result is robust feature detector which has a lot of the same properties as the SIFT, but a lot less computational power is required [22]. 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 VSLAM method ORB-SLAM is based on this descriptor [2].

2.4 Feature matching

After collecting the features in one image, it often is desired find the same key point in another image. This is done thought feature matching. There are many ways of performing matches, but only two methods are available in OpenCVs library. These methods are presented below.

2.4.1 Brute force

The Brute-Force matcher (BF) is as the name implies a simple matcher. The matcher tries to match a key point in one image with every key point in the other image. A distance is calculated between every match, 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 favored [16]. Since every feature gets a match, no matter how bad, it is important to have a threshold of an acceptable distance in order 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 stands for Fast Library for Approximating Nearest Neighbours. The matcher is a collection of different algorithms that solves the nearest neighbours problem by applying ether randomized kd-trees or hierarchical k-means trees to search for solutions. 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 [17]. The main drawback with working with search trees is the fact that even though a key point gets a match, but there is no way of guaranteeing that the best available match is selected. This might result in more outliars or mismatches than BF.

2.5 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 relationship between them can be utilized to acquire knowledge about the world coordinates.

2.5.1 Two-view geomtry

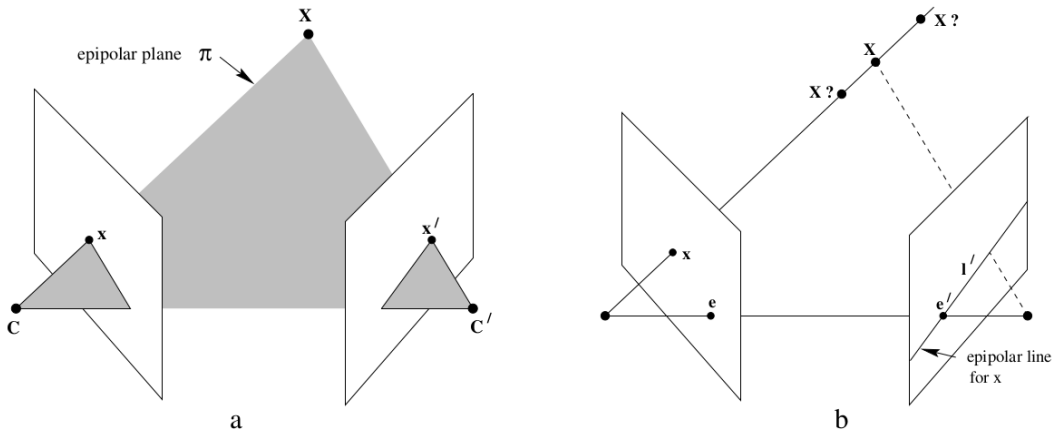


Figure 2.4: Epipolar geometry
Courtesy of Hartley and Zisserman [13]

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 $\boldsymbol{C}'$, intersecting 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 $\boldsymbol{\pi}$. [13]

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 correspondence $\boldsymbol{x}'$ must be located on this epipolar line, hence reducing the matching problem quite drastically.

The algebraic representation of the epipolar geometry is called the fundamental matrix $\boldsymbol{F}$. A special case of the fundamental matrix called the essential matrix $\boldsymbol{E}$ relates fundamental matrix with camera matrices $\boldsymbol{K}$ and $\boldsymbol{K}'$.

$$\boldsymbol{E} = \boldsymbol{K}'\boldsymbol{F}\boldsymbol{K}. \tag{2.21}$$

2.5.2 Three-view geometry

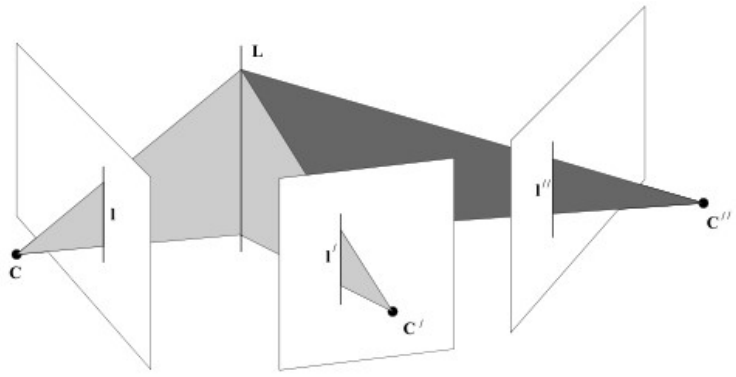


Figure 2.5: Three-view geometry
Courtesy of Hartley and Zisserman [13]

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. The three cameras leave 18 degrees of freedom, meaning that the tensor can algebraically be calculated with these constraints [13].

Scale Restraint Equations

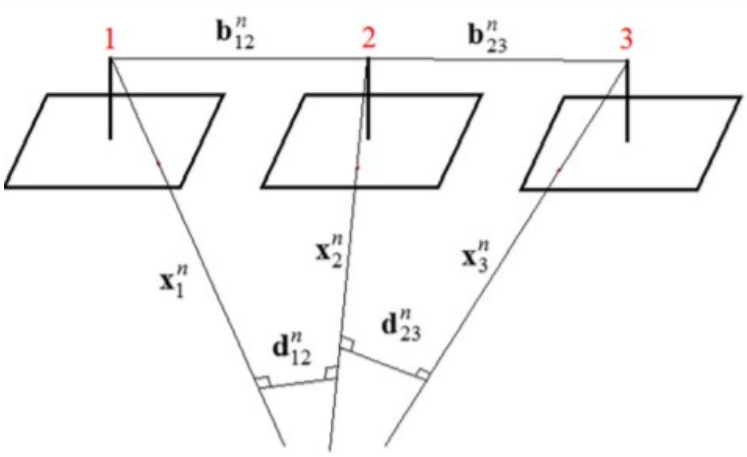


Figure 2.6: 3-view Scale restraint
Courtesy of Gopaul [9]

The Scale Restraint Equation (SRE) is an equation that utilizes a restraint forcing the three cameras to operate on the same scale. If image 1 and 2 are relatively orientated to each other and

image 2 and 3 to each other, there is no guaranteeing that image 1 and 3 are the same. Due to the scaling factor, this can lead to the three image point vectors $\boldsymbol{x_1^n}, \boldsymbol{x_2^n}$ and $\boldsymbol{x_3^n}$ not intersecting at a common point in the scene. The relationships from fig. 2.6 are used created these equations:

$$\begin{aligned} k_1\mathbf{x}_1^n - k_2\mathbf{x}_2^n + k_{12}\mathbf{d}_{12}^n &= \mathbf{b}_{12}^n \\ k_2'\mathbf{x}_2^n - k_3\mathbf{x}_3^n + k_{23}\mathbf{d}_{23}^n &= \mathbf{b}_{23}^n \end{aligned} \tag{2.22}$$

Where $k_1, k_2, k_{12}, k_2', k_3'k_{23}'$ are unknown scaling factors. In order for the cameras to operate on a common scale the two scaling factors of the common image 2 must be equal. That is $k_2 - k_2' = 0$. k_2 and k_2' are defined as:

$$k_2 = \frac{\mathbf{x}_1^n \diamond \mathbf{d}_{12}^n \times \mathbf{b}_{12}^n}{\mathbf{x}_1^n \diamond \mathbf{d}_{12}^n \times \mathbf{x}_2^n}, \quad k_2' = \frac{\mathbf{b}_{23}^n \diamond \mathbf{d}_{23}^n \times \mathbf{x}_3^n}{\mathbf{x}_2^n \diamond \mathbf{d}_{23}^n \times \mathbf{x}_3^n} \tag{2.23}$$

$$\frac{\mathbf{x}_1^n \diamond \mathbf{d}_{12}^n \times \mathbf{b}_{12}^n}{\mathbf{x}_1^n \diamond \mathbf{d}_{12}^n \times \mathbf{x}_2^n} - \frac{\mathbf{b}_{23}^n \diamond \mathbf{d}_{23}^n \times \mathbf{x}_3^n}{\mathbf{x}_2^n \diamond \mathbf{d}_{23}^n \times \mathbf{x}_3^n} = 0 \tag{2.24}$$

Where the image point vectors are defined as:

$$\mathbf{x}_{i=1,2,3}^n = \boldsymbol{R}_{c,i}^n \begin{pmatrix} x_i - x_o - \Delta x_{d,i} \\ y_i - y_o - \Delta y_{d,i} \\ -f \end{pmatrix} \tag{2.25}$$

$\boldsymbol{R}_{c,i}^n$ is the rotation of the camera centre relative to a common frame n. The point (x_i, y_i) is key point correspondences, (x_0, y_0) the principle point, $(\delta x d_i, \delta y d_i)$ the correction for distortion at key point i and f is the cameras focal length. Using eq. (2.24), the up to scale parameters can be found.

Chapter 3

Setup

3.1 Hardware

3.1.1 Camera

(a) FLIR Blackfly S GigE

(b) Edmund Optics C Series

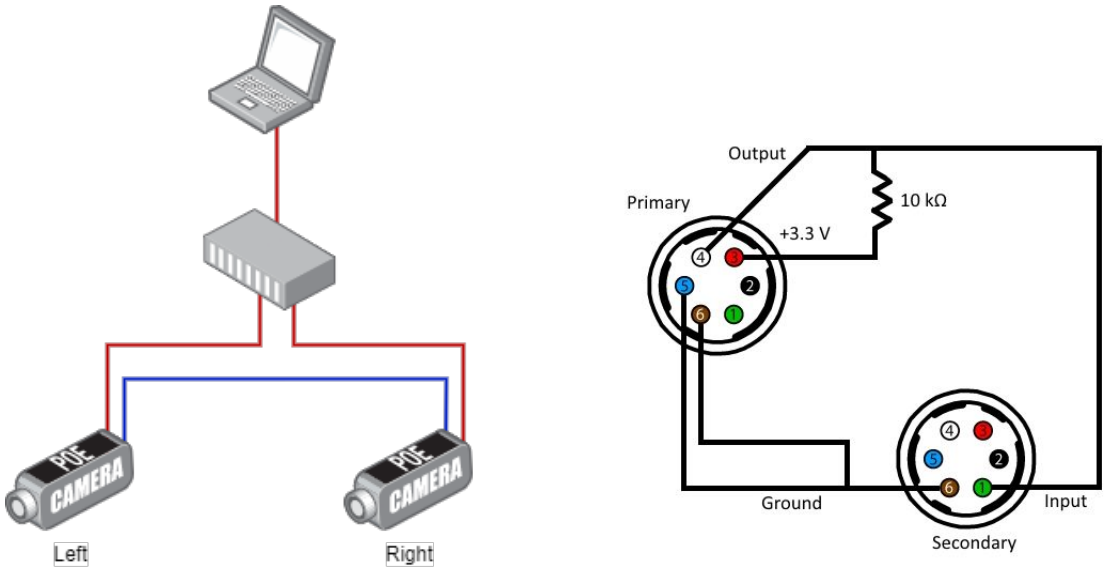
Figure 3.1: Components in stereo system

The cameras used on the stereo system are two Blackfly S GigE cameras from FLIR. They use the GigE interface to communicate, making it simple to connect multiple cameras together. They support power over ethernet (PoE), which simplifies wiring. The cameras have global shutters, which means that they capture the whole image in one instance, making them less susceptible to fast moving objects in the scene. The camera sensors are Sony IMX264, which utilizes CMOS with 2/3" format and has a resolution of 2448x2048 [14].

On both cameras there are fitted a lens from Edmund Optics. The lenses are from the 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°[4]. The high resolution and the lens makes the camera system able to detect objects far away. But the resolution comes at a price; the computational complexity when searching for matches in the stereo system can be heavy.

The operating temperature range for the cameras are from 0°C to 50°C, and they are not waterproof [14]. This makes them not well suited for use outdoors use in the Nordic conditions of Trondheim. Before the cameras can be put in service, a casing must be constructed. The casing has to withstand water, but also provide isolation from the cold winds.

3.1.2 System connection



(a) System overview

(b) GPIO wiring

Figure 3.2: Components in stereo system

The two cameras are connected through a router. To reduce the need of extra wiring, a router which can provide PoE is used. One Ethernet cable goes to each of the cameras, and one Ethernet cable is connected to the main computer. In order to synchronize the image capturing, the General Purpose IO (GPIO) connectors on the cameras needs to be wired together as in fig. 3.2b. This is done using a custom built GPIO cable where the interconnections and the resistance are a part of the cable. The cable enables the left camera, which is chosen to be the primary camera, to trigger the right camera simultaneously as the it receives its trigger command from the main computer.

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 are based on the work done by Theimann and Olsen, who mounted the cameras on a 1.8 meter long beam[28]. 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:

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

$$\boldsymbol{t} = \begin{bmatrix} 1800 & 0 & 0 \end{bmatrix} \tag{3.2}$$

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 [28]. While this 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 will be calculated in the calibration algorithm.

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. MilliAmperes computer runs Ubuntu 16.05 and the modules comunicates over a ROS system.

3.2.1 ROS

ROS is a 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 broadcasted 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 [23].

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 bag, which makes it possible to look at the history of the data gathered.

The ROS community holds a lot of open-source packages for different types of hardware. Drivers for the cameras are one of those packages. They are found on Github[19], and implemented in C++.

3.2.2 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. Since the user has the most experience in Python, this was the chosen language. OpenCV was used to extract and match features.[21]

Chapter 4

Experiments

Figure 4.1: Provisional mounting of cameras

To be able to test the algorithms on realistic data, the stereo camera rig was mounted on the ferry while the ferry sailed on the channel of Trondheim. Different paths were followed to capture different types of image-series. A total of 25 scenarios were recorded, resulting in more than 8000 images. In eight of these scenarios, a target boat with measured position was present. In other scenarios the boat observed april tags with known positions in the scene. The aim for these experiments was to cover a broad range of scenarios for the data set to be relevant for other specialization and master projects in the future.

4.1 Ground truth

MilliAmpere is equipped with sensors enabling the ferry to calculate its position and heading very precisely. This information is logged by ROS and recorded in ROS bags. Every message in ROS is timestamped by the computer's internal clock, making it easy to extract data later. Position of the april tags and target boat is measures using and SBG Ellipse GNSS receiver. The unit has a built IMU and post processing software enabling position accuracy down to 1cm [25]. During testing the precision of position averaged ±0.4 meters in longitude and ±0.3 meters in longitude.

(a) The unit

GNSS 1

Position			
Solution status			Solution Computed
Solution type			SBAS
Latitude	63.44228928 °	±	0.397 m
Longitude	10.39346862 °	±	0.292 m
Altitude (MSL)	0.333 m	±	0.794 m
Velocity			
Solution status			Solution Computed
Solution type			Doppler
Velocity (2d)	2.9 m/s	±	0.071 m/s
Track Course	241.3 °	±	0.6 °
True heading (HDT)			
Solution status			Insufficient Obs.
Heading	0.00 °	±	360.00 °
Pitch	0.00 °	±	90.00 °
Baseline			- m

GNSS information					
GPS			L1	L2	L5
GLONASS			L1	L2	L3
Galileo	E1	E5a	E5b	E5Alt	E6
BeiDou			B1	B2	B3
QZSS			L1	L2	L5
Num Sv Used					23
Base Station Id	136	Differential Age			3.60

(b) GNSS data during testing

Figure 4.2: SBG Ellipse used for ground truth

4.2 Synchronize with milliAmpere

Currently the stereo system is running on a separate computer and are not yet integrated with the onboard computer on milliAmpere. This means that during data gathering the data from milliAmpere are timestamped with milliAmperes inner clock, while the images are stamped with the clock of the external computer. To be make sure the clocks where synchronized the Network

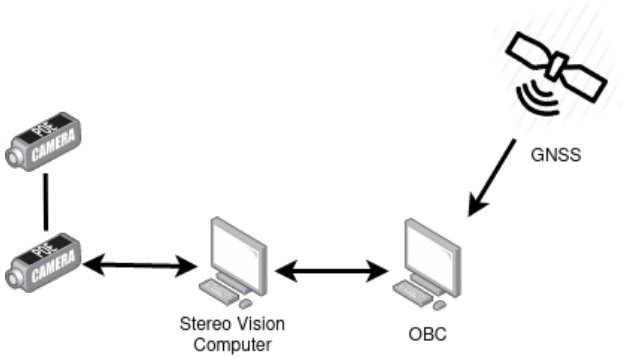


Figure 4.3: Synchronizing the stereo system with milliAmpere.
Figure courtesy of Kristian Auestad.

Time Protocol (NTP) was used. The computer on milliAmpere was configured as the server and the external as a client. With a Ethernet-cable between the two computers, the timestamps was synchronized down to a millisecond.

4.3 Structured data sets

Figure 4.4: Structure with april tags

17 different images series were captured in the harbour with and without april tags. These were captured over several days creating different light settings. Different angels and distances to the targets ensured a variety in the data. Some of these scenarios was tailored for the self-calibration algorithm. To be able to capture distortion in the entire frame, some of the scenarios involved following the facade shown on fig. 4.5a, hoping to capture a lot of the structure. Other trajectories were during normal operation. These series involved less structure to follow, but it might be a better representation of the data which will be available once the ferry is put into daily service.

(a) Facade followed

(b) Stereo image

Figure 4.5: Facade scenario

4.4 Traffic scenarios

In order for the ferry to be able to navigate safely in a marine environment, it has to be able to react to other boats in the area. This involves detecting and estimate distance to surrounding objects. The target boat with the SBG position unit was used to simulate a variety of different scenarios that might occur at sea. These scenarios were captured in the channel:

- Being overtaken
- Approached from a distance
- Crossed in front of milliAmpere
- Appearing from ship tunnel
- Entering ship tunnel
- Accelerating in front of milliAmpere
- Decelerating in front of milliAmpere
- Rapid and unexpected movement in front of milliAmpere

Including the scenarios listed a joyride data set was collected in open water, where the target boat moved in front of the cameras in different directions.

Figure 4.6: Leisure boat used as target in experiments.

Figure 4.7: Scenario in channel

Figure 4.8: Joyride in open water

Chapter 5

Calibration

Since the stereo vision system is to be implemented on a ferry that is going to be used by the public, it is desired to develop a self-calibrating algorithm that does not need a calibration rig to perform well. It will lessen the need of taking the ferry out of service if calibration could be done on image series collected during normal operation. By utilizing known geometry of multiple cameras and sensors data from milliAmpere, the calibration matrix could be estimated. A lot of the self-calibrating algorithms stems from the car industry [3][9][18]. When driving on a road, there are a lot of features that can easily be tracked. When at sea good matches might be sparser, which could lead to some problems. This ferry is going to operate in the channel of Trondheim, which can lead to more landmarks to be tracked.

Zhang et al.[31] created a self-calibrating algorithm for a stereo vision system for the Chang'e 3 lunar rover. They utilized collinearity equations to relate matches in each camera and the viewed point in the world. A continuous self-calibrating algorithm where implemented by Dang et al. [3], where the system had actuators to rotate the cameras. To handle this rotation, their system needed to recalibrate in real-time. The used an iterative extended Kalman filter to achieve effective and accurate calibration. Gopaul [8] presented an interesting approach where Scale Restraint Equations (SRE) where used in order to simplify the collinear equations. SRE relies on observing a landmark from three different angles. Gopauls approach was the starting point for this report's calibration algorithm.

Gopuals proposed method boils down to a set of equations that are to be minimized to estimate the intrinsics and extrinsics of the system. Since the system that the algorithm is design for only has two cameras, and SRE needs three viewpoints to be computed, a third image from a neighbouring epoch is used. For every three-point match three equations are added. These equations are minimized using Levenberg-Marquardt (LM). Gopauls method estimates all the

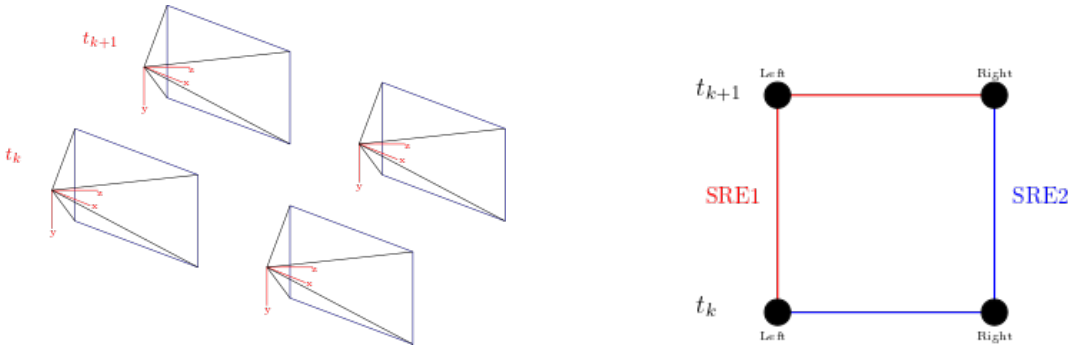


Figure 5.1: Three view matching between two time steps

intrinsic for both cameras, and all but one of the extrinsic variables. Since the equations suffers from rank deficiency, one of the variables in the baseline is kept free [8]. The horizontal distance between the two cameras is easy to measure by hand, and a physical measurement will often be more accurate than any estimation, so that is chosen as the free variable. This leaves a total of 16 variables to be determined. Gopaul suggested estimating the translation between every epoch in an image series. This adds a total of 6 variables per image to be estimated, making the estimation problem a lot harder. The ferry is equipped with sensors accumulating very accurate position and attitude measurements. Instead of estimating all the extra variables the data from the ferry is utilized. Gopaul suggested using two SRES per epoch. The first SRE is matching one stereo pair with the previous left image, while the second SRE matched the trailing epoch of images with the newest right camera. The constellation can be scene in fig. 5.1.

5.1 Algorithm

Figure 5.2: Algorithm steps

The calibration algorithm works in three steps. The first step synchronizes data with the images in the calibration series. Every image is timestamped with a clock that is synchronized with the one on milliAmpere. Position and heading from milliAmpere are stored in ROS bags, and after they are extracted, every image pair gets assigned the correct state.

The next step performs the feature extraction and matching. Since this project is a proof of concept which is running offline, SIFT descriptors are used. SIFT creates solid key points, but it is usually to slow for a real time implementation. The three-way matching is performed over

two operation. The one image that is common for both matching processes are first matched with one of the other images. Only the features that survives the first matching, are reused to be matched with the last image. For matching the, brute force method is used. Again, safety over speed is selected to guarantee good matches. Bad matches are filtered out by only using the 80% of the best matches in every image. Since one three-view match goes through two matching processes, the number of outliars are reduced.

The last step is the calibration. It is in this step LM is utilized. All the variables that are to be estimated is collected in a vector. This vector is ordered so that all the intrinsics of the left camera a put first, then the intrinsics of the right camera. The extrinsics are put last.

$$\boldsymbol{X} = \left[f_{x_L},\ f_{y_L},\ x_{0_L},\ y_{0_L},\ K_{1_L}, K_{2_L}\ K_{3_L},\ f_{x_R},\ f_{y_R},\ x_{0_R},\ y_{0_R},\ K_{1_R},\ K_{2_R},\ K_{3_R}, x,\ z,\ \phi,\ \theta,\ \psi\right] \tag{5.1}$$

To model the system SRE equations from section 2.5.2 are used. For every key point the point vector is constructed according to eq. (2.25). The distortion is modelled according to section 2.1.1 with 3 radial distortion coefficients. It is assumed that the tangential distortion is so low that it can be neglected. The relative translation between the two cameras is to be estimated, while the translation between to epochs are collected from the position data from milliAmpere. Since the point vectors are to be viewed in relation to the position to milliAmpere, it is important that they operate according to the same reference frame. The key points are described in camera frame, with z pointing into the scene and x horizontally, while milliAmpere has z pointing down, while x is following the fore bow. The image vectors are rotated to fit milliAmpere. The SREs are presented below.

$$\frac{\mathbf{x}_{R,k} \diamond \left(\mathbf{x}_{R,k} \times \mathbf{x}_{L,k}\right) \times \mathbf{R}_{c(k)} \mathbf{l}_{LR}^{c}}{\mathbf{x}_{R,k} \diamond \left(\mathbf{x}_{R,k} \times \mathbf{x}_{L,k}\right) \times \mathbf{x}_{L,k}} - \frac{\mathbf{x}_{L,k} \diamond \left(\mathbf{x}_{L,k} \times \mathbf{x}_{L,k-1}\right) \times \Delta \mathbf{X}_{L,k,k-1}}{\mathbf{x}_{L,k} \diamond \left(\mathbf{x}_{L,k} \times \mathbf{x}_{L,k-1}\right) \times \mathbf{x}_{L,k-1}} = 0 \tag{5.2}$$

$$\frac{\mathbf{x}_{L,k-1} \diamond \left(\mathbf{x}_{L,k-1} \times \mathbf{x}_{R,k-1}\right) \times \mathbf{R}_{c(k-1)} \mathbf{l}_{LR}^{c}}{\mathbf{x}_{L,k-1} \diamond \left(\mathbf{x}_{L,k-1} \times \mathbf{x}_{R,k-1}\right) \times \mathbf{x}_{R,k-1}} - \frac{\mathbf{x}_{R,k-1} \diamond \left(\mathbf{x}_{R,k-1} \times \mathbf{x}_{R,k}\right) \times \Delta \mathbf{X}_{R,k,k-1}}{\mathbf{x}_{R,k-1} \diamond \left(\mathbf{x}_{R,k-1} \times \mathbf{x}_{R,k}\right) \times \mathbf{x}_{R,k}} = 0 \tag{5.3}$$

Where

$$\mathbf{x}_{L,k} = \mathbf{R}_{c(k)} \mathbf{x}_{L,k}^{c} \tag{5.4}$$

$$\mathbf{x}_{R,k} = \mathbf{R}_{c(k)} \mathbf{R}_{cR}^{c} \mathbf{x}_{R,k}^{cR}, \tag{5.5}$$

$$\Delta \mathbf{X}_{R,k,k-1} = \Delta \mathbf{X}_{L,k,k-1} + \left(\mathbf{R}_{c(k)} - \mathbf{R}_{c(k-1)}\right) \mathbf{l}_{LR}^{c}. \tag{5.6}$$

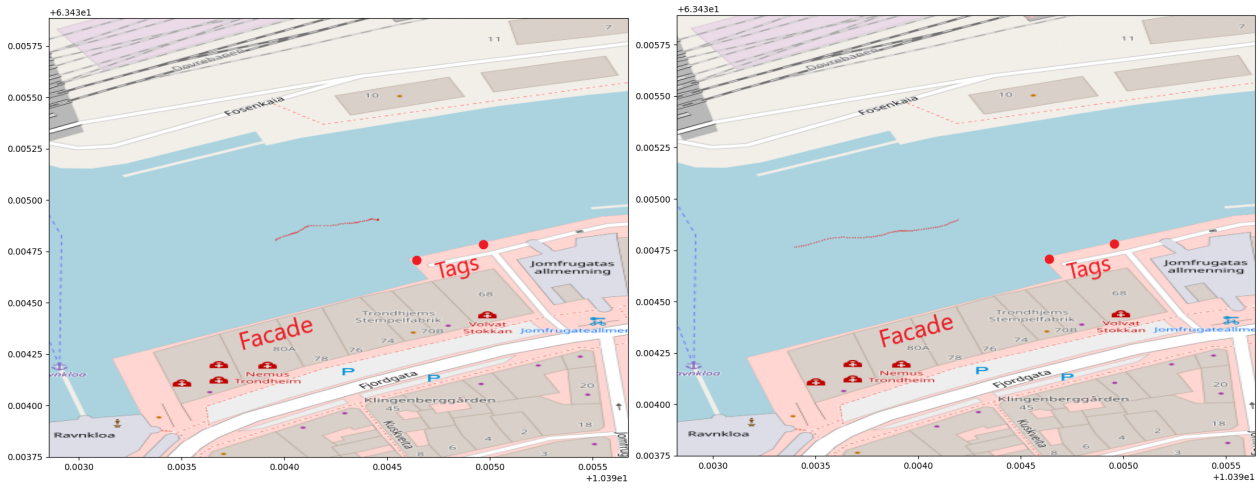
Equation (5.2) is SRE1 and represents the matches between one epoch of stereo cameras and the left image from previous epoch, while eq. (5.3) is SRE2 which matches the previous stereo pair with the current right image [9].

Since every image vector has three elements, one three-way match results in 3 residual equations which are feed into the LM optimizer. For every new epoch, the new residual functions are calculated, and the LM solver is rerun. To ensure that the LM to performs well, a good initial guess of the state $\boldsymbol{X}$ needs to be provided. When implemented on the ferry, the old calibration data would be providing a good starting point. When the algorithm is running, the results from previous epochs are used as initialization for the current epoch. In that way, the algorithm will eventually converge to the final results. The LM solver has a cap of maximum 100 iterations per epoch. Additionally, the solver will be stopped if the innovation between 2 iteration is too small. The results from the algorithm can be read from the final state in the algorithm.

Chapter 6

Results

The calibration algorithm was tested on two different scenarios. In scenario 1 the ferry is moving sideways, with while heading towards the april tags. The measured position of the april tags is not used but the contrast in the tags itself may provide some safe matches in the scene. Surrounding the tags there are little structure to find matches on, especially in the top and bottom of the images as these parts of the image are only containing sea and sky. In the second scenario the boat is moving sideways facing the facade. The facade provided more structure in the entire frame. Scenario 1 contains 126 stereo images, while scenario 2 has 147 images.



(a) Scenario 1

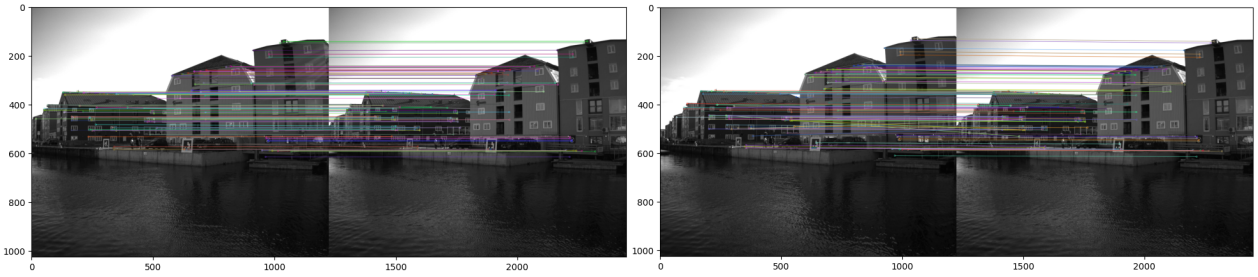
(b) Scenario 2

Figure 6.1: Trajectory of milliAmpere

6.1 Matching

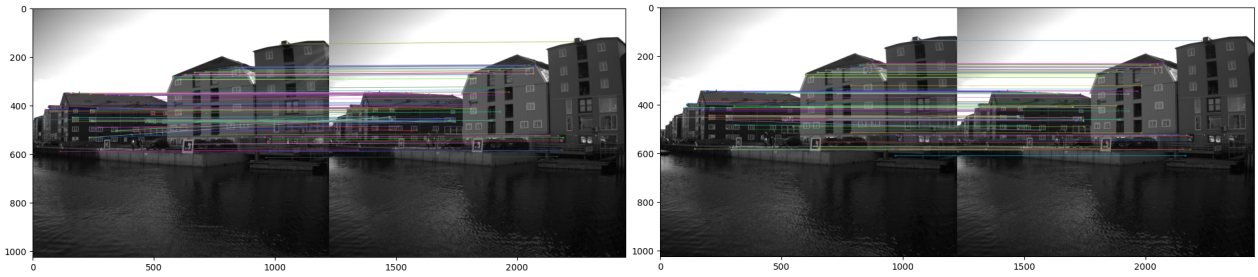
For the two SRE equations two different constellations of three view matching was needed. For every new stereo image pair, the preceding pair were also needed. The resulting image matching at a given iteration is presented below.

6.1.1 Scenario 1



(a) 24th left to 25th left (b) 25th right to 25th left

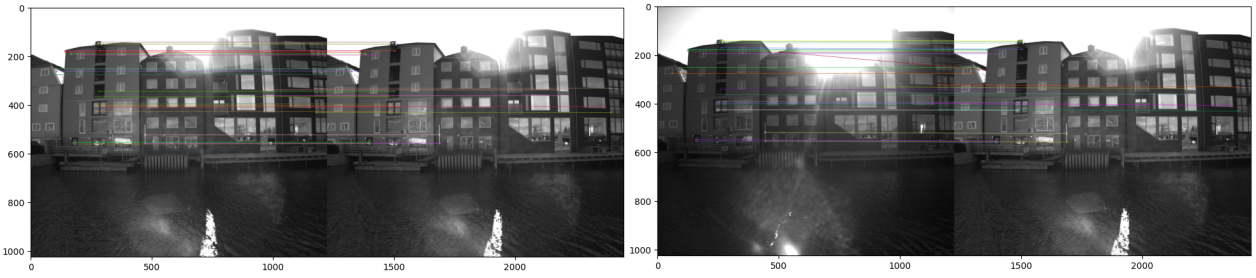
Figure 6.2: SRE1



(a) 24th left to 24th right (b) 25th right to 24th right

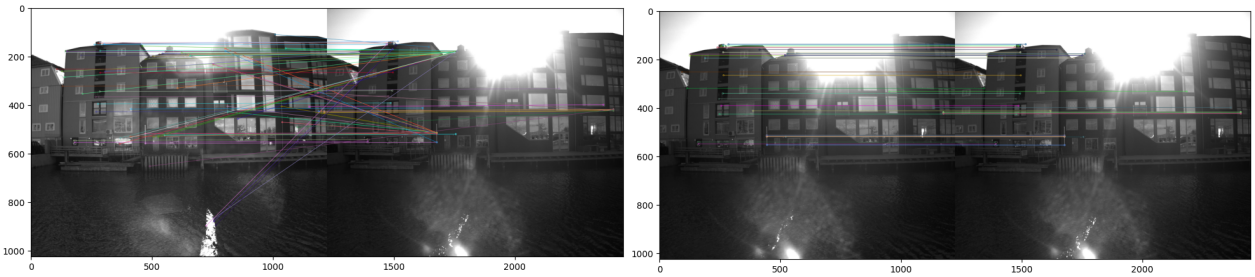
Figure 6.3: SRE2

6.1.2 Scenario 2



(a) 24th left to 25th left (b) 25th right to 25th left

Figure 6.4: SRE1



(a) 24th left to 24th right (b) 25th right to 24th right

Figure 6.5: SRE2

6.1.3 Number of matches

	Scenario 1	Scenario 2	Scenario 2 (One SRE)
SRE1	18221	8378	8378
SRE2	23221	20269	-
Total	41442	28647	8378

6.1.4 Discussion

Both scenarios managed to find a lot of three-view matches. Even though scenario 1 has fewer images, the total amount of matches was higher than for scenario 2. Although there are more matches, they seem to be placed more in the centre of the image. Since the cameras are observing so much of the sea and skyline, it is impossible to find reliable matches in these parts of the image frame. This can lead to trouble during calibration. In scenario 2 more of the frame is

covered, but also here there are some areas of the image that are not covered. Since the boat is to so far away from the facade, the water is taking up a lot of the scene.

One major discovery during testing was that the SRE2 constellation, that is the one matching between one stereo pair and the following right image, picked up a lot of outliars. It was assumed that since three-view matching was used, the number of outliars would be held to a minimum and therefor a simple removing of the 20% worst matches would be sufficient. In the SRE2 for scenario 2 this assumption obviously does not hold. The matching would have benefited from adding a filtering step to remove those outliars. Different RANSAC method should be considered [6] to remove this. From the other constellations there are far less outliars. The calibration algorithm should be able to handle some outliars, as it can be very difficult to remove all outliars in a stereo system.

6.2 Calibration

To evaluate the performance, a reference calibration of the intrinsics was performed using Zhang's method from an onshore calibration. Since this calibration was done at a distance that would not represent the operational distance in which the camera are to be used in a real life, only intrinsic calibration for the cameras was performed. A 3.0mx1.5m checkerboard was used, and the cameras was moved around to capture multiple views of the pattern. The rig was mounted on top of planar surface. It would be expected that both x- and z-translation should be very low. Due the rotation of the cameras, a small value in x, are to be expected. Since cameras are tilted inwards slightly, an angle of -2°is expected in ψ (rotation around z-axis). There are fine margins when working with angles, so some rotation along the other axis is also to be expected, but these should be considerably lower than ψ.

The calibration is performed using the Levenberg-Marquardt optimization scheme. Optimization problems are dependent on a good initial guess of the solution to perform well. The initial intrinsic values were based on the results from Zhang's method. For the extrinsic values, all values were set to zero expect yaw, which is expected to be around rotated by -2°. Calibration was performed on both scenarios. Because of the outliar problem with SRE2 in scenario 2, an additional run was performed on this scenario where only the matches from SRE1 were used.

Equation (6.1) shows the initial variable state.

$$\begin{aligned}\boldsymbol{X}_0 = \big[&f_{x_L}:1200,\ f_{y_L}:1200,\ x_{0_L}:600,\ y_{0_L}:500,\ K_{1_L}:-0.39,\ K_{2_L}:0.21,\ K_{3_L}:-0.08,\\ &f_{x_R}:1200,\ f_{y_R}:1200,\ x_{0_R}:600,\ y_{0_R}:500,\ K_{1_R}:-0.40,\ K_{2_R}:0.25,\ K_{3_R}:-0.14,\\ &x:0,\ z:0,\ \phi:0,\ \theta:0,\ \psi:-2\big]\end{aligned} \tag{6.1}$$

6.2.1 Reference

Intrinsics

(6.2)

$$\boldsymbol{K}_{Left} = \begin{bmatrix} 1238.288 & 0 & 609.203 \\ 0 & 1238.267 & 538.544 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} K_{1_L} = -0.396 \\ K_{2_L} = 0.217 \\ K_{3_L} = -0.078 \end{matrix} \tag{6.3}$$

(6.4)

$$\boldsymbol{K}_{Right} = \begin{bmatrix} 1238.262 & 0 & 625.479 \\ 0 & 1238.411 & 538.514 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} K_{1_R} = -0.402 \\ K_{2_R} = 0.251 \\ K_{3_R} = -0.137 \end{matrix}$$

6.2.2 Scenario 1

Intrinsics

(6.5)

$$\boldsymbol{K}_{Left} = \begin{bmatrix} 1200.717 & 0 & 519.154 \\ 0 & 1200.036 & 462.245 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} K_{1_L} = -0.390 \\ K_{2_L} = 0.196 \\ K_{3_L} = -43.470 \end{matrix} \tag{6.6}$$

(6.7)

$$\boldsymbol{K}_{Right} = \begin{bmatrix} 1200.563 & 0 & 616.470 \\ 0 & 1200.106 & 491.884 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} K_{1_R} = -0.400 \\ K_{2_R} = 0.234 \\ K_{3_R} = -40.917 \end{matrix}$$

Extrinsics

(6.8)

$$\boldsymbol{R} = \begin{bmatrix} -0.699 \\ 1.721 \\ -3.382 \end{bmatrix} \quad \boldsymbol{t} = \begin{bmatrix} 0.391 \\ 1.800 \\ 7.0e-4 \end{bmatrix} \tag{6.9}$$

6.2.3 Scenario 2

Intrinsics

(6.10)

$$\boldsymbol{K}_{Left} = \begin{bmatrix} 1200.203 & 0 & 610.915 \\ 0 & 1200.703 & 469.704 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} K_{1_L} = -0.390 \\ K_{2_L} = 0.194 \\ K_{3_L} = -26.378 \end{matrix} \tag{6.11}$$

(6.12)

$$\boldsymbol{K}_{Right} = \begin{bmatrix} 1200.431 & 0 & 650.250 \\ 0 & 1200.106 & 518.345 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} K_{1_R} = -0.400 \\ K_{2_R} = 0.225 \\ K_{3_R} = -32.014 \end{matrix}$$

Extrinsics

(6.13)

$$\boldsymbol{R} = \begin{bmatrix} 1.817 \\ 2.868 \\ -3.275 \end{bmatrix} \quad \boldsymbol{t} = \begin{bmatrix} -2.739 \\ 1.800 \\ -0.074 \end{bmatrix} \tag{6.14}$$

6.2.4 Scenario 2 with only SRE1 active

Intrinsics

(6.15)

$$\boldsymbol{K}_{Left} = \begin{bmatrix} 1200.173 & 0 & 569.892 \\ 0 & 1200.115 & 484.391 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} K_{1_L} = -0.390 \\ K_{2_L} = 0.198 \\ K_{3_L} = -27.518 \end{matrix} \tag{6.16}$$

(6.17)

$$\boldsymbol{K}_{Right} = \begin{bmatrix} 1200.067 & 0 & 598.441 \\ 0 & 1199.330 & 502.438 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} K_{1_R} = -0.400 \\ K_{2_R} = 0.250 \\ K_{3_R} = -0.131 \end{matrix}$$

Extrinsics

(6.18)

$$\boldsymbol{R} = \begin{bmatrix} -0.476 \\ 0.653 \\ -1.765 \end{bmatrix} \quad \boldsymbol{t} = \begin{bmatrix} -0.634 \\ 1.800 \\ 0.423 \end{bmatrix} \tag{6.19}$$

6.2.5 Discussion

All the scenarios deliver intrinsic values that might look reasonable compared with the reference. Regarding the distortion coefficients, all scenarios managed to find K_1 and K_2 values relatively low and similar to those the reference, but K_3 blew up to abnormally high numbers in almost all the cases. One theory why this happens might be the lack of matches along the edges of the image, and therefor might not be able to pick of the distortion in the frame. The only K_3 that looks normal is the right one in scenario 2 with only one SRE. Since the boat is moving sideways, the SRE1 in scenario 2 might be able to cover the whole image well for the right camera, but it lacks information for the left camera to obtain the last distortion coefficient of that camera. The principle point values in scenario looks a bit off, especially in the left camera

where this point are shifted by almost 100 pixels in both directions. The scenario that seems to be performing the best regarding principle point is the scenario 2.

Regarding the extrinsics the one scenario which delivered the least plausible values was the scenario 2. This was to be expected considering the amount of outliars there were in SRE2. When SRE2 was not used, the system seamed to deliver better extrinsics, but still x and z values are too high to be considered plausible. The values resulted from scenario 1 are the one that are the most likely to be true, with the very low z value. An x of 0.391 is still too high. Looking at the rotation all the angles are close to doubled from scenario 1 to scenario 2 with one SRE. Since there are no values to compare with, it is difficult to say which ones are closest to the correct value.

6.3 Run time

Both the matching and calibration was performed on a computer with an Intel Core i7-8700 processor running on Ubuntu 20.04. Extracting data from ROS and acquiring the images are not included in the timing, as this is assumed done beforehand. The timing is measured in seconds.

	Scenario 1	Scenario 2	Scenario 2 (One SRE)
Matching	52	59	32
Optimizing	2411	801	191
Total	2463	860	223

6.3.1 Discussion

With the most amount of matches, there is no surprise that scenario 1 takes the most amount of time to compute. Every point match result in three equations, resulting in a big jacobian to be computed. There was also a correlation between how hard the optimization problem was to solve and the amount of time taken to compute. If the initial values were set to some initial values other than those based on the reference result, the calculation time increased a lot. Scenario 1 adjusting the principle point so much might be the reason for the more than doubling of computational time in optimization. The low run time of Scenario 2 with only one SRE is of course due the low amount of matches it considers.

Chapter 7

Conclusion

The proposed calibration yields results that indicates that it might work as a calibration method on milliAmpere. The basis of which the results are evaluated however are a bit too thin in order to conclude that the algorithm was a success. During matching there were found to be to many outliars in one of the three-view matching, leading to the results in scenario 2 being sub optimal. The good distortion coefficients for one of the cameras when only one SRE was active might indicate that images with a lot of structure in the entire image is important for detecting distortion. The promising extrinsic values from scenario 1 might indicate that both SREs should be active to get good calculate rotation and translation of the cameras.

In developing the algorithm further, there should be considered implementing a RANSAC method to eliminate outliar in matching. That would lead to a more robust method that can withstand more noisy images. For robustness, the algorithm needs to be tested on different scenarios. The calibration results, especially the extrinsic needs to be evaluated in a better way. Now the only measurements for extrinsics are by reasoning and logic. One way of checking that the calibration results are satisfactory could be to use them in other applications, such as creating disparity maps, to evaluate how well that method performs with the given calibration values.

Considering the run time of the calibration method, it looks like there is some work needed for it to be able to run on the onboard computer of milliAmpere. With around 40 minutes of calculation time on a powerful computer, the run time would be a lot higher on the ferry, where the computational power needs to be shared with other tasks as well. With an implementation of RANSAC the matching time will be increased, but the reduction of numbers of matches will be reduced. To speed up the matching process, using ORB as a descriptor should be considered. With the introduction of RANSAC, the FLANN matcher could also be utilized since the risk of mismatches will be reduced. It should also be studied who many matches are needed to ensure

a good result and maybe consider adding capping the amount of matches added per epoch. Finding the optimal length for images-series needed to converge to the correct result will also reduce the run time.

7.1 Future work

Additional to the further development of the calibration algorithm, there are some problems needed to be solved for the stereo system to be permanently installed on milliAmpere.

- The camera rig needs to be mounted permanently on milliAmpere. Currently the cameras are mounted on a 1.8 meter long beam that are mounted on the hood of milliAmpere using tape. With a permanent instalment, there would be a lot more consistency between days of experiments.
- Weatherproofing of the cameras. The cameras are not waterproof, and should not be subjected to temperature below 0 °C [14]. A casing for the cameras should be created so that the stereo system can be used in all weather conditions.
- Integration with milliAmpere. The stereo system is operated using a separate computer. If system is to be a permanent part of milliAmpere, the launch code needs to be transferred to the onboard computer of milliAmpere and integrated with the ROS-system.

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

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

NTP Network Time Protocol

ORB Oriented FAST and Rotated BRIEF

PoE Power over Ethernet

RANSAC RANdom SAmple Consensus

ROS Robot Operatin System

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization And Mapping

SRE Scale Restraint Equation

SSD Sum of Squared Differences

Bibliography

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

[2] 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.

[3] 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.

[4] Edmund Optics Inc. *8.5mm C Series Fixed Focal Length Lens*. URL: https://www.edmundoptics.com/p/85mm-c-series-fixed-focal-length-lens/14947/.

[5] 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.

[6] Mehran Fotouhi et al. "SC-RANSAC: Spatial consistency on RANSAC". In: *Multimedia Tools and Applications* 78.7 (2019), pp. 9429–9461. ISSN: 15737721. DOI: 10.1007/s11042-018-6475-6.

[7] 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.

[8] 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).

[9] 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.

[10] 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.

[11] 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.

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

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

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

[15] 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.

[16] Alexander Mordvintsev and K Abid. *Feature Matching*. 2013. URL: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html.

[17] 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.

[18] 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.

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

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

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

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

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

[24] 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.

[25] SBG Systems. *Ellipse Series*. URL: https://www.sbg-systems.com/products/ellipse-series/.

[26] 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.

[27] 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.

[28] Line Charlotte Kristoffersen Theimann and Trine Ødegård Olsen. “Stereo vision for autonomous ferry”. In: *Master’s Thesis* (2020).

[29] 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.

[30] Jinghao Yang et al. “Precision calibration method for binocular vision measurement systems based on arbitrary translations and 3D-connection information”. In: *Measurement Science and Technology* 27.10 (2016). ISSN: 13616501. DOI: 10.1088/0957-0233/27/10/105009.

[31] 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.

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