Comparison of depth information from stereo camera and LiDAR

Specialization project submitted to
Norwegian University of Science and Technology,
Department of Cybernetics

January 2020

Lina Charlotte Kristoffersen Theimann

Supervisor: Edmund Førland Brekke
Co-supervisors: Annette Stahl, Øystein Kaarstad Helgesen

NTNU
Norwegian University of
Science and Technology

Abstract

The detection of 3D objects is an essential task for autonomous vehicles. A common approach is to obtain the 3D measurements from LiDAR technology, which is precise but expensive. A cheaper approach is to utilize stereo cameras for 3D object detection purposes. The problem is gaining the same accuracy as the LiDAR. This report presents work done in comparing the depth information obtained by a stereo camera with a LiDAR.

Disparity map algorithms based on block matching with Sum of Absolute Differences as a cost function and Semi-Global Matching are implemented and the depth compared with the LiDAR. Both a visual comparison and a calculation of the distance between corresponding points in the two sensors is performed. The depth estimates from the LiDAR is considered as the ground truth. A method for finding corresponding points between two point clouds using a nearest neighborhood search is proposed. It requires the point clouds to be aligned and represented in the same coordinate frame. An extrinsic calibration of the sensors is performed using a Iterative Closest Point algorithm on the point clouds. When used on aligned point clouds the method provide a good measure of the accuracy of the depth estimate obtained by the stereo camera. An experiment is conducted and the analysis of the data shows that there is a possibility for the stereo camera to obtain an accurate depth estimate compared with the LiDAR. Through Semi-Global matching only a small deviation from the LiDAR data is present.

Preface

This project report presents the work done for the mandatory course TTK4550 - Engineering Cybernetics Specialization Project, taken during the fall of 2019.

Some parts of the project is written in collaboration with Trine Ødegård Olsen who has written the complementary report; *Stereo vision using local methods for autonomous ferry*. This includes section 2.1, 2.2, the stereo camera part of section 4.1 and the calibration in section 5.1 and 5.2. Together we would like to thank our supervisors Edmund Førland Brekke, Annette Stahl and Øysten Kaarstad Helgesen for their help and knowledge throughout the project. We would also like to thank Glenn Angel and all the technical staff at ITK for their help with the sensor setup. We would like to thank Stefano Brevik Bertelli for always being helpful in times of need, and Egil Eide for taking the time to show us the ferry Milliampere.

Table of Contents

Abbreviations

LiDAR	=	Light Detection And Ranging
SAD	=	Sum of Absolute Differences
ICP	=	Iterative Closest Point
CCD	=	Charged Coupled Device
FOV	=	Field of View
GigE	=	Gigabit Ethernet
GPIO	=	General Purpose Input/Output
VLP	=	Veoldyne LiDAR Puck
IP	=	Internet Protocol
ROS	=	Robot Operating System
TCP	=	Transmission Control Protocol
OpenCV	=	Open Source Computer Vision Library
RANSAC	=	Random Sample Consensus

Chapter 1

Introduction

1.1 Motivation

Situational awareness for an autonomous vehicle depends on sensors that can perceive obstacles and other vehicles, and track and predict their motion so that collisions are avoided. The ability to perceive depth is a crucial part of the process. LiDAR is one of the most popular sensors used for this purpose when operating in urban environments. On the autonomous ferry Milliampere LiDAR is one of the primary sensor used for 3D object detection.

The LiDAR is a precise but expensive sensor. At Tesla Autonomy Investor Day Elon Musk said: ”LiDAR is a fool’s errand. Anyone who relies on LiDAR is doomed. Expensive sensors that are unnecessary” [30]. Tesla’s self-driving cars only rely on cameras and radar, while most other developers use LiDAR combined with cameras and radars. The reason why LiDAR is most often used is that 3D approaches based on stereo cameras have up until now have not been able to reach the same level of accuracy as the LiDAR. But in 2019 researchers at Cornell University [33] presented a report where a method for detecting objects using to inexpensive cameras achieved almost the same accuracy the LiDAR. With this they have shown that in theory it is possible that stereo cameras could replace the LiDAR in the future. The task of comparing the depth accuracy obtained by a stereo camera with the LiDAR is thus proven very interesting.

1.2 Background and previous work

A substantial amount of research has been done in the field of remote sensing. For autonomous vehicles a reliable depth estimation is a crucial task. Previous research into depth estimation by stereo cameras has not been able to reach the level of accuracy that the LiDAR has. That is not until in April 2019 when researchers at Cornell University published the paper *Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in*

3D Object Detection for Autonomous Driving [33]. They were able to triple the accuracy of depth estimation by stereo cameras at 30 meters by changing the way one represents the data. With this they showed that in principle it is possible to replace the LiDAR with stereo cameras.

For over 60 years research has been done in the field of computer vision. The stereo vision is one of the more popular techniques from this field of study. A substantial amount of research on different disparity map algorithms has been done. However, most of the research tends to focus on object detection at small distances around one to two meters. Depth estimation with a stereo camera at longer distances needs to be looked more into.

Several literature surveys comparing different matching methods for creating disparity maps have been published. Among them are [15], [20] and [19]. The different matching methods can be divided into local and global methods. The global methods are generally more accurate but less efficient than local methods. Since stereo cameras are supposed to be used on an autonomous ferry real-time performance is important. Thus local algorithms are more feasible. One of the local algorithms performing well is the Sum of Absolute Differences, SAD, due to its low computation time [28]. This is a well-known algorithm and different implementations are available online. In 2005 Hirschmuller [17] proposed a Semi Global Matching Method performing pixel-wise matching based on Mutual Information and the approximation of a global smoothness constraint. It was proven that the algorithm performed the matching almost as efficient as local methods but with higher precision. The semi global matching method and the SAD algorithm are thus used in this project. The accuracy achieved by the algorithms are compared with the LiDAR.

Comparing depth information from the LiDAR and stereo camera is typically done by utilizing the sensors on the same data set and not by representing the data together. The 3D data obtained from both the stereo camera and the LiDAR can be represented as point clouds. To get a more direct comparison computer vision techniques for point clouds are used in this project. To do the comparison the data needs to be represented in the same coordinate frame. Estimating the extrinsic calibration parameters can be performed by using methods for point cloud registration. In 1992 Besl and Kay [4] proposed a method for registering 3D shapes of any form. It uses the Iterative Closest Point (ICP) algorithm and is able to register two data sets of different sizes. The method is fast and perform well. However, it requires the number of statistical outliers to be near zero.

Other techniques for performing extrinsic calibration of a LiDAR and cameras have been developed mostly for sensor fusion purposes. A method for estimating extrinsic calibration parameters using 3D-3D correspondences was proposed by Dhall et al. [6]. The method finds the extrinsic parameters between a LiDAR and a monocular camera. In 2017 Guindel et al. [13] proposed a method for automatic extrinsic calibration of LiDAR-stereo sensor setups. The methods allow for different relative poses of the sensors and do not require user interference. The published code requires a specific calibration target of size 80cm by 120cm with four symmetrically disposed circular holes. The problem with techniques like these is that they are used for monocular cameras or on a specific calibration target. The objective is to use the stereo camera for long distances which might affect the calibration accuracy. The calibration target needs to be fully visible for both sensors and when using a small target some LiDARs might see less of the structure of the object when placed at a great distance. This is dependent on the vertical resolution of the

LiDAR used.

Another helpful tool in analyzing point clouds is the nearest neighborhood search. In 2009 Muja and Lowe[27] published a paper presenting a fast and accurate nearest neighborhood search method. The method includes different search algorithms depending on the data it is used on. The best algorithm for the search and the optimal parameters are automatically selected. Thus the method can be used to find the nearest neighbor of a point in a point cloud.

1.3 Problem description

The objective of this specialization project is to compare the depth information obtained from a stereo camera with the one obtained from a LiDAR. With this an analysis of the possibility of using the stereo setup for 3D object detection on the autonomous ferry Milliampere can be performed. The accuracy of different algorithms for obtaining depth information should be evaluated. The LiDAR is considered as the ground truth and the necessary steps in order to compare the information retrieved by the two sensors should be defined and implemented. This project includes the following tasks:

1. Design a stereo setup to be used on the ferry Milliampere.
2. Perform a calibration of the stereo camera such that the images can be undistorted and rectified.
3. Integrate a velodyne LiDAR with the stereo setup.
4. Implement a method for retrieving depth from the stereo camera.
5. Perform an extrinsic calibration of the LiDAR and stereo camera.
6. Implement a method for comparing depth information from stereo camera with the LiDAR.
7. Perform the comparison on a test scene.

1.4 Report outline

The structure of the projects is as follows. First, some theoretical background of the sensors used in the project is presented in chapter 2. Chapter 3 presents the algorithms used for estimating depth from the stereo camera and comparing the depth information of the two sensors. The used method for comparing depth information by the two sensors is presented. Chapter 4 deals with the hardware and software used. The sensors used and the decisions behind the setup are given. In chapter 5 the calibration process of both the stereo camera and the calibration of the LiDAR and the stereo camera is described. Chapter 6 the experiment performed is presented along with the results obtained and the following discussion. Chapter 7 concludes the project and gives suggestions for further work to be done.

Chapter 2

Sensor Fundamentals

This chapter introduces the theory behind the sensors used in this project. Two single cameras are connected to generate a stereo vision and a 360° LiDAR is used as ground truth to the depth information retrieved by the stereo camera. First, the principles behind a camera is presented. Then, the theory behind the stereo camera and how one can retrieve depth information from it is given. In the end, some background of the LiDAR is presented. The theory presented in chapter is mainly based on the VSLAM compendium by Trym Haavardsholm [14] and the textbook ”Multiple View in Geometry in computer vison” by Hartley and Zisserman [16].

2.1 Camera

2.1.1 Pinhole model

The pinhole model is one of the most widely used geometric camera models [14]. It is a mathematical model describing the correspondence between real-world 3D points and 2D points in the captured image.

Figure 2.1 is an illustration of the model. The box represents the camera body. On one side of the camera body we have the camera film or the image plane which captures the light emitted through the pinhole. Only light rays from one direction are allowed through at a time. Thus the light rays coming from other directions is filtered out. The light rays will move through the hole in a straight line, as illustrated by the red lines in the figure. Thus light from the bottom of the object will end up at the top of the image, and light from the top of the scene will end up at the bottom. In the images produced everything will be in focus. For the model to work properly, long exposure time is necessary to avoid blur or the absence of moving objects. In practise cameras are usually equipped with a lens in order to produce useful images. The pinhole is mathematical convenient and the geometry of the model is an adequate approximation of the imaging process.

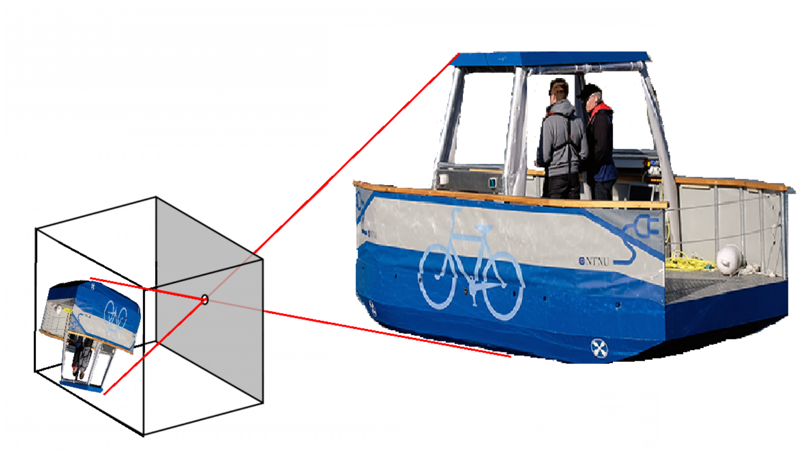


Figure 2.1: Pinhole model

2.1.2 Camera parameters

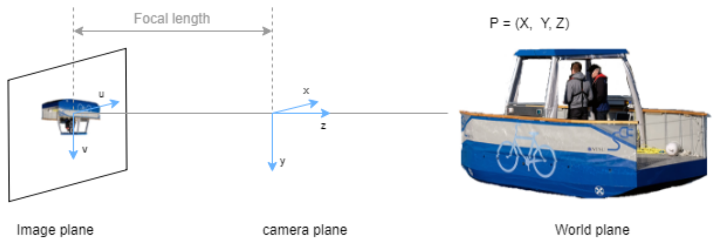


Figure 2.2: Extrinsic and extrinsic geometry

The camera parameters are what describe the relationship between the camera coordinate system and real-world coordinates. They can be divided into two groups; intrinsic and extrinsic parameters. The intrinsic parameters represent the focal length and the optical center of a camera and it is described as a mapping between camera coordinates and pixel coordinates in the image frame. The extrinsic parameters represent the position of the camera in real-world coordinates and it is described as a mapping between the camera coordinates and the real-world coordinates. See figure 2.2. The image plane is parallel to the camera plane at a fixed distance from the pinhole. This distance is called the focal length, f. The gray line in the figure represents the principal axis. The principal axis intercepts the image plane in a point called the principal point.

Homogeneous coordinates

Homogeneous coordinates are an alternative way to describe Cartesian coordinates in Euclidean space. It is convenient to use when representing geometric transformations. Mapping a homogeneous vector to a Cartesian space and Cartesian to homogeneous is given by equation 2.1 and equation 2.2 respectively [14].

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \longrightarrow \tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1}$$

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \\ \tilde{w} \end{bmatrix} \longrightarrow \mathbf{x} = \begin{bmatrix} \tilde{x}/\tilde{w} \\ \tilde{y}/\tilde{w} \\ \tilde{z}/\tilde{w} \end{bmatrix} \tag{2.2}$$

Intrinsic parameters

The intrinsic parameters are described by the transformation between camera coordinates and pixel coordinates in the image frame. The world coordinates are assumed to be given in the camera coordinate frame, where the origin is in the optical center. Considering the reference frames showed in figure 2.2, the mapping between the Euclidean 3D space, (X, Y, Z), to Euclidean 2D space (u,v) is expressed as

$$u = \frac{f}{Z}X \quad v = \frac{f}{Z}Y \tag{2.3}$$

The corresponding mapping in homogeneous coordinates is thus given by

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \to \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.4}$$

This mapping does not take into consideration that the origin of the image plane may not coincide with the principal point. If this is the case, then the mapping becomes

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \to \begin{bmatrix} f\frac{X}{Z} + p_x \\ f\frac{Y}{Z} + p_y \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \to \begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

where $(p_x, p_y)^T$ are the coordinates of the principal point and

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

is the camera calibration matrix of a true pinhole camera.

Most often cameras are not true pinhole cameras. The pinhole model assumes pixels to be square which may not be the case for other camera models. CCD (Charged-coupled device) is the model most often found in digital cameras. By a few adjustments the pinhole model can be made to fit the CCD model. In CCD cameras there is the possibility of having non-square pixels. This may cause an unequal scale factor in each direction if image coordinates is measured in pixels. To account for this m_x and m_y are defined as the number of pixels per unit distance in image coordinates in x and y directions, respectively. The camera calibration matrix then becomes

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represents the focal length in pixel units, and $x_0 = m_x p_x$ and $y_0 = m_y p_y$ represents the principal point in pixel units [16]. The s is called the skew parameter. It is non-zero in cameras where the image axes are not perpendicular. This is usually not the case for most cameras.

Extrinsic parameters

The camera frame and the world reference frame are related via a translation and a rotation.

If we have two coordinate frames $\mathcal{F}_a$ and $\mathcal{F}_b$ with different position and orientations, the transformation between the two frames can be described by the homogeneous transformation matrix

$$\mathbf{T}_{ab} = \begin{bmatrix} \mathbf{R}_{ab} & \mathbf{t}^a_{ab} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \text{where} \quad \mathbf{x}^a = \mathbf{T}_{ab}\mathbf{x}^b, \tag{2.8}$$

$\mathbf{R}_{ab}$ is the orientation of $\mathcal{F}_b$ relative to $\mathcal{F}_a$, and $\mathbf{t}^a_{ab}$ the translation vector given in the coordinate frame of $\mathcal{F}_a$ [14].

When describing the intrinsic parameters the world coordinates were assumed to be given in the camera frame. Now, to describe the extrinsic parameters the scene points are expressed in the world coordinate frame (X, Y, Z), the one to the far right in figure 2.2. A point, x_w in the world frame can be expressed as a point, x_c, in the camera frame by

$$\mathbf{x}_c = \mathcal{R}\mathbf{x}_w + \mathbf{t} \tag{2.9}$$

$\mathcal{R}$ and $\mathbf{t}$ is thus the extrinsic parameters of the camera.

Combining the intrinsic parameters in 2.6 or 2.7 with 2.9 we get the camera matrix. It relates world coordinates with pixel coordinates and can be written as

$$\mathbf{P} = \begin{bmatrix} \mathcal{R} \\ \mathbf{t} \end{bmatrix} K \tag{2.10}$$

Distortion

Distortion occurs in lenses. It makes straight lines in a scene appear bent in the image. An ideal pinhole camera does not have a lens and thus it is not accounted for in the camera matrix.

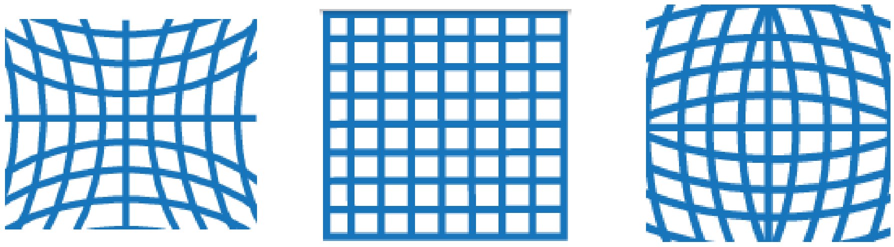
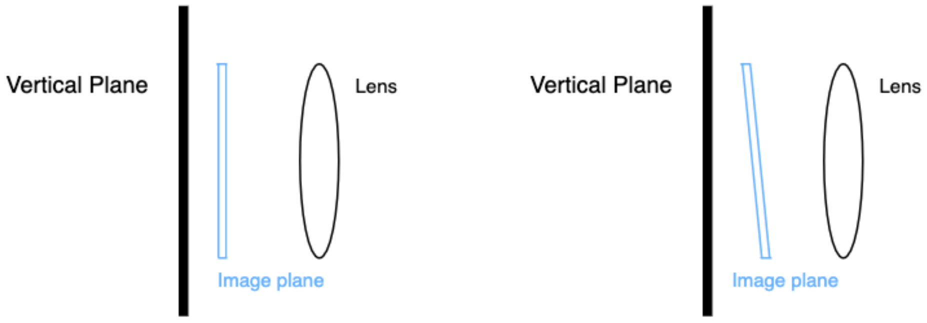

(a) Negative radial distortion **(b)** No distortion **(c)** Positive radial distortion

Figure 2.3: Radial distortion by MathWorks [23]

There are different kinds of distortion; radial distortion and tangential distortion. The most important one is the radial distortion. It occurs when light rays are bent more near the edges of the lens than in the center. Figure 2.3 shows how different types of radial distortion looks like in an image. Letting (x, y) be the ideal points and (x_d, y_d) the radial distortion can be modeled as

$$\begin{aligned} x_d &= x + x\left[k_1(x^2+y^2) + k_2(x^2+y^2)^2 + k_3(x^2+y^2)^3 ...\right] \\ y_d &= y + y\left[k_1(x^2+y^2) + k_2(x^2+y^2)^2 + k_3(x^2+y^2)^3 ...\right] \end{aligned} \tag{2.11}$$

where k_1, k_2, k_2, ... are the radial distortion coefficients of the lens[35].



(a) No tangential distortion **(b)** Tangential distortion

Figure 2.4: Tangential distortion

Tangential distortion occurs when the lens and the image plane are not parallel, see figure 2.4. It can be modeled as

$$\begin{aligned} x_d &= x + \left[2p_1xy + p_2(x^2 + y^2 + 2x^2)\right] \\ y_d &= y + \left[p_1(x^2 + y^2 + 2y^2) + 2p_2xy\right] \end{aligned} \tag{2.12}$$

where as before (x, y) are the undistorted pixel locations and (x_d, y_d) the distorted ones [23]. p_1 and p_2 are the tangential distortion coefficients of the lens.

2.1.3 Camera Calibration

Camera calibration is the process where intrinsic and extrinsic parameters are estimated. Lens distortion is modeled and correct for. The intrinsic and extrinsic parameters are what relates the world and the camera coordinate system. These parameters can be used to get the size of an object in the real world, or determine where in the scene the camera is located. There exist different techniques, but the two main ones according to Zhang[35] are:

- **Photogrammetric calibration**: calibration is performed by using an object with known 3D world points. The correspondences between the real world points and the corresponding 2D image points can be found by having multiple images of a known calibration object. The most commonly used object is the checkerboard where the number of squares and their size is known.
- **Self-calibration**: no calibration object is used. The camera is moved around while the scene remains static. The correspondence between the three images is used to estimate the intrinsic and extrinsic parameters.

Zhangs method

Zhang's method is a calibration technique developed by Zhengyou Zhang at Microsoft Research [35]. This method lies between the two main techniques described above, exploiting the advantages of both methods. The only requirement is that the camera has to observe a planar pattern, typically a checkerboard, in at least two different orientations. The algorithm then detects feature points in the images. Since the plane z = 0 is given by the pattern itself only 2D metric information is used. The linear transformation between planes is computed and by using singular value decomposition an initial estimation of the intrinsic parameters is computed. Further, applying the Levenberg-Marquardt algorithm to the algebraic error the parameters are refined. The same approach is used to make an estimation of the extrinsic parameters. Lens distortion also needs to be dealt with. An estimation of the radial distortion parameters is calculated by comparing the real pixel values and the ideal ones given by the pinhole model.

2.2 Stereo Camera

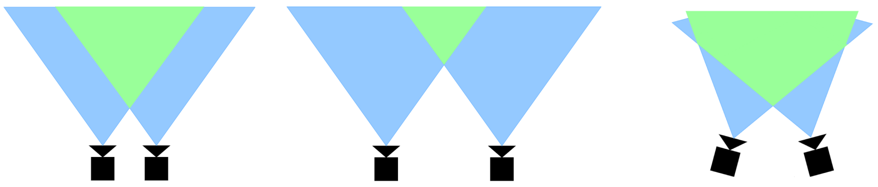
The pinhole model allows for a reconstruction of a pixel coordinate into a world point, but using a monocular camera one cannot without having prior knowledge of the world perceive the accurate depth of an object. The stereo setup allows two cameras to simulate the human binocular vision, and recover a 3D structure of a scene.

Retrieving depth information from a stereo camera is based on the principle of stereo geometry. When pictures are taken with the two cameras the scene is captured in two different viewpoints. To obtain the 3D position of the scene the stereo geometry needs to be known and for each point x in one camera the corresponding point x' in the other needs to be located. This is called the correspondence problem. To solve the correspondence problem and retrieving the 3d world point the following steps needs to be taken

- Find a suitable stereo setup
- Calibrate and rectify
- Find matching points in the two images
- Make a disparity map
- Make a 3D reconstruction from the disparity map

2.2.1 Stereo setup

When deciding the stereo setup it is important to choose the appropriate hardware setup according to the application area. A good starting point is to outline the area where you want to look. The cameras should be set up so that their field of view covers that area.



(a) Small baseline **(b)** Large baseline **(c)** Angled

Figure 2.5: Stereo setup

When deciding the field of view of the system it is important to consider the area the system is going to be implemented, how large an area it is necessary to capture in combination with the size of the desired object to be detected. The field of view of the system is directly linked to the distance between the two cameras, the baseline, and the orientation the cameras has according to one another, the vergence angle [26]. Figure 2.5 shows some possible setups, where the field of view for the matching image is marked in green. It illustrates how the resulting field of view is influenced by the size of the baseline and the orientation of the cameras. From this one can conclude that the distance one can detect objects, the field of view and the blind spots are correlated and dependent on the setup.

The pixel accuracy is also dependent on the baseline. From figure 2.6 one can see that when the baseline is small several points will project to the same pixel. Thus with a small baseline the depth error will be larger. A larger baseline will have higher accuracy but with

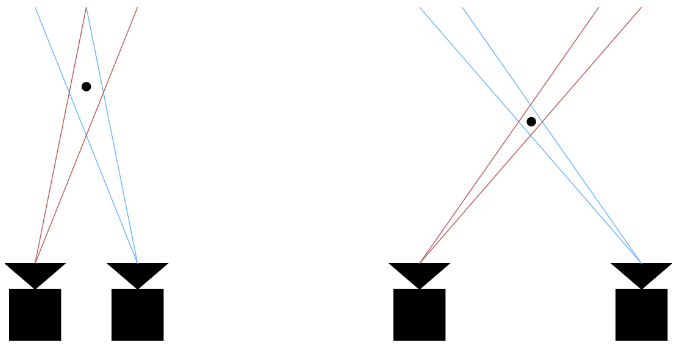


Figure 2.6: Pixel accuracy dependent on baseline

a larger field of view searching for corresponding pixels will become more difficult. The level of depth-accuracy the system will achieve is dependent on the size of the baseline, the field of view and the blind spots.

It is only the area captured in both of the images that are considered for further processing. As a rule of thumb $\frac{2}{3}$ of each image should overlap the other. If this is not the case only a small part of the images captured provide useful information. Thus the cameras should be placed accordingly. The overlap is dependent on both the baseline and the angle between the cameras.

The depth accuracy achieved is dependent on the baseline. The size of the baseline is directly proportional to the distance at which one is able to detect objects. Thus stereo systems with a small baseline are well suited for detecting objects at short distances, while a larger baseline is better for detecting objects at greater distances. There is a rule of thumb saying that the relation of the baseline and the distance at which the depth is the most accurate perceived is $\frac{1}{30}$. E. g. when the baseline is of 1 meter the best depth-accuracy is achieved at 30 meters [5].

Angling the cameras towards each other will influence the field of view and where the focus lies. Having the axis through the optical centers intercept at the desired operating distance will make that the distance at which the cameras are in focus.

It is also crucial to have a sense of the optics of the two cameras. When processing the images from the stereo camera different optics will output different results. To ease the matching of the images it is common to use the same type of cameras. One thing to consider is the resolution of the camera. To accurately measure the depth the images should have multiple pixels on the surface of the objects. The operating distance and the field of view of the camera are dependent on the focal length. Both resolution and focal length should be taken into consideration when selecting the camera sensor.

$$f = \frac{h * OD}{FOV}, \qquad \begin{aligned} f &= \text{focal length} \\ h &= \text{horizontal sensor size} \\ OD &= \text{operating distance} \\ FOV &= \text{horizontal field of view} \end{aligned}$$

2.2.2 Epipolar geometry

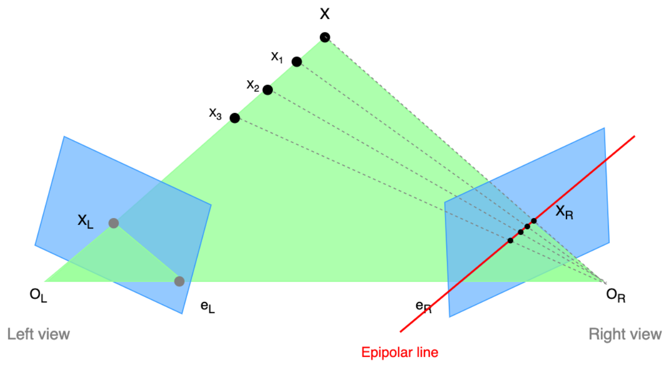


Figure 2.7: Epipolar geometry

The geometry of stereo vision is called the epipolar geometry and is illustrated in figure 2.7. The green triangular connecting the two optical centers and the scene point X is called the epipolar plane. The baseline is the line connecting the two optical centers, O_L and O_R. The points where the baseline intersects the image planes, e_L and e_R are called the epipoles. The lines between the epipoles and the points X_L and X_R are called the epipolar lines. The epipolar lines is where the epipolar plane intersects the image planes. When a scene point X is given the projected image points x_l and x_r must lie on the corresponding epipolar line. This is called the epipolar constraint. Exploiting this makes the correspondence problem go from a region search to a search along one line.

The epipolar constraint can be represented mathematically by the essential matrix, $\mathbf{E}$, and the fundamental matrix, $\mathbf{F}$. The essential matrix represents the constraint in the normalized image coordinates while the fundamental matrix represents it in pixel coordinates. A correspondence between the points x_L and x_R is represented in normalised image coordinates as

$$\mathbf{x_L E x_R} = 0$$

The essential matrix is given by the following equation in [14]

$$\mathbf{E} = [\mathbf{t^L_{LR}}]_x \mathbf{R_L f}$$

This is related to the pose between the left and right image planes in figure 2.7 which can be given by the rigid body transformation

$$\mathbf{T_{LR}} = \begin{bmatrix} \mathbf{R_{LR}} & \mathbf{t^{L}_{LR}} \\ \mathbf{0} & 1 \end{bmatrix}$$

This can be extended to pixel correspondences between u_L and u_R via the related camera calibration matrices K_L and K_R of each camera. The epipolar constraint can now be represented by the fundamental matrix as

$$\mathbf{u_L}^{\top}\mathbf{F_{LR}}\mathbf{u_R} = 0$$

The fundamental matrix is related to the essential matrix and it can be written as

$$\mathbf{F_{LR}} = \mathbf{K_L}^{-\top}\mathbf{E_{LR}}\mathbf{K_R}^{-1}$$

2.2.3 Calibration and Rectification

Even though the pose and distance between the cameras in the stereo setup can be measured it is desirable to find an exact model of the stereo camera. This can be obtained through a stereo calibration. Calibrating a stereo camera is the process of estimating the essential and fundamental matrix. This allows for the rotation and translation of one camera according to the other to be known.

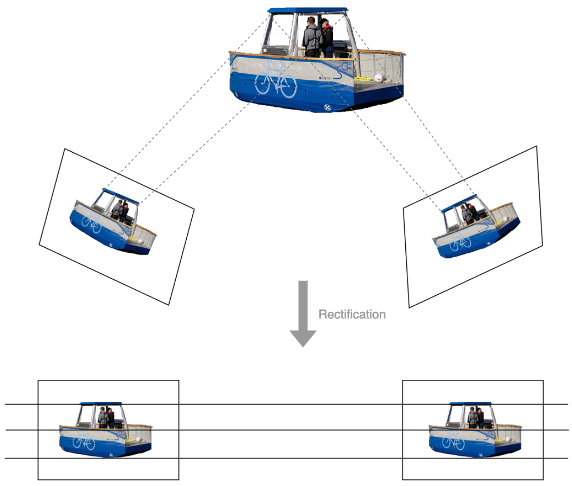


Figure 2.8: Image rectification

When the rigid body transformation between the two cameras is known this can be used to rectify the images. Rectification is the process of reprojecting the images such that the image planes are row aligned and coplanar [18]. The result of the process can be seen in figure 2.8. Having two rectified images makes the correspondence problem easier. Looking for corresponding pixels in the two images is now a 1D problem. From the figure one can see that the corresponding points in the images are aligned in the same straight leveled line that can be drawn between the two images.

2.2.4 Stereo matching

A stereo matching algorithm needs to be implemented to find the corresponding points in two rectified images. The different approaches are usually divided into two methods; local and global methods.

In local methods a small number of pixels surrounding the pixel of interest are considered. This is usually referred to as block matching. A small region of pixels is taken from one of the images and a search for the closest matching region in the other is performed. The search starts at the same location as where the block was taken, and moves to the left and right up to a maximum distance. To compare the image regions a cost function is needed.

In global methods a scan-line or even the whole image is taken into consideration [20]. The methods use optimization techniques such as dynamic programming and graph cuts. Solving the correspondence problem using a global approach makes the matching process more robust to among others uniform textures and occlusions which local methods are sensitive to. The disadvantage on the other hand is that the methods are usually computational heavy and slower than local methods.

2.2.5 Disparity map

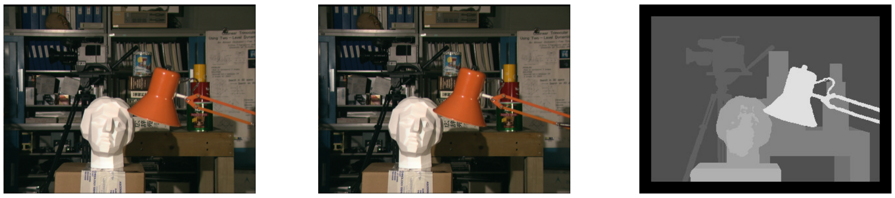


Figure 2.9: Left and right stereo image and the resulting disparity map given by [11]

Disparity map is a representation of the motion between the two images taken by the stereo camera [11] and it is calculated based on the distance between the corresponding pixels. It can also be called a range map. Figure 2.9 shows an example of what a disparity map can look like. The darkest colors represent pixels with low disparity, high depth, and the brightest pixels represents pixels with high disparity, low depth.

The depth, z, of each point in the disparity map can be calculated by

$$z = \frac{b \cdot f}{d} \tag{2.13}$$

where b is the baseline, f the focal length and d the disparity.

2.3 LiDAR

Light Detection and Ranging (LiDAR) is an active remote sensing system. The LiDAR emits a laser pulse and the reflected light energy is returned and recorded by the sensor. The system measures the time it takes for the emitted pulse to be reflected to the sensor. The travel time is used to calculate the distance to the object that reflects the light. This is done by using the formula stated in equation 2.14. [34]

$$\text{Distance} = \frac{\text{Speed of light} \cdot \text{time of flight}}{2} \tag{2.14}$$

A LiDAR consists of lasers, scanners and photodetectors. The laser emits the laser light and the photodetector records the echoed light. The scanner then processes the distances and angles into the system. The speed of developing pictures depends on how fast the scanner is. [31]

There are different types of LiDARs, but in this project the focus will be on 3D LiDARs. A 3D LiDAR measures azimuth and elevation of the reflection in addition to the range. This gives us a 3D point in spherical coordinates, see illustration in figure 2.10. In the figure φ is the azimuth angle, θ is the elevation angle and r the distance from the point to the LiDAR. A Cartesian point is thus given by

$$\begin{aligned} x &= r\cos(\theta)\sin(\varphi) \\ y &= r\cos(\theta)\cos(\varphi) \\ z &= r\sin(\theta) \end{aligned} \tag{2.15}$$

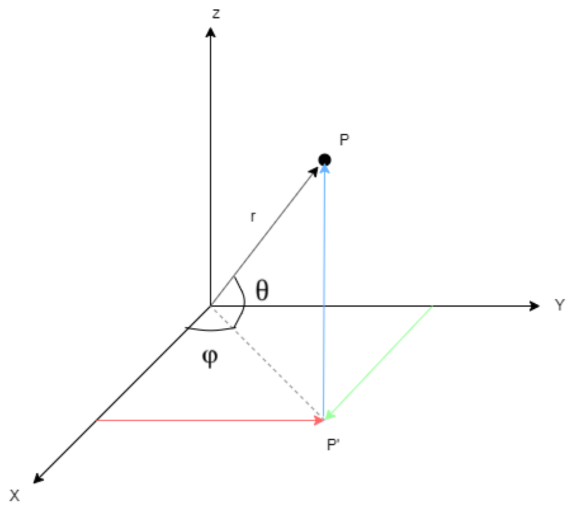


Figure 2.10: Spherical coordinates

The output of the LiDAR called a point cloud and it contains 3D points in spherical coordinates.

2.3.1 Ground Truth

Ground truth refers to the accuracy of the data collected. In depth estimation the ground truth is the real depth and tells how accurate the depth estimation is. When using stereo vision to acquire depth information a good ground truth can be obtained from a 3D LiDAR. The LiDAR is considered as a highly accurate sensor and is adequate as ground truth for long distances. Comparing the points obtained by the stereo camera with the ones obtained by the LiDAR gives a good estimate of how exact the stereo depth estimates are.

Chapter 3

Depth estimation and comparison

The objective of the project is to be able to compare depth estimates from the stereo camera with the LiDAR. The data produced by the LiDAR include an array of 3D points of the observed surroundings, a point cloud. For the stereo camera to produce 3D points a matching function must be implemented in order to produce a disparity map. To compare the depth estimates some operations on the obtained point clouds must be performed. The LiDAR is considered the ground truth and thus the accuracy of the stereo camera is evaluated based on the LiDAR data.

3.1 Stereo matching algorithms

As mentioned in 2.2.4 stereo matching algorithms can be divided into local and global methods. The global algorithms performs better in terms of accuracy but are generally less efficient than the local ones. For autonomous vehicles real-time performance is essential. Thus local algorithms are more suitable.

3.1.1 Sum of Absolute Differences

In this project block matching is used with the Sum of Absolute Differences (SAD) algorithm as a cost function. The local algorithm is used due to its simplicity and low computation time.

$$SAD(d) = \sum_{(u,v) \in W_m} \mid I_L(u,v) - I_R(u-d,v) \mid \tag{3.1}$$

Equation 3.1 states the SAD algorithm where $I_L(u,v)$ is the left image point's intensity, I_R the right image point's intensity and d the pixel disparity [28]. To compare two blocks the absolute value of the difference between the corresponding pixels in the blocks is computed. These values are summed together and are used as an estimate of how similar the blocks are. The lower the value the more similar the blocks are.

3.1.2 Semi-Global Matching

Semi-Global Matching is a method that takes advantage of both local and global properties. The idea is to have a method that is more accurate than local methods and faster than global methods. A Semi-Global Matching approach introduced by Hirchmüller [17] matches each pixel based on Mutual Information and the approximation of a global smoothness constraint. Mutual information is defined by the entropy of the two images which is calculated from the probability distributions of the intensities. Mutual information thus acts as a cost function as to whether pixels match or not. It is calculated for each pixel and it is not sensitive to illumination changes. A hierarchical approach is used in order to decrease runtime. Calculating the cost pixel-wise can often lead to wrong matches, due to for example noise. For this reason a penalization of changes of neighboring disparities is added. The cost aggregation approximates a global smoothness constraint. The matching costs are connected with one another by calculating cost paths. Thus the cost of a pixel is calculated by the sum of all 1D minimum cost paths leading to that pixel. The matching pixels are the ones that correspond to the minimum cost. The method uses a median filter with a small window to remove outliers.

3.2 Point cloud operations

In order to get an accurate comparison of the depth estimation obtained by the two sensors some operations on the obtained point clouds must be performed.

3.2.1 Nearest Neighborhood Search

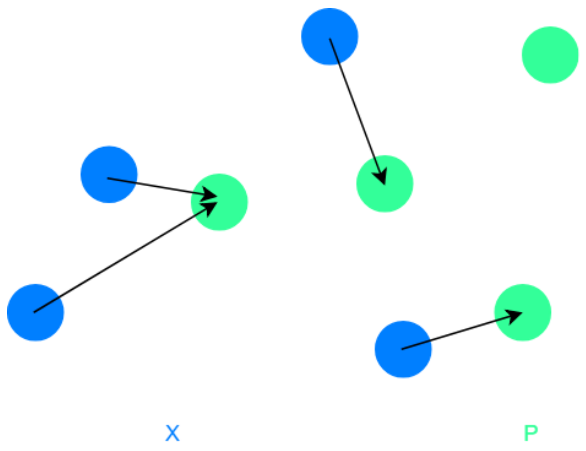


Figure 3.1: Nearest neighbour

When using the LiDAR as ground truth it is crucial to be able to compare the 3D points given by stereo camera to the LiDAR. This can be achieved by calculating the distance between a point given by the stereo camera and the nearest point given by the LiDAR.

Thus it is essential to be able to find the point in a point cloud that is closest to a given point. This is called a nearest neighborhood search and is illustrated in figure 3.1.

Finding the nearest neighbor of a point in a point cloud is the operation of finding the point that yields the smallest distance. The metric distance between two points is simply given by the difference of the points. Thus the nearest point, $\vec{y}$, of a reference point cloud, X, to a point, $\vec{p}$, is given by the solution of

$$\|\vec{p} - \vec{y}\| \leqslant \|\vec{p} - \vec{x_i}\| \tag{3.2}$$

where $\vec{x_i}$ is a point in X. The search can be made efficient by using a k-d tree approach, for more information see [12].

3.2.2 Iterative Closest Point

The point clouds produced by the sensors are given in different coordinate frames. Thus an extrinsic calibration of the two sensors must be performed.

The Iterative Closes Point (ICP) algorithm by Besl and McKay [4] calculates the rotation and translation that minimizes the Euclidean distance between two point clouds. A base cloud, P, and a reference cloud, X. ICP is independent of representation, it utilizes a shape matching metrics that is both global and local.

First, the closest point in the reference cloud must be found for each point in the base cloud. This can be solved by performing a nearest neighborhood search on each point in the base cloud. When the corresponding points in the two point clouds is known the rotation, R, and translation, t, between the point clouds can be calculated. This is done by minimizing the error

$$e(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|\vec{x_i} - R\vec{p_i} - t\|^2$$

where N_p is the number of points in the base cloud P, $\vec{x_i}$ is a point in the reference point cloud X and $\vec{p_i}$ is a point in the base cloud P [4].

To find the optimal rotation and translation the centroids of the point clouds must be found. The centroid of a point cloud is the average of the points and can be calculated as follows

$$\vec{\mu_p} = \sum_{i=1}^{N_p} \vec{p_i} \qquad \vec{\mu_x} = \sum_{i=1}^{N_x} \vec{x_i} \tag{3.3}$$

The cross-variance matrix of the point clouds is given by

$$\Sigma_{px} = \frac{1}{N_p} \sum_{i=1}^{N_p} [(\vec{p_i} - \vec{\mu_p})(\vec{x_i} - \vec{\mu_x})^\top] = \frac{1}{N_p} \sum_{i=1}^{N_p} [\vec{p_i}\vec{x_i}^\top] - \vec{\mu_p}\vec{\mu_p}^\top \tag{3.4}$$

The optimal rotation can be found by finding the maximum eigenvalue of the following matrix

$$Q(\Sigma_{px}) = \begin{bmatrix} \mathrm{tr}(\Sigma_{px}) & \Delta^\top \\ \Delta & \Sigma_{px} + \Sigma_{px}^\top - \mathrm{tr}(\Sigma_{px})\mathbf{I_3} \end{bmatrix} \tag{3.5}$$

where $\mathbf{I_3}$ is the 3×3 identity matrix and Δ is given by $[A_{23} \quad A_{31} \quad A_{12}]^\top$ where $A_{ij} = (\Sigma_{px} - \Sigma_{px}^\top)_{ij}$ [4].

When the rotation is found the translation is given by

$$t = \vec{\mu_x} - R\vec{\mu_p} \tag{3.6}$$

3.3 Depth comparison

Two methods for comparing depth information from the stereo point cloud with the LiDAR point cloud is proposed.

3.3.1 Depth comparison using a nearest neighborhood search

In theory the stereo camera produces a flat point cloud of objects with flat surfaces. When aligning the point cloud of the LiDAR and the stereo camera, a point in the LiDAR will have its corresponding point in the nearest point produced by the stereo point cloud. Thus when selecting points in one of the point clouds the corresponding points in the other can be found by performing a nearest neighborhood search as described in section 3.2.1.

The following procedure for comparing depth estimates by the LiDAR and the stereo point cloud is proposed

1. Perform an extrinsic calibration of the two sensors
2. Transform the LiDAR point cloud into the stereo reference frame
3. Visualize the two point clouds in the same plot
4. Select points from one of the point clouds one want to compare the depth of the other with
5. For each extracted point search for the nearest neighbour in the other point cloud
6. Calculate the distance between the corresponding points

The extrinsic calibration is performed by using the ICP algorithm as described in section 3.2.2.

3.3.2 Depth comparison using manually selected points

It is important to notice that the nearest neighborhood search only returns the corresponding point in the other point cloud if they are aligned and the distance between the point clouds is reasonable. If the point clouds are too far apart noise could appear between them and the nearest neighbor might be a different point than the desired corresponding point. If that is the case corresponding points should be selected manually for both point clouds. Thus an evaluation after step 3 must be made. If a nearest neighborhood search is deemed unreasonable the following procedure is proposed

1. Perform an extrinsic calibration of the two sensors

2. Transform the LiDAR point cloud into the stereo reference frame
3. Visualize the two point clouds in the same plot
4. Select points in both point clouds that corresponds to roughly the same place on the object.
5. Calculate the distance between the corresponding points

Chapter 4

Hardware and Software

The system is designed so that it can be taken in use on the autonomous ferry Milliampere. To do that some considerations need to be taken into account. This chapter describes the sensors used in this project and what considerations that need to be made in order to integrate them into the system. Furthermore, the software used for the implementation of the point cloud processing, the calibration of stereo camera, LiDAR-camera calibration and disparity map is presented.

4.1 Hardware

4.1.1 Camera sensor

Figure 4.1: Blackfly S GigE cameras from [10]

For the camera sensor two cameras of the model Blackfly S GigE from FLIR are used together with lenses of the model 8.5mm C Series Fixed Focal Length Lens from Edmund Optics. The datasheet of the camera and lens can be found in [10] and [7] respectively.

The Blackfly S GigE cameras are high-performance machine vision cameras that are easy to use. They are compact sensors in the form of an ice cube. The specifications of the camera are stated in table 4.1. A resolution as high as 2248×2048 makes the sensor very

Frame rate	24 FPS
Resolution	2248×2048
Pixel Size	3.45 μm
Sensor Type	CMOS
Sensor Size	$2/3''$
Readout Method	Global shutter
Machine Vision Standard	GigE Vision v1.2
Operating Temperature	0° C to 50° C
Exposure Range	13 μ s to 30 s
Dimensions (W x H x L)	29 mm x 29 mm x 30 mm
Power Requirements	Power over Ethernet (PoE), or 12 V nominal (8 - 24 V) via GPIO

Table 4.1: FLIR Blackfly S GigE specifications given by [10]

attractive for the use of object detection at long distances. Taking into consideration the different qualities presented in section 2.2.1 the Blackfly S GigE camera is a good choice. Having a GigE, Gigabit Ethernet, interface makes it easy to connect several cameras. It provides fast transfer over ethernet cable.

The lens has a focal length of 8.5 mm. It allows for iris and focus to be adjusted manually and it can be tuned in code. When the lens is used together with a camera with a sensor size of $2/3''$ the resulting field of view is given in the datasheet of the lens and it is

$$FOV = 59.1^{\circ}$$

A field of view of that size is very satisfactory and makes it possible to see most of the area in front of the ferry. This also gives more freedom in deciding the stereo setup because there will be much overlap in the field view of the two cameras.

A drawback of the cameras is the operating temperature. In a city like Trondheim this is not feasible when for large portions of the year the temperature is below zero. Testing outside is thus dependent on the weather.

4.1.2 Stereo setup

The setup of the stereo camera was designed so that it could fit on the ferry Milliampere. Placing the cameras below the sensor rig already installed on the ferry gives a space of two meters where the cameras can be installed. To exploit the space available a mount of 2 meters was designed where the distance between the cameras and their pose can be adjusted, see figure 4.2.

Considering the rule of thumb presented in section 2.2.1 a distance of 2 meters between the cameras allows for objects 60 meters away to be captured. This is feasible for such a slow moving ferry and thus the cameras were placed at each end of the mount measuring the exact baseline of 1.75 meters. Seeing as the cameras have such a broad field of view the resulting field of view at 60 meters is approximately 60 meters wide giving a blind spot of only one meter on each side, see figure 4.3. Thus the rule of thumb saying that $\frac{2}{3}$

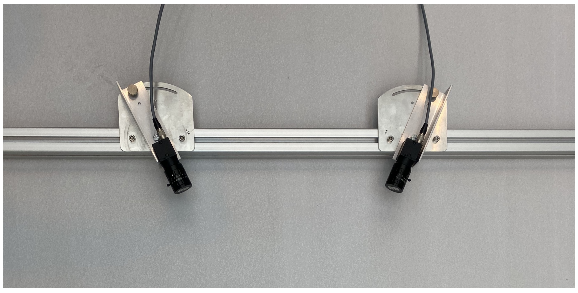


Figure 4.2: Designed stereo setup

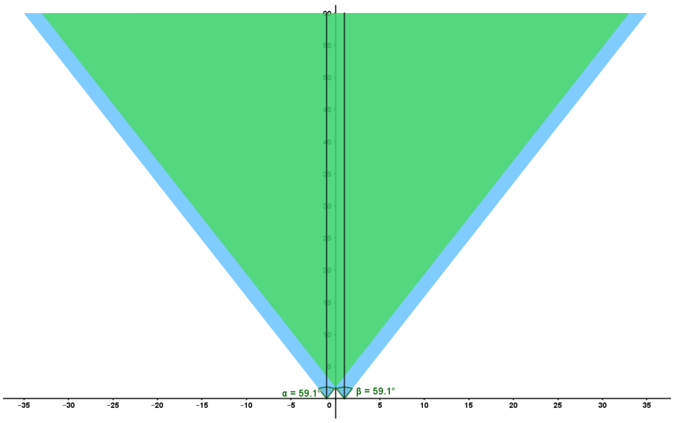


Figure 4.3: FOV stereo setup with baseline of 1.75 m

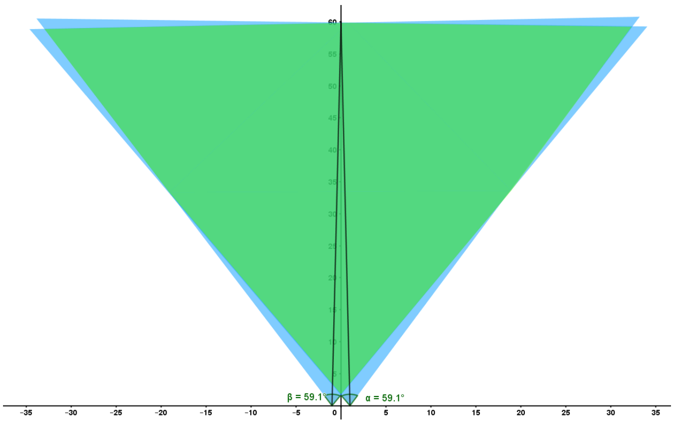


Figure 4.4: FOV stereo setup with baseline of 1.75 m and cameras angled $0.8°$ towards each other

of views must be overlapping is fulfilled. The cameras were angled approximately $0.8°$ towards each other so that the optical axis of the cameras intersect at 60 meters making that the area of focus. The angle was found by using basic trigonometric

$$\tan^{-1}(\frac{1.75/2}{60}) \approx 0.8^\circ$$

See figure 4.4 for the resulting field of view.

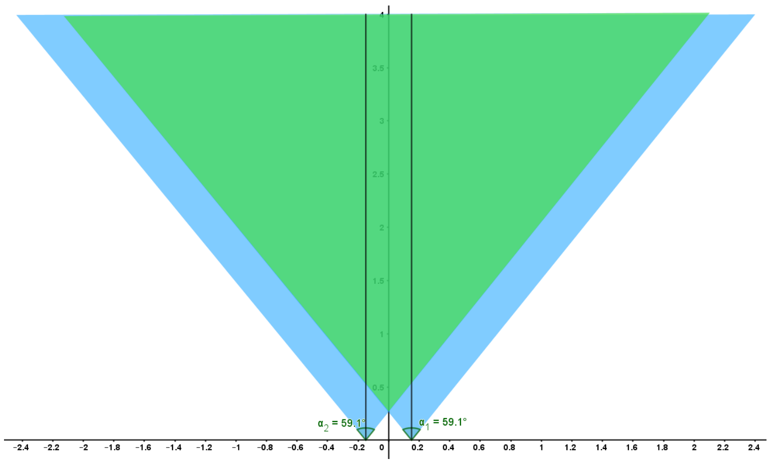


Figure 4.5: FOV stereo setup with baseline of 26 cm

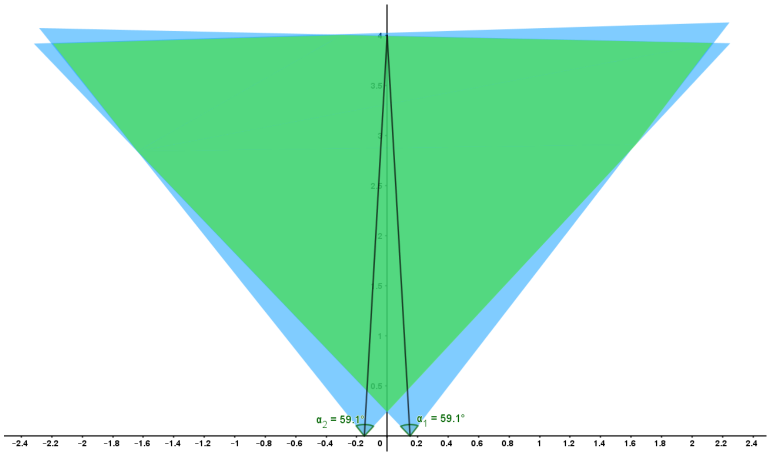


Figure 4.6: FOV stereo setup with baseline of 26 cm and cameras angled 1.86° towards each other

Due to the cameras operating temperature testing outside is not feasible during the winter and most of the autumn. A setup optimal at 60 meters is proven challenging. For this reason an experimental setup was designed for the purpose of this project. For simplicity the focusing distance was chosen to be at four meters. By the rule of thumb the baseline should be around 13 centimeters. Looking at the theory behind the stereo setup in section 2.2.1 a small baseline will give a low accuracy due to several points projecting to the same pixel. For that reason the baseline was set to 26 centimeters. This also makes it possible to place the LiDAR between the two cameras. To make the focusing distance at 4 meters the cameras were angled such that the optical axis intercepts at 4 meters. Again using basic trigonometric the angle of the cameras was computed to

$$\tan^{-1}(\frac{26/2}{400}) \approx 1.86^\circ$$

See figure 4.6 for the resulting field of view. The setup causes the cameras to have a blind sport of approximately 22 cm before the field of view starts.

4.1.3 Synchronized Capture

Synchronized capture is when two or more cameras take pictures at the same time. To retrieve exact depth of moving objects it is crucial that the cameras captures images at the exact same time. To achieve this a master-slave architecture is implemented, where one camera triggers the other. The implementation is done by following the suggested approach in [9]. When the master camera starts capturing an image a strobe occurs, and this is used to trigger the second camera. The frame rate of the secondary camera will now follow the master cameras frame rate. To implement this a physical connection linking the cameras GPIO pins is used.

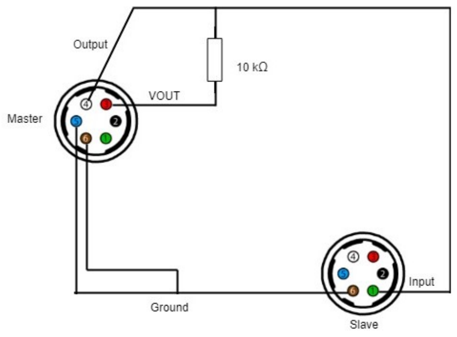


Figure 4.7: GPIO connection circuit

A sketch of the implementation can be seen in figure 4.7. The master's output voltage pin, 3.3 V, is connected to the secondary's input voltage pin. Since the optoisolated output is used the strobe signal from the master camera need to be strengthened. This is achieved by using a pull-up resistor. A 10KΩ resistor is connected to the master camera's non-isolated output voltage and its opto-isolated output to the secondary camera's non-isolated input. Next master's ground is connected to the secondary's ground. The opto-isolated ground on master is also connected to the secondary's ground.

Two Hirose cables were connected as described above. To get the length need between the cameras an extension was added between the cables. A heat shrink tube was added over the connections to make it waterproof, see figure 4.8.

Figure 4.8: GPIO connection circuit

Next, the cameras need to be configured. This is done by using the demo program SpinView available with Spinnaker SDK, FLIR's GenICam3 API library. For the master camera the 3.3 V line is enabled. For the secondary camera the trigger source is set to the input voltage pin and the Trigger mode is set On. The trigger overlap is set to Read Out, which means that the camera will start capturing as soon as the image area has been cleared.

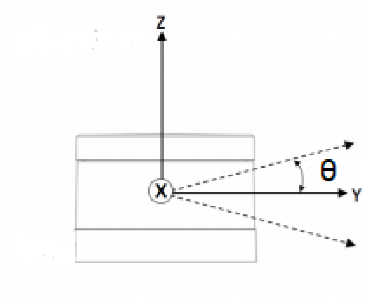
4.1.4 LiDAR

Figure 4.9: Velodyne LiDAR Puck 16

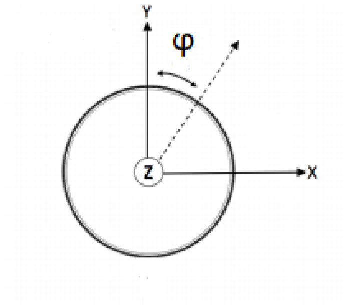
The LiDAR used in this project is the Velodyne LiDAR Puck, VLP 16, see figure 4.9. Its specifications are given in [32]. It is a small, compact LiDAR that has a range up to 100 meters and it has a 360° horizontal and 30° vertical field of view. It has an accuracy of ±3 cm making it attractive both in price and accuracy.

The VLP 16 has 16 laser/detector pairs mounted 2 degrees apart in the vertical direction starting at -15° and ending at 15°. In the horizontal direction the VLP 16 has a resolution between 0.1° and 0.4° and the rotation rate can be chosen as a value between 5 and 20 Hz. The number of measurements obtained by the LiDAR is dependent on horizontal angular resolution and the rotation rate.

It is desirable to represent the spherical points given by the sensor in a Cartesian coordinate system. This is achieved by creating a coordinate frame centered in the LiDAR, see



(a) Side view



(b) Top view

Figure 4.10: The LiDAR coordinate frame

figure 4.10. Here φ represents the azimuth angle and θ the elevation angle. As described in section 2.3 a 3D point from the LiDAR can be represented in the LiDAR frame as:

$$\mathbf{P}_l = \begin{bmatrix} rcos(\theta)sin(\varphi) \\ rcos(\theta)cos(\varphi) \\ rsin(\theta) \end{bmatrix} \quad where \quad \mathbf{P}_l = \begin{bmatrix} x & y & z \end{bmatrix}^\top \tag{4.1}$$

4.1.5 LiDAR setup

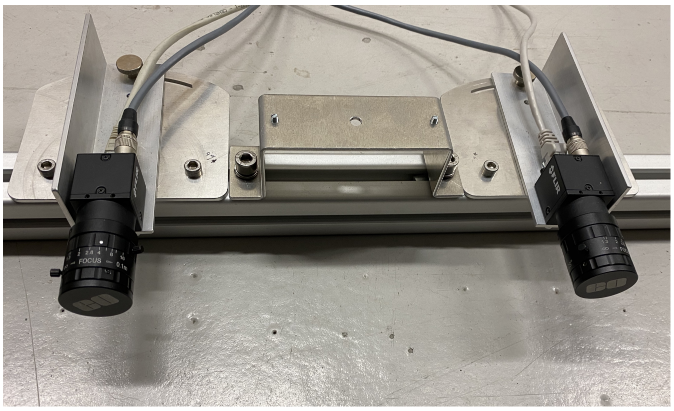


Figure 4.11: LiDAR mount

The purpose of the LiDAR is to serve as ground truth to the stereo camera. Seeing as the LiDAR and the camera should observe the same points the simplest setup is to place the LiDAR in the center between the two cameras. Thus an extra mount was created to be attached to the same bar as the cameras. The LiDAR mount is designed so that the LiDAR can easily be attached and detached. What is important here is that the LiDAR is placed in the same place and orientation every time it is attached. To fulfill these requirements the mount is designed so that the LiDAR can only be placed either facing forward or backward. Figure 4.11 shows the final design of the mount. The two metal screws are placed to fit with the two holes underneath the LiDAR making sure it can only be placed in a certain orientation.

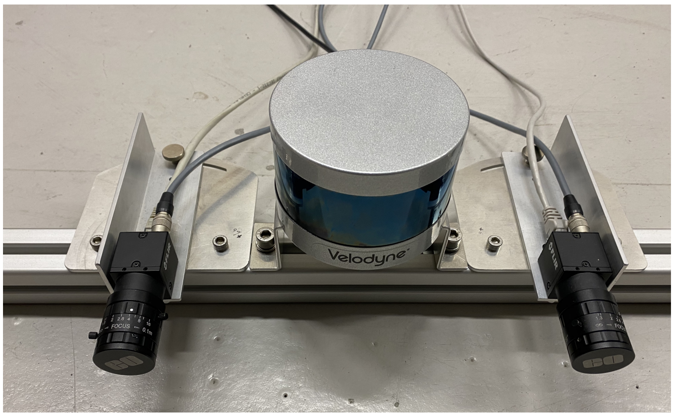


Figure 4.12: Complete setup of stereo camera and LiDAR

The complete setup of the LiDAR together with the cameras is shown in figure 4.12

Connection

The VLP 16 primary interface is its Ethernet interface. All the data, command and control are transmitted over it. To integrate it in the system together with the stereo setup the LiDARs Ethernet cable needs to be connected in the same switch at the cameras. The LiDAR does not get its power through Ethernet as the cameras, it has its separate power supply. Ethernet is only used for data transfers and command and control. To gain access to the data from the LiDAR a network need to be set up as an interface. When creating the network the LiDARs IP address is set and a static route between the Ethernet port in use and the LiDARs IP is setup.

4.2 Software

4.2.1 Robot Operating System - ROS

ROS is an open-source framework with a collection of tools and libraries [1]. It provides device drivers and communication between multiple processes. It is in this project used to collect and organize data. Ubuntu Linux is a supported operating system. This work is done as a part of the autonomous ferry ”Milliampere”. To be compatible with the already existing system on the ferry ROS Kinetic was downloaded on a computer running on Ubuntu 16.04 LTS. The main languages for writing ROS code is `C++` and Python.

A node in ROS in an executable that is connected to the ROS network. The nodes communicate using publishers and subscribers for sending and listening to messages. ROS also provides a built-in command-line tool for recording and playing published messages, called rosbag. This makes it possible to store data from experiments. When playing the stored data the sensor outputs and inputs are simulated without actually having the sensors present in the system.

Drivers

ROS provides open source packages for different sensors. The `spinnaker_sdk_camera_driver` and the `velodyne_driver` driver is used for the FLIR cameras and the LiDAR. The drivers are downloaded from [3] and [2] respectively.

The Velodyne driver collects all the TCP packages sent over the ethernet cable from the LiDAR into the computer. A message containing the raw data from one scan from the LiDAR is published i.e. all points captured when the 16 lasers do a $360°$ scan. The raw data is represented in a point cloud format. The point cloud message is of type `PointCloud2` which stores the height and width of the point cloud and the intensity of each point in it. This message is easily converted into a 3D Cartesian representation using built-in functions. This results in an array of 3D points. Once the LiDAR is turned on it will publish data continuously and it is thus not triggered by the user. The camera driver is a driver for running cameras that uses the Spinnaker SDK. As mentioned in section 4.1.3 Spinnaker SDK is a API Library by FLIR and it provides the interface with the camera. It needs to be downloaded on the computer running the driver. The camera driver also collects data sent over an ethernet cable. It reads the image stream sent through GigE ports and publishes the raw data of a single frame of a video stream. The driver allows for different parameters to be set when launching the code. Exposure time, the number of frames to save, location of where to save the images and whether to use color or not are just some of the parameters that can be set. The driver provides code for launching several cameras at once. The ID of both cameras is specified in the code along with which camera is the master and which one is the slave. Thus by using the tools provided by the driver a subscriber that listens for data published by both cameras can be implemented. Only the master needs to be triggered when wanting to capture images, the slave is triggered by the master. The driver also labels the frames captured with a timestamp which is helpful in ensuring synchronized capture.

4.2.2 MATLAB

MATLAB's ROS toolbox provides functions making it possible to connect to an already existing ROS network. The function `rosinit('IP_address')` where the ip-address of the ROS network is inserted makes it possible to use the messages published in the network. Connecting MATLAB to the ROS network is especially helpful due to the Toolboxes provided by MATLAB. The Computer Vision Toolbox provides functions and apps helpful for designing 3D vision. It contains functions for processing point clouds. Different function for denoising, downsampling and removing invalid points are available for preprocessing. It also provides complete methods for registering point clouds where the preprocessing, registration and alignment and stitching are all done by the same function. This makes MATLAB a suitable environment to perform the extrinsic calibartion of the LiDAR and the stereo camera.

The Computer Vision Toolbox also provides apps for camera calibration, both monocular and stereo calibration. The Camera Calibrator app is deployed to estimate a single camera's intrinsic, extrinsic and lens distortion parameters. The app implements the calibration algorithm by Zhang, see section 2.1.3. Uploading images of a calibration pattern in different orientations the app calibrates and provides tools for evaluating the calibration. Parameters or images can be updated in order to improve the calibration. The Stereo Camera Calibrator App is deployed to estimate the rigid body transformation between the two cameras. The result of the single camera calibration is used to account for distortion. The app requires one to upload images of a calibration pattern in different orientations taken simultaneously by the two cameras. As the Camera Calibrator App this app also provides tools for evaluating and improving the calibration.

4.2.3 OpenCV

Open Source Computer Vision Library, OpenCV, is a library of programming functions for computer vision applications. The library provides among other functions for processing the images captured by a stereo camera. It has interfaces for both MATLAB and C++ and it supports Linux. It is supported by ROS by adding it as a dependency and including it in the build process of the nodes where it is used. Through the Computer Vision Toolbox MATLAB provides an interface to OpenCV and thus making several functions and methods available for processing of stereo images and disparity maps.

4.2.4 Disparity maps

Functions for creating disparity maps from stereo matching algorithms is available through MATLAB's Computer Vision Toolbox.

The block matching method using SAD for pixel matching is implemented using the function `disparityBM(I1,I2,Name,Value)` where $I1$ and $I2$ are the left and right image of a stereo camera [21]. The images need to be rectified before they can be used as input for the function. When the stereo parameters obtained from the stereo calibration are known the function `[J1,J2] = rectifyStereoImages(I1, I2, stereoParams)` can be utilized in order to obtain the rectified stereo images. The inputs $name$ and $value$ represents different parameters to be set. To achieve an acceptable disparity map a proper tuning of the parameters is necessary. The parameters available for tuning are given in table 4.2.

A method for computing disparity map through Semi-Global Matching, see section 3.1.2, is implemented using the function `disparitySGM(I1,I2,Name,Value)` [22]. The function requires rectified stereo images as input. First, it calculates the Consensus Transform of the images. The consensus transform calculates whether a pixel has a smaller or higher intensity than each of its neighbors. Next the difference in bit position between pixels of the consensus transform is calculated, the Hamming distance. The disparity values are calculated based on Semi-Global Matching, see section 2.2.4. The function requires two parameters to be set. The parameters available for tuning and their interpretation is given in table 4.3.

Disparity range	The range of disparity. Input as vector with two elements $[maxDisparity, minDisparity]$
Blocksize	Size of the search block. How big of a neighbourhood to consider when finding corresponding pixels. Input as an odd integer.
Contrast Threshold	Scalar value in the range of $(0, 1]$. If a pixels contrast value is below the range the disparity is marked as unreliable
Uniqueness Threshold	Specifies how unique the value of the pixel has to be, how big the computed SAD value has to be. Input as a integer. Pixels are marked unreliable if $v < SAD \times (1 + 0.01 \times UniquenessThreshold)$ where v is the smallest SAD value of the disparity range.
Distance Threshold	The maximum distance between corresponding pixels. Input as integer.
Texture threshold	Specifies the minimum texture value a pixel can have to be reliable. Input as integer. Pixels below the threshold is marked as unreliable.

Table 4.2: Parameters that can be used as input in the function `disparityBM()` [21]

Disparity range	The range of disparity. Input as vector with two elements $[maxDisaprity, minDisparity]$. The difference between $minDisparity$ and $maxDisparity$ cannot exceed 128.
Uniqness Threshold	Specifies how unique the value of the pixel has to be, how big the computed SAD value has to be. Input as a integer. Pixels are marked unreliable if $v < SAD \times (1 + 0.01 \times UniquenessThreshold)$ where v is the smallest SAD value of the disparity range.

Table 4.3: Parameters that can be used as input in the function `disparitySGM()` [22]

Finding the disparity range

To find the disparity range of the scene the distance between the corresponding points in the two images needs to be known. The distance between corresponding pixels to the objects closes to the stereo camera corresponds to the maximum disparity value. The minimum disparity thus corresponds to the distance between corresponding pixels to the objects furthest away. This distance can be measured by combining the rectified stereo images in a red-cyan anaglyph.

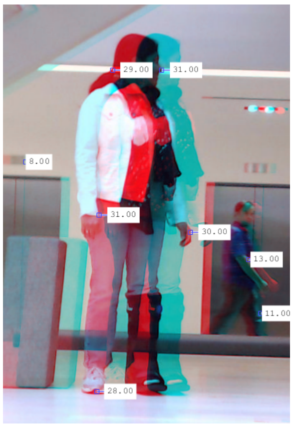


Figure 4.13: Measure disparity in a stereo pair [21]

Figure 4.13 shows how the disparity can be measured when the rectified images are combined in a anaglyph.

4.2.5 Comparing depth in two point clouds

In order to compare the depth obtained by the stereo camera with the LiDAR a point cloud has to be reconstructed from the disparity map. This is done by using functions available by MATLAB. The depth comparison method presented in section 3.3.1 is implemented by performing a nearest neighbor search based on the method presented by Muja and Low in [27]. This requires the point clouds to be given in the same coordinate frame. The package `Click3dPoint` by Babak Taati [29] is downloaded for extracting points in a point cloud by clicking on the desired point. The user can select which points it wants to compare with the other point cloud, then the corresponding point in the other point cloud is found by searching for the nearest neighbor. The distance between the points is outputted.

If using the comparison method presented in section 3.3.2 points from both point clouds is selected using the function `ClickA3DPoint(pointCloud)` from the `Click3dPoint` package. The difference in depth between the corresponding points is calculated.

Chapter 5

Calibration: setup and results

In order to retrieve depth from the stereo camera it needs to be calibrated, i.e. the rigid body transformation between the two cameras need to be calculated. The stereo camera is put together by two monocular cameras that needs to be calibrated individually. This allows for the relation between pixel values and world points to be known and distortion can be accounted for. For the LiDAR to serve as a ground truth a calibration of the LiDAR and the stereo camera must be performed.

5.1 Single Camera Calibration

In order to calibrate the monocular cameras MATLAB's calibration app was used [25]. The calibration app uses the following workflow

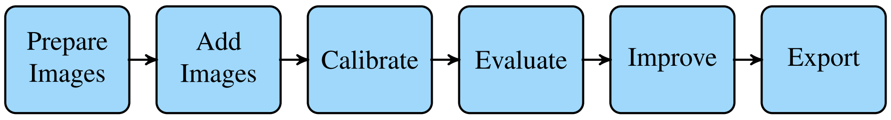


Figure 5.1: Workflow of Matlab Camera Calibration

Before adding images into the MATLAB application, the calibration pattern and the camera setup must satisfy a set of requirements. The calibration application needs images capturing a checkerboard pattern. The checkerboard used in this project is a 5x7 squared checkerboard with each square measured to 10.1cm. It is printed on aluminium to minimize distortion and imperfections/inaccuracy. To obtain an accurate estimation of the parameters it is important to capture images of the pattern at different orientations and in as much of the image frame as possible. The checkerboard should not be angled more than 45 degrees relative to the camera plane. To account for lens distortion it is important that the pattern appear where the distortion is biggest. The lens used causes barrel distortion, i.e. distortion is bigger close to the edges. When capturing images it is important to keep the pattern in focus, i.e. no autofocus, and keep the camera parameters constant. Changing the zoom between images results in miscalculated focal length.

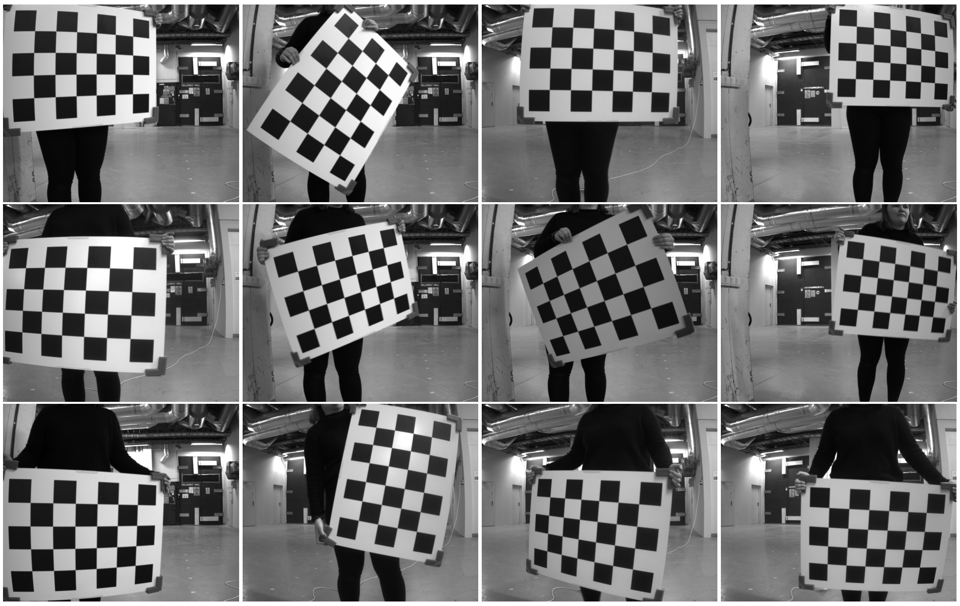


Figure 5.2: Poses of checkerboard

When uploading the images to MATLAB the app attempts to detect checkerboards in each added image and rejects the ones duplicated or with no detected checkerboard. The browser further displays a list of the images with green circles around detected corners.

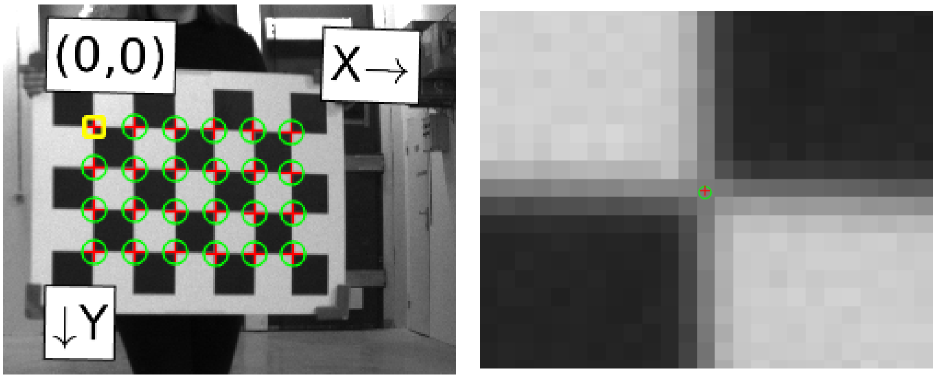


Figure 5.3: Detected corners

In figure 5.3 the green circles are shown in addition to the yellow square indicating the origin. To verify that the corners are detected correctly zoom control is used as illustrated in the figure.

Due to the size of the field of view of the lens two radial distortion coefficients is sufficient. The cameras do not deploy any tangential distortion and thus it is not calculated in the calibration. Some cameras have defects causing the x- and y-axis not to be perpendicular. This can be modeled by a skew parameter. Seeing as though this is not the case for the cameras used here computing the skew is not necessary.

There are several ways to evaluate calibration accuracy, including calculating the reprojection errors, the parameter estimation error, viewing the undistorted images and plotting the relative locations of the camera and calibration pattern.

The projection error is the pixel difference between the measured point and the reprojected point based on the estimated internal and external parameters of the camera. I.e. the error depends on the quality of the estimation of the intrinsic and extrinsic parameters, as well as the quality of the marked (green) points in the image. As a general rule, mean reprojection errors of less than one pixel are acceptable. The calibration app allows you to evaluate the mean reprojection error of each image. Large outliers are removed if they are larger than one or if they don't contribute with necessary poses of the checkerboard. Images that adversely contribute to the calibration should not be removed. As mentioned earlier to obtain a good calibration all poses and positions needs to be covered. To make sure of this MATLAB's 3D extrinsic parameter visualization is a useful tool.

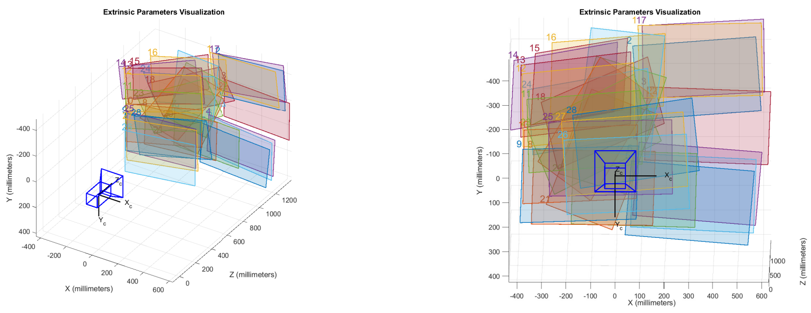


Figure 5.4: Extrinsic parameter visualization of the left camera

Figure 5.4 and 5.5 shows the location of the calibration pattern plotted in the camera's coordinate system for both the left and the right camera. The camera is displayed in dark blue. The captured images cover the whole image plane for both the cameras. If this were not the cases new images of the missing poses would have to be added. The distances from the cameras to the calibration boards seem accurate, and they are all placed in front of the cameras.

After the calibration one should be able to produce undistorted images by using the obtained parameter. Inspecting the undistorted images is thus a good basis for evaluating the calibration. This is an important step even if the reprojection errors are small. If the images covers too small areas the distortion estimation might be incorrect even though the reprojection error is small.

Figure 5.6 shows the resulting undistorted images for both the left and the right camera. A ruler is used in order to verify the accuracy. In the undistorted images the calibration board should be displayed with straight lines.

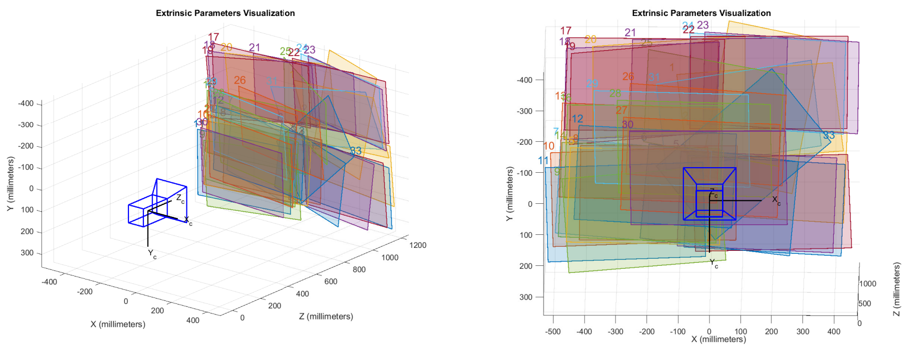


Figure 5.5: Extrinsic parameter visualization of the right camera

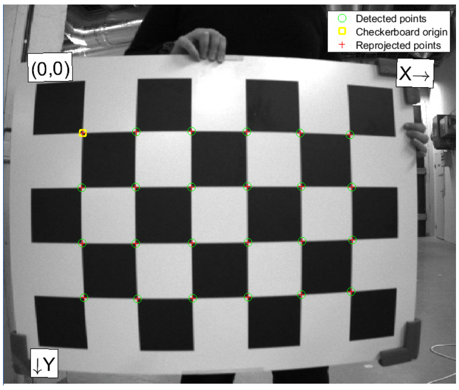
The obtained camera parameters are given in the following equations

$$
\begin{aligned}
K_{left} &= \begin{bmatrix} 1229.37 \pm 0.5 & 0 & 618.3702 \pm 0.3 \\ 0 & 1228.70 \pm 0.5 & 536.6535 \pm 0.2 \\ 0 & 0 & 1 \end{bmatrix} \\
K_{right} &= \begin{bmatrix} 1228.96 \pm 0.7 & 0 & 640.1066 \pm 0.2 \\ 0 & 1228.37 \pm 0.7 & 533.0613 \pm 0.2 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{5.1}
$$

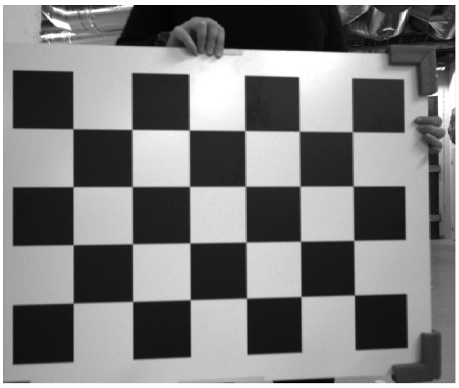
$$
\begin{aligned}
\text{Left cam:} \quad & k_1 = -0.3891 \pm 0.000448, \quad k_2 = 0.1732 \pm 0.0011 \\
\text{Right cam:} \quad & k_1 = -0.3875 \pm 0.000472, \quad k_2 = 0.1681 \pm 0.000984
\end{aligned} \tag{5.2}
$$

The resulting intrinsic parameters and the distortion parameters of both cameras is stated in equation 5.1 and 5.2 respectively. Even though the cameras and the lenses are the same the parameters are not identical. This is due to there being some physical difference even though they are of the same model. It is not possible to produce two identical physical objects. The focal lengths and the distortion parameters of the two cameras are sufficiently equal.

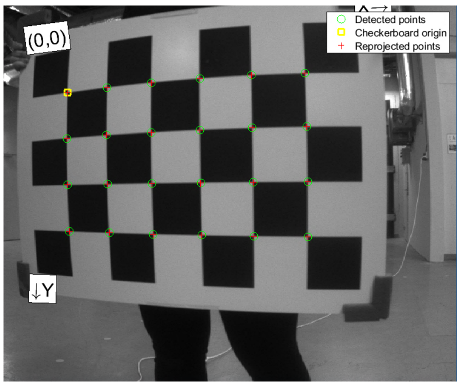
The principal points of the cameras are affected by both the lens and the camera itself. It is influenced by how the lens is screwed on to the camera and what the iris and focus is set to. Seeing as they are set manually a difference between the two cameras is expected. The principal points are given in the upper right corner of the intrinsic matrices. One can see that the principal points differ between the two cameras which is expected.



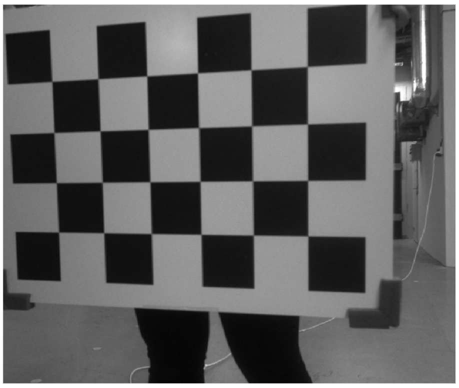
(a) Image from the left camera without accounting for distortion



(b) Image from the left camera after accounting for distortion



(c) Image from the right camera without accounting for distortion



(d) Image from the right camera after accounting for distortion

Figure 5.6: Images of each camera before and after calibration

5.2 Stereo camera calibration

To find the rigid body transformation between the two cameras the Stereo Camera Calibrator App is used. The calibration app uses the same workflow as the single camera calibration app. The same considerations as described for single camera calibration also need to be made for the stereo calibration in addition to some new ones.

The calibration images should be captured at a distance roughly equal to the operating distance [24]. The camera setup is build to be optimized on objects distanced 4 meters away from the camera. Hence, the calibration pattern should be captured at the same distance. There is a rule of thumb saying that the checkerboard should fill at least 20 percent of the captured image [24]. The checkerboard used in this calibration is of size 0.8x0.6 meters. This is somewhat small considering the field of view of the camera. As the checkerboard is smaller than desired a lot more photos are necessary to obtain a good calibration rather than the suggested number of 10 to 20 by MATLAB. Implementation of code and hardware is utilized to continuously and simultaneously capture pictures form both the cameras. With a broad range of pictures the ones noticeably blurry and the ones where the pattern was captured at an angle above 45 degrees were removed before uploading to MATLAB. When preparing images for the stereo calibration it is crucial to capture the stereo images at the same time and the pattern has to be fully visible in both images.

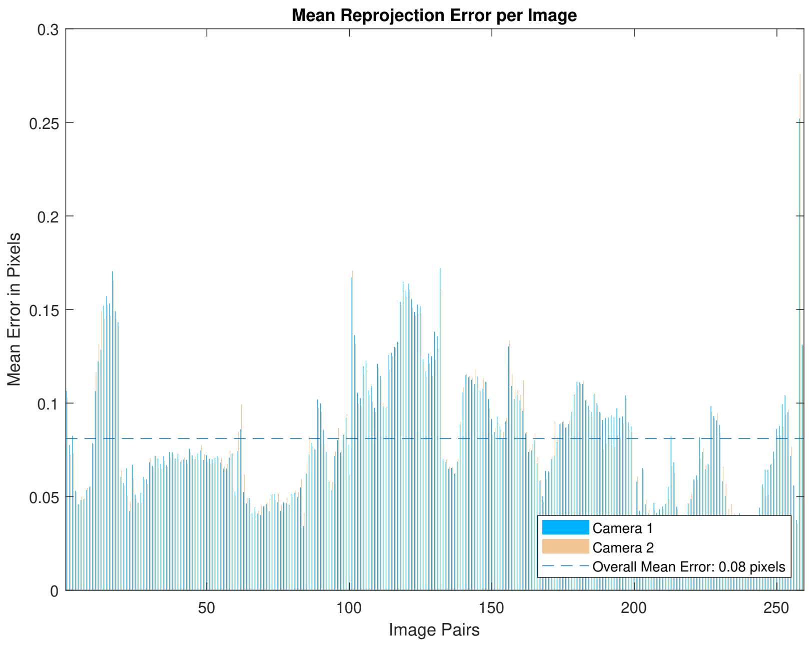


Figure 5.7: Reprojection Error of Stereo Calibration

In figure 5.7, the final result of the reprojection error is displayed. The blue bars show the reprojection error of the left camera while the yellow shows the reprojection error of the right camera. As in single camera calibration the reprojection error needs to be less than one pixel, but here the difference between the reprojection error of the two corresponding images needs to be considered. High spikes that correspond to images that adversely contribute to the calibration are not removed from the calibration. The low mean reprojection error and the approximately equal error between corresponding image pairs makes this an overall satisfying result.

In figure 5.8 the location of the calibration pattern is plotted in the camera's coordinate system. The cameras are displayed in dark blue and red. The relative positions match what was expected and there are no obvious problems. The distances from the cameras to the calibration boards seem accurate, and they are all placed in front of the cameras.

The goal of calibrating the stereo camera is to be able to row-align the image pair. Looking at the rectified images is thus a good basis for evaluating the calibration. In

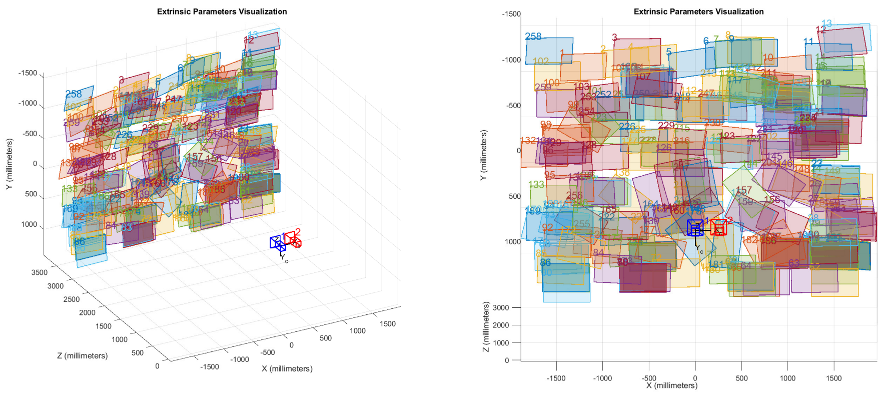


Figure 5.8: Extrinsic parameter visualization of stereo camera

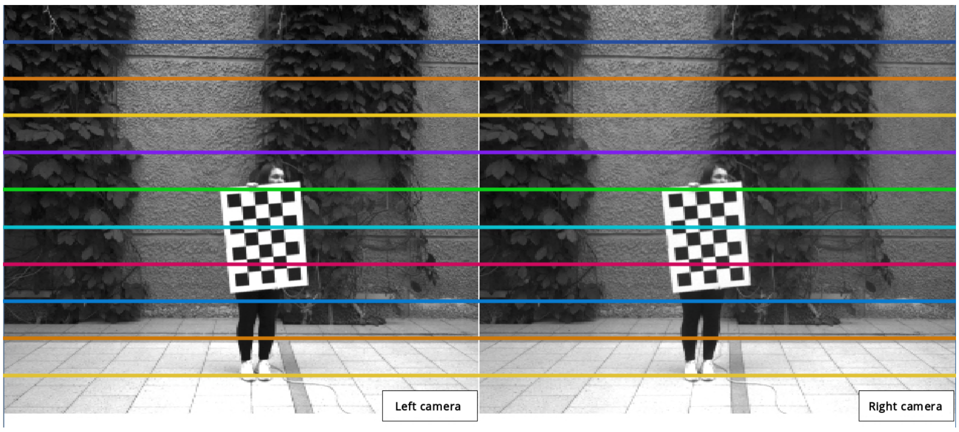


Figure 5.9: Rectified image after stereo calibration

the rectified images pixels of the same world points in the two images should lie along the same line. This can be measured by eye and it is important to evaluate even if the reprojection errors are small. Figure 5.9 shows the final result of a selected image pair after calibrating. One can see that the lines matches points in the two images.

The obtained calculated rotation and translation of the right camera relative to the left is given by the following equations

$$\mathcal{R} = \begin{bmatrix} 0.9986 & -0.00002487 & -0.053019 \\ 0.000057438 & 1 & 0.0006128 \\ 0.053019 & -0.000615000 & 0.998593 \end{bmatrix} \tag{5.3}$$

$$\mathbf{t} = \begin{bmatrix} -260.5121 \pm 0.05 & 2.239238 \pm 0.07 & 6.779297 \pm 0.02 \end{bmatrix} \tag{5.4}$$

Considering the stereo setup presented in section 4.1.2 the cameras are not angled much towards each other thus the rotation matrix given in 5.3 seems reasonable. The translation matrix given by equation 5.4 is stated in in millimeters. The measured distance between the two cameras is 26 cm corresponding to the first entry of the translation matrix. The second and the third entries represent the difference in height and depth respectively. Seeing as the cameras are screwed on the metal mount made for the setup a small difference in both height and depth is to be expected. Overall the final result of the calibration is reasonable.

5.3 LiDAR - stereo camera calibration

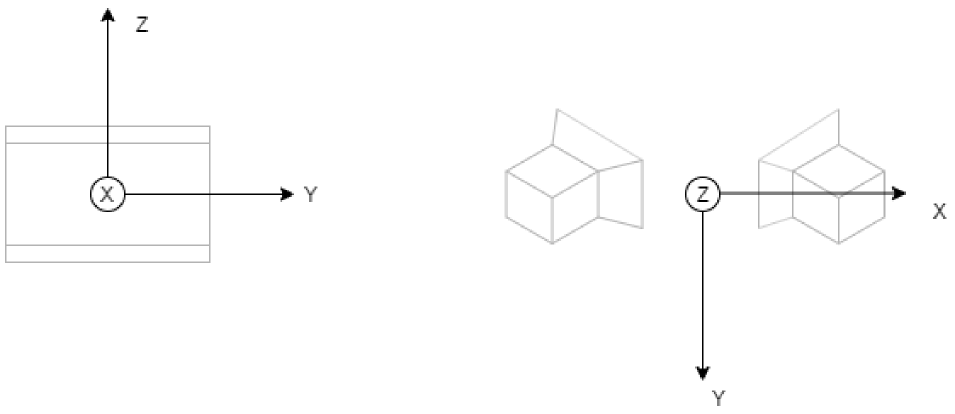


Figure 5.10: LiDAR and camera coordinate frames

To use the LiDAR as ground truth one need to be able to compare a point from the stereo camera with the corresponding point from the LiDAR. As illustrated in figure 5.10 the LiDAR and the camera represent 3D world points in different coordinate frames. In the LiDAR coordinate system the z-axis represents the height, the x-axis the depth and the y-axis the width of a point. While as for the camera the z-axis is used to represent the depth and thus the x-axis represents the width and y-axis the height. The comparison is also dependent on the placement of LiDAR in relation to the camera coordinate frame. It is thus essential to know the rigid body transformation between the stereo camera and the LiDAR. In this project a transformation of coordinate systems is made before utilizing the ICP algorithm described in section 3.2.2 to find the rigid body transformation between the two sensors.

5.3.1 Calibration setup

To obtain the best possible result from both the LiDAR and the stereo camera a large object with variable pattern was chosen as the calibration object. The object needs to be large to ensure enough points are generated by the LiDAR and the appearance needs to be full of variable patterns such that the correspondence problem of the stereo camera can be easily

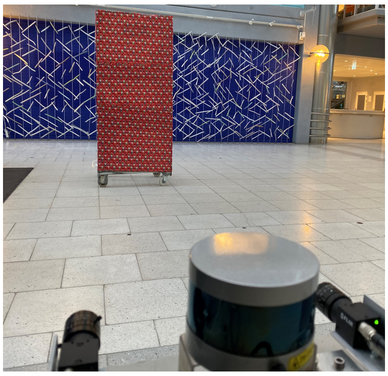


Figure 5.11: Lidar-Stereo camera calibration setup

solved. Considering the sensor setup the object is placed at the distance where the stereo camera can estimate depth with the most accuracy, approximately 4 meters away from the sensors. The calibration setup was chosen as in figure 5.11.

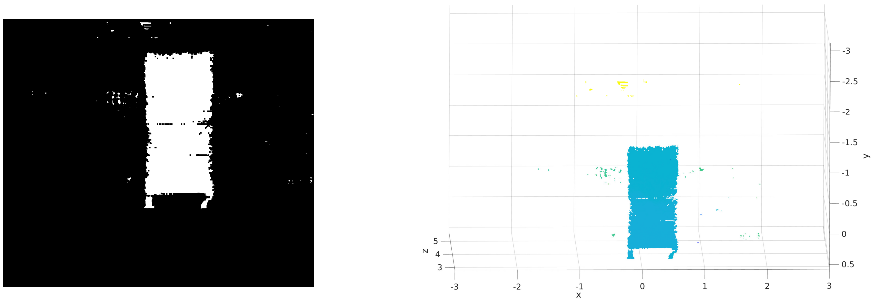


Figure 5.12: Disparity map and reconstructed 3D points

To obtain 3D points from the camera a disparity map has to be calculated. After trail and error it was found that the function

```
disparitySGM(image1, image2, name, value)
```

in MATLAB with a disparity range of $[68, 116]$ and a uniqueness threshold of 0.3 produces the most exact disparity map. The disparity map and the reconstructed 3D points are illustrated in figure 5.12.

As the object of interest is perpendicular to the sensors the field of view of the LiDAR is reduced. The resulting point cloud of the LiDAR is shown i figure 5.13. Next, both point clouds are reduced to only include the object of interest, thus removing noise and points of floor and walls. The two point clouds can be represented in the same coordinate frame by performing a transformation on the LiDAR point cloud so that it is represented in

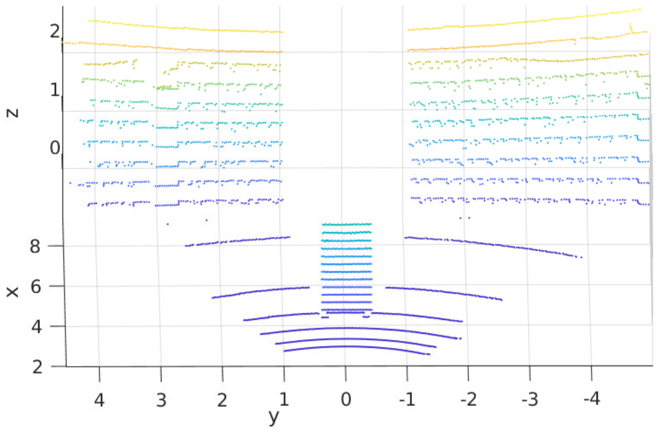


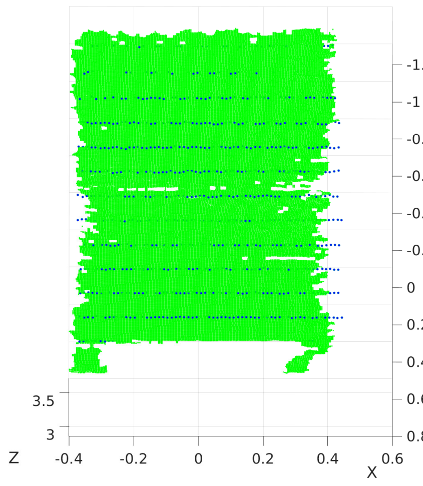
Figure 5.13: LiDAR point cloud

the camera coordinate frame. This requires a change in axis and a rotation of 180 degrees about the z-axis in order to represent positive height downwards, as it is in the camera reference frame. Next, the rigid body transformation between the LiDAR and the stereo camera is calculated using ICP. Since the camera point cloud is denser than the LiDAR point cloud two calibration approaches are performed. In one approach the ICP-algorithm is applied to entire point clouds and in the second it is applied to a few specific points.

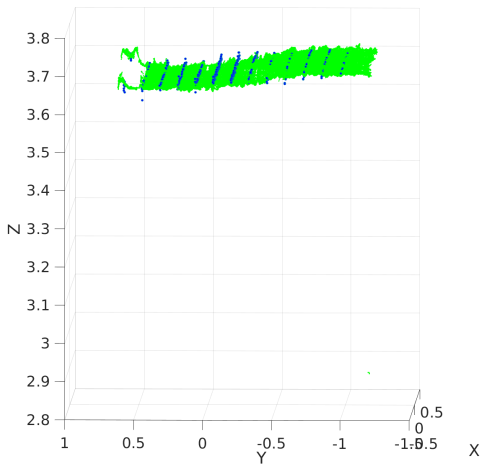
5.3.2 ICP on full point clouds

The ICP-algorithm is performed on the two point clouds in their entirety. Applying the calculated transformation on the camera point cloud and visualizing the two point cloud together yields the result in figure 5.14. Here the point clouds are plotted using the function `Scatter3(X, Y, Z, S)` in MATLAB where S, the circle size, is set to 6 for the LiDAR in order to better illustrate the difference between the point clouds. The LiDAR point cloud is plotted in blue and the camera point cloud in green. After careful inspection of the plot it can be seen that the LiDAR produces more points along the z-axis and thus produces a point cloud with larger depth than the camera that only observes a flat object. In spite of this the ICP matches the points along the z-axis quite nicely. On the other hand the algorithm is not able to match the points of the wheels of the calibration object in the two point clouds. This results in a mismatch along the y-axis.

To avoid mismatched points a method for outlier rejection can be deployed. For example the algorithm Random Sample Consensus (RANSAC) fits a model to experimental data [8]. It finds the smallest amount of data that can be used to describe a model and adds on points that are compatible with the model when possible. The points not added to the



(a) Front view



(b) Side view

Figure 5.14: Camera point cloud after transformation plotted with LiDAR point cloud

model are labeled as outliers. In this case the calibration target used is of such a simple from outliers can simply be cut out of the point cloud. The mismatch that occurs here is due to the wheels on the calibration object. Thus in order to improve the calibration the wheels are cut out of both point clouds before performing the ICP.

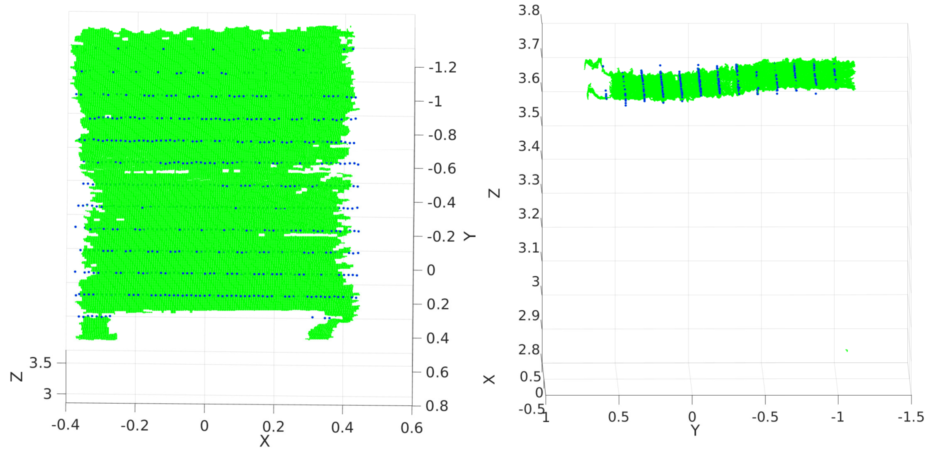
$$\mathbf{T_{cl}} = \begin{bmatrix} 0.9995 & 0.0309 & -0.0018 & 0 \\ -0.0309 & 0.9995 & 0.0004 & 0 \\ -0.0055 & 0.0005 & 1.0000 & 0 \\ -0.1980 & 0.0954 & 0.0570 & 1 \end{bmatrix} \tag{5.5}$$

The obtained transformation is given in equation 5.5. The last column tells us that no translation is needed which makes sense considering the placement of the cameras according to the LiDAR. The cameras are placed on each side of the LiDAR thus the resulting camera frame of the stereo camera will almost coincide with the LiDAR frame. The small rotation is most likely due to the cameras being angled the way they are.

The results is plotted in figure 5.15. Comparing with figure 5.14 some improvement in the alignment of the two point clouds can be observed. There will still be some margin of error due to that the lasers from the LiDAR might not hit exactly at the top of the calibration object.

5.3.3 ICP algorithm on key points in point clouds

The inaccuracy due to the lasers of the LiDAR not hitting the top of the object can be reduced by applying the ICP-algorithm on key points of the point clouds. The key points need to be extracted from the same place of the object in both point clouds. In order to



(a) Front view

(b) Side view

Figure 5.15: Camera point cloud after transformation plotted with LiDAR point cloud

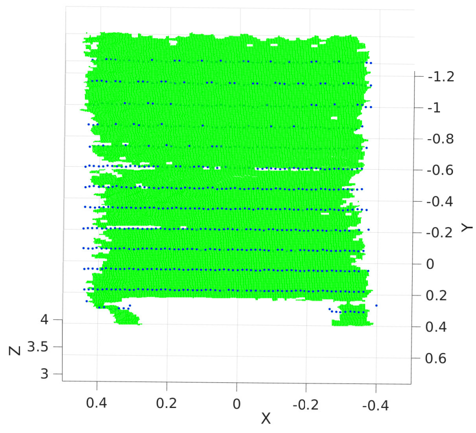
accurately calculate the transformation of each of the axis at least 3 points is necessary. The extracting of the key points is performed by utilizing the function `clickA3DPoint(pointCloud)` from the package `Click3dPoint` by Babak Taati [29].

A challenge here is extracting 3D points from both point clouds generated by the same place on the object. The LiDAR point cloud key points lie along the edges of the object while the camera on the other hand struggles with constructing 3D points along edges. Comparing the objects height and distance from the sensors in the real world and the vertical resolution of the LiDAR, as described in section 4.1.4, one can assume that the LiDAR misses approximately 10 centimeters of the top of the object. This is taken into consideration when extracting the key points.

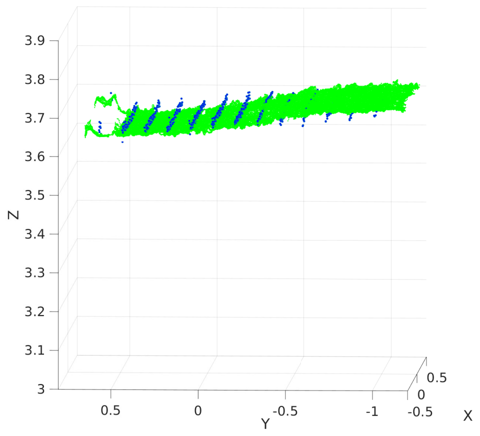
$$\mathbf{T_{cl}} = \begin{bmatrix} 0.9999 & 0.0118 & -0.0062 & 0 \\ -0.0119 & 0.9997 & -0.0197 & 0 \\ 0.0059 & 0.0197 & 0.9998 & 0 \\ -0.1849 & 0.0252 & 0.0502 & 1 \end{bmatrix} \tag{5.6}$$

The obtained transformation is given in equation 5.6. There is no noticeable big difference from the transformation obtained in equation 5.5. Looking at the resulting plots in figure 5.16 some improvement from the calibration done by using the full point cloud can be seen. The wheels are more accurately aligned and it is visible that the camera point cloud sees more of the top part of the object than the LiDAR does. We can tolerate some uncertainty here seeing as it is the depth that is in focus in this project. The plotted side view shows that the points in the z-axis is not as nicely aligned as they are in both figure 5.14 and 5.15.

After evaluating the different transformations the one obtained by performing ICP on the full point clouds and cutting out the wheel seems to be the most exact one. To obtain



(a) Front view



(b) Side view

Figure 5.16: Camera point cloud after transformation plotted with LiDAR point cloud

a good ground truth for the depth of the stereo cameras depth is the most important factor when aligning the two point clouds. Thus the transformation given by equation 5.15 is utilized further in this project.

Chapter 6

Experiment and result

6.1 Experimental set up

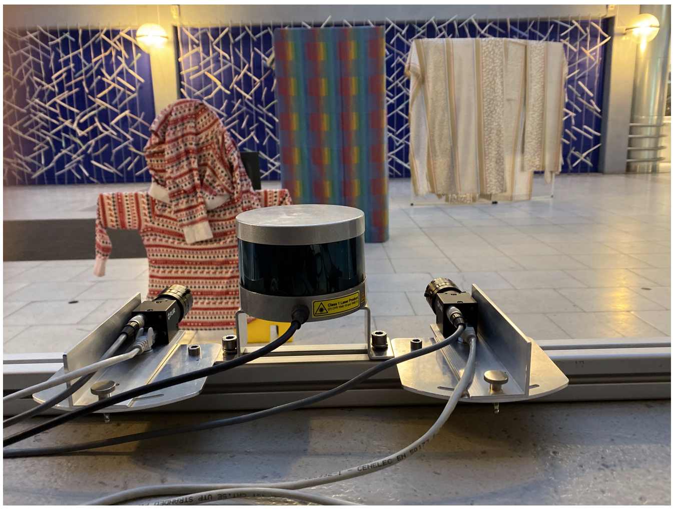


Figure 6.1: Experimental set up

To evaluate the depth obtained by the stereo camera an experiment is performed. A scene with three objects placed in different positions and depths according to the camera and LiDAR was created. The resulting scene is given in figure 6.1. The objects are wrapped in fabric with variable patterns making it easier for the camera to match pixels in the two images. The scene is selected due to its noisy background. Having a background with structure and pattern helps to avoid wrong pixel matches between the object and the background. The objects are placed at roughly 2, 4 and 6 meters away from the sensors.

Figure 6.2 shows the resulting point cloud of the scene obtained by the LiDAR given

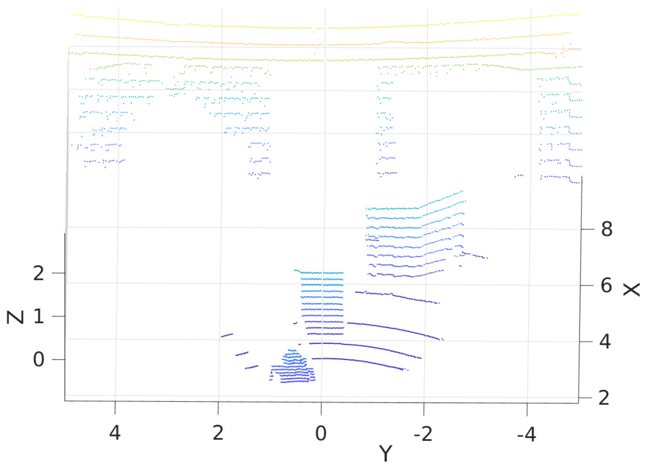
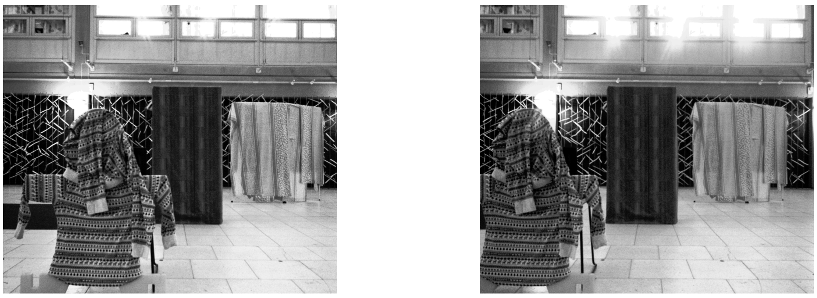


Figure 6.2: Lidar point cloud of experimental scene given in the camera frame

in the camera frame. The objects are of such a size that they are all visible for the LiDAR at the different distances. The distances measured by the LiDAR coincides with the measurements taken of the scene.



(a) Rectified image from the left camera **(b)** Rectified image from the right camera

Figure 6.3: Rectified stereo images of the scene

Figure 6.3 shows the images captured by the stereo camera after they have been rectified. The right camera is not able to see the left arm of the first object. Thus the stereo camera will not be able to perceive the depth of the left arm and it will not appear in the reconstructed point cloud for either one of the matching approaches.

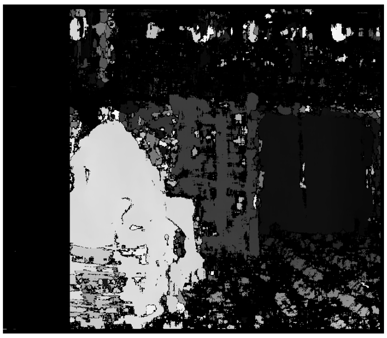
6.1.1 3D points from stereo camera

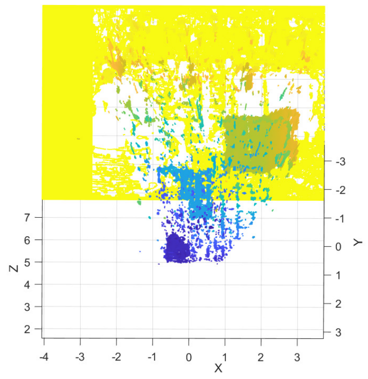
A disparity map using block matching and SAD as cost function is created. To obtain a good result a parameter tuning must be done.

Disparity range	$[43, 203]$
Blocksize	31
Contrast Threshold	0.3
Uniqueness Threshold	5
Distance Threshold	disabled
Texture threshold	0.0002

Table 6.1: Final parameter values for creating disparity map using block matching

Table 6.1 shows the obtained values after tuning the parameters. The minimum and maximum disparity are set to the distance in pixels between the stereo images of the last and first object. Thus the wall should not appear in the disparity map. The rest of the parameters were found by trial and error. The resulting block size is somewhat big and thus makes the algorithm slower. A smaller block size resulted in more mismatched pixels. Seeing as the objective is to see how accurate the estimated depth of the stereo camera is the running time of the algorithm is not prioritised.


(a) Disparity map

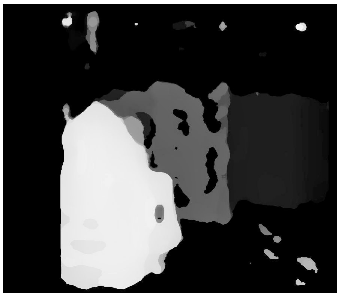


(b) Point cloud

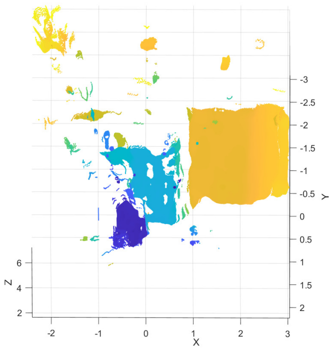
Figure 6.4: Disparity map from block matching and the resulting point cloud

The resulting disparity map and the reconstructed point cloud can be seen in figure 6.4. From the point cloud one can see that the algorithm creates a yellow artificial wall. When pixels are not matched their disparity value is set to the minimum disparity. This artificial wall can be removed by simply cutting it out of the point cloud. Both the disparity map and the point cloud are somewhat noisy. The objects can be seen clearly but they are surrounded by noise. For visualization purposes a median filter is deployed.

The resulting disparity and point cloud after applying a median filter and removing the artificial wall are given in figure 6.5. A median filter of grid $[50, 50]$ is deployed. For the



(a) Disparity map



(b) Point cloud

Figure 6.5: Disparity map from block matching and the resulting point cloud

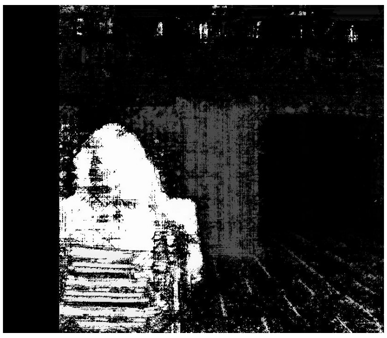
purpose of using the stereo camera on an autonomous ferry this is not feasible due to the slow running time of filters of that size. It will also become problematic if one wants to estimate the distance of small objects. From the point cloud in figure 6.4 one can see that a filter is not necessary in order to make out the objects. The filter is deployed here solely for visualization purposes.

Disparity range	$[53, 181]$
Uniquness Threshold	20

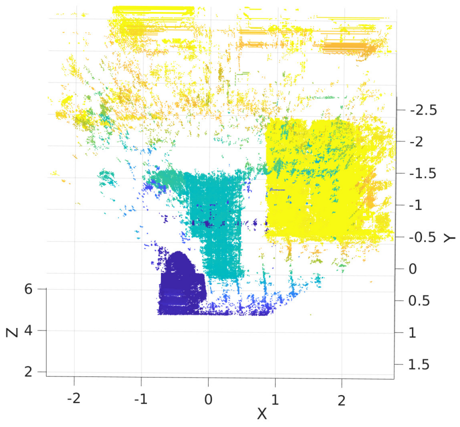
Table 6.2: Final parameters used for creating disparity map using Semi-Global Matching

A disparity map using Semi-Global Matching is also created. The resulting parameters used are presented in table 6.2. The disparity range differs from the one used for block matching. This is due to the restriction on the disparity range by MATLAB's function `disparitySGM()`. The difference between min and max disparity must be less or equal to 128. This calls for finer tuning of the disparity range due to the objects in the scene being placed at such different distances.

The resulting disparity map and the reconstructed point cloud is given in figure 6.6. For visualization purposes the noise above the objects can be cut out of the point cloud.



(a) Disparity map



(b) Point cloud

Figure 6.6: Disparity map from Semi-Global Matching and the resulting point cloud

6.2 Results and analysis

A comparison of the depth estimate obtained by the matching algorithms with the LiDAR is made by following the procedure presented in section 3.3. First, for a visual comparison, the point cloud obtained by the stereo camera is plotted together with the point cloud obtained by the LiDAR. Next an evaluation of the point clouds is done in order to decide if a nearest neighborhood approach to finding corresponding points is reasonable. Then the distance between corresponding points is calculated.

In this section the object placed closest to the sensors is referred to as the first object (object 1), the object in the middle as the second object (object 2) and the object furthest away as the third object (object 3). Figure 6.7 illustrates which points are selected from the LiDAR. The points are selected from different places on the surface of each object. Using points of different heights and widths makes it possible to ensure that the entire object surface is estimated to the same depth. The point clouds produced by the stereo camera are somewhat noisy. The depth of the edges of the objects is not accurately estimated and thus creates noise in the scene. For this reason points along the edges are avoided. When using objects that do not have a flat surface it is important to select points of the different depths of the object. Here the third point on object 3 is selected for that reason. If the corresponding points are found using the nearest neighborhood approach points are selected from the LiDAR because it is more sparse than the stereo point cloud. Picking points from the stereo point cloud and finding the nearest point given by the LiDAR may result in distances being calculated between two points that are not of the same height.

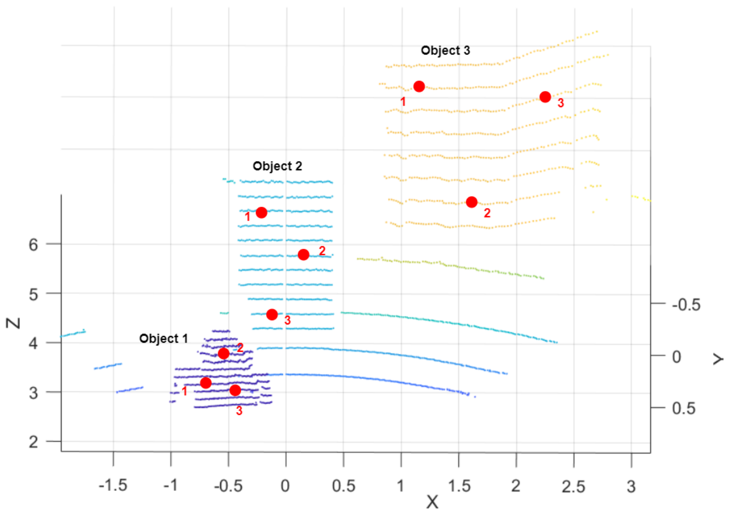


Figure 6.7: Points selected from LiDAR point cloud

6.2.1 Depth estimation by block matching

In figure 6.8 the LiDAR point cloud and the point cloud reconstructed by the disparity map obtained by using block matching are plotted together. The LiDAR point cloud is plotted in purple and the stereo point cloud in green. The first plot shows the two point clouds plotted together from the front, and it seems as though they align quite nicely. But for further inspection the two point clouds are visualized from above in the next figure. Here it can be observed that on the first object the depth is more or less the same. On the second object some deviation can be observed. The stereo point cloud is plotted slightly behind the LiDAR point cloud. On the third and last object a more clear deviation is present. The depth estimated by the stereo camera is greater than the one estimated by the LiDAR.

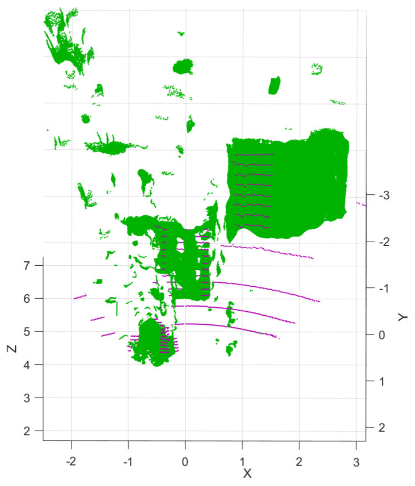
Object 1	
Point	**Distance**
1	0.0090
2	0.0078
3	0.0067

Object 2	
Point	**Distance**
1	0.0275
2	0.0428
3	0.0415

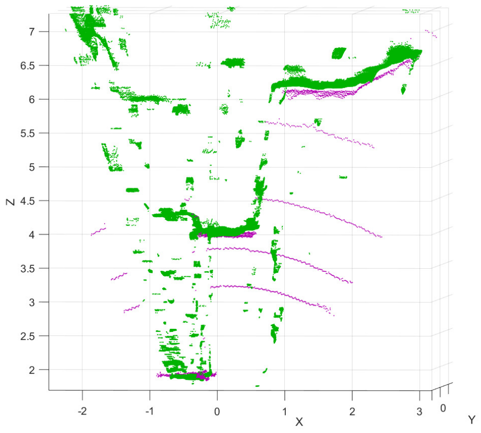
Object 3	
Point	**Distance**
1	0.1531
2	0.1404
3	0.1298

Table 6.3: Distance in meters between selected points from LiDAR and points obtained from block matching

Even though there is some distance between the point clouds finding corresponding points using a nearest neighborhood search is still reasonable. The stereo point clouds is flat and noise is not present right behind the objects. The distance between the perceived depth of the LiDAR and the stereo camera is calculated and given in table 6.3. As observed in the comparison of the two point clouds the deviation increases for each object. The best



(a) Front view

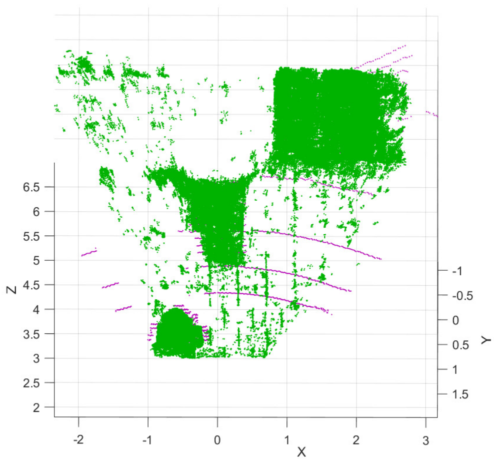


(b) View from above

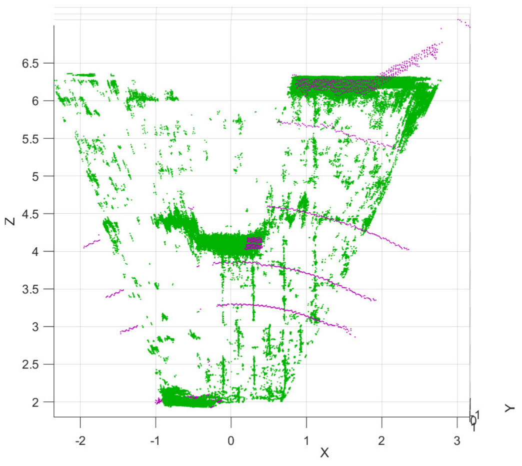
Figure 6.8: Camera point cloud obtained from block matching plotted together with the LiDAR point cloud

result is gained from the first object. The difference in the distance lies between 0.7 cm and 0.9 cm. For the second object the difference is around 4 cm and for the third object the difference is around 14 cm.

6.2.2 Depth estimation by Semi-Global Matching



(a) Front view



(b) View from above

Figure 6.9: Camera point cloud obtained by Semi-Global Matching plotted together with the LiDAR point cloud

In figure 6.9 the LiDAR point cloud and the point cloud reconstructed by the disparity map obtained by Semi-Global Matching are plotted together. The LiDAR point cloud is plotted in purple and the stereo point cloud in green. The points making out the objects of the two point clouds seem to align quite well. A deviation can be observed on the third object. The stereo point cloud does not align with the part of the object that is bent.

Object 1	
Point	**Distance**
1	0.0069
2	0.0192
3	0.0135

Object 2	
Point	**Distance**
1	0.0060
2	0.0043
3	0.0092

Object 3	
Point	**Distance**
1	0.0284
2	0.0065
3	0.1396

Table 6.4: Distance in meters between selected points from LiDAR and points obtained from Semi-Global Matching

The distance between the perceived depth of the LiDAR and the stereo camera is calculated and given in table 6.4. The distances are calculated from the same points in the LiDAR point cloud as they are for block matching. A nearest neighborhood search is used for finding corresponding points. For the first object the difference in the distance lies between 0.7 cm and 1.9 cm. The best result is gained from the second object where the difference lies between 0.4 cm and 0.9 cm. For the third object the difference lies between 0.7 cm and 14 cm which is quite a large gap.

6.3 Discussion

At first glance it seems as though the point cloud obtained from Semi-Global Matching yields a better result than the one obtained by block matching. In the block matching approach the deviation from the LiDAR becomes greater for each object. The difference in distance in the first two objects is in an acceptable range but an error of 15 cm might be somewhat large. Overall the distances calculated for Semi-Global Matching are smaller. The second object is observed with the most accurate depth. This is as expected due to the object being placed in the distance where the stereo setup has its focus. For the third object a larger deviation is observed as the distance of the third point is calculated as high as 14 cm. This is due to the misalignment with the bent part of the object. This misalignment most likely occurs due to the restriction in the difference of the disparity range. When using block matching the minimum disparity is set lower which yields a better result for the third object. Implementing a Semi-Global Matching with OpenCV in Python or C++ instead of MATLAB one does not have this restriction. OpenCV's implementation is a modified version of [17].

The acceptable margin of error depends on the application. If the stereo camera is to be used for object detection on the autonomous ferry Milliampere for collision avoidance purposes then an error of 15 cm could be accepted. A slow moving boat will need to adjust its course early and pass the object with a much greater distance than 15 cm. The problem for the estimation done by block matching is the increasing error of objects of greater distance. The actual depth increases by approximately 2 meters between each object. The error in the estimation increases from first being around 0.8 cm, to 4 cm and in the end 14 cm. From the depth estimates obtained here no clear correlation can be found. If a correlation can be found when testing on more objects of different depths then the error can be modeled and accounted for.

The depth estimation of the stereo camera is highly dependent on the tuning of the parameters of the matching algorithms. Semi-Global Matching requires fewer parameters to be tuned which makes it easier to use. The block matching approach might yield a better result with more tuning. Here one could also try to implement the algorithm using OpenCV in Python or C++. When using Python or C++ more code is available online for pre-processing of the images before calculating disparity. This could result in less noise in the disparity map and the reconstructed point cloud. It is important to take into consideration how this affects the running time.

The depth estimation is also dependent on the calibration performed on the stereo camera. The calibration performed here yield a reasonable result considering the parameters. But performing a calibration with a checkerboard covering less than 20 percent of the field of view is a long and tedious process. An ideal checkerboard should have a width of approximately 5 meters which is over 4 meters wider than the one used. Since the operating distance is at 4 meters the resulting calibration included over 250 images. An attempt on calibrating at approximately 10 meters when the baseline was set to 1.75 meters was done. Capturing images that cover the whole field of view proved difficult and a high number of images was required. Obtaining a good calibration proved challenging.

The camera setup is a factor affecting the obtained depth estimate. A more accurate result could be obtained with a larger baseline and angling the cameras more towards each other. In theory a higher accuracy is achieved from a larger baseline and a more angling of the cameras. A small baseline could cause several points to project to the same pixel. Increasing the baseline could, on the other hand, make the search problem more difficult. By analyzing the point clouds and the disparity maps one can conclude that both algorithms are able to solve the problem of finding corresponding pixels in the objects. Thus switching to a larger baseline could provide a reasonable result.

The result of the depth comparison is dependent on the calibration between the LiDAR and the stereo camera. Examining the transformation obtained and the visual illustration of the alignment of the point clouds the calibration yields a reasonable result. The small misalignment in height does not affect the analysis of the depth estimates. When selecting points for comparison the edges of the object are avoided due to the noise in the stereo point cloud. Thus the alignment of the point clouds proves reasonable.

When observing the transformed stereo point cloud together with the LiDAR a good basis of evaluating the difference in depth is achieved. Calculating the distance between corresponding points in the point clouds generates a more accurate comparison. From selected points in the LiDAR point cloud the corresponding points in the stereo point cloud are found using a nearest neighborhood search. The point clouds produced by the stereo camera is noisy but the noise mostly occurs above, below and next to the object, while the objects detected are flat. From this it follows that when finding the nearest neighbour of the selected points they should correspond to the same point on the object surface in the stereo point cloud. The distances calculated also correspond with what one can observe from the plots. Points along the edges of the stereo point cloud may cause large errors when calculating the distance due to the noise present in the stereo point cloud here. Thus the comparison is dependent on the points selected. When using the method it is important to look at several points on the objects and that the points a selected in the LiDAR point cloud and not the stereo point cloud due to the difference in density.

Chapter 7

Conclusion and future work

7.1 Conclusion

A procedure for comparing the depth estimate of a stereo camera and a LiDAR has been implemented. The results and the obtained parameters show that the extrinsic calibration of the stereo camera and the LiDAR proves to give a reasonable result. The point clouds obtained from the stereo camera align well with the point cloud from the LiDAR. The visualization of the point clouds and the calculation of the distance between corresponding points provide a good measure for evaluating the accuracy of the depth estimation from the stereo camera. How accurate the corresponding points found are is dependent on how noisy the stereo point cloud is and which points are selected from the LiDAR. An evaluation of weather the nearest neighborhood search approach to finding corresponding points will give a reasonable result must be made when visualizing the point clouds together. When using the procedure some considerations should be taken when selecting points

- The area the points are selected from needs to be visible in both point clouds.
- Points in different heights and widths should be selected.
- If the object surface is not flat points should be selected in the different depths of the object.
- Try to avoid choosing points along the edges of the objects.

A stereo setup has been implemented and designed. The results show that the Semi-Global Matching method performed well and only small deviations from the LiDAR are present. Based on this one can conclude that the stereo camera has the potential to be used for object detection purposes on long distances. Some of the noisy parts of the point cloud was cut out for visualization purposes. If taken in use a clustering procedure must be implemented in order to distinguish objects of interest from the noise.

7.2 Future work

To get a proper evaluation of the use of depth estimation by the stereo camera on the autonomous ferry Milliampere tests at greater distances must be performed. The stereo setup designed to be optimal at 60 meters can be utilized. A new calibration of the sensors must be performed. At greater distances the LiDAR point cloud will become sparser which can affect the calibration accuracy. Cutting out areas of the point clouds causing mismatches may prove more difficult. An outlier rejection method such as RANSAC [8] can be implemented to ensure the quality of the calibration.

In order to use the Semi-Global Matching algorithm on a scene including objects with large variations of depth the implementation should be done by using OpenCV in Python or `C++`. The implementation in this project has the restriction of only allowing a difference of 128 in maximum and minimum disparity. This is not the case in the implementation in Python and `C++`. Further Processing of the disparity map should also be performed in order to remove noise and avoid mismatched point correspondences.

To ensure the quality of the depth estimate tests in different scenes and under different illuminations should be performed. If the stereo camera is taken in use on Milliampere the performance must be ensured in a varying environment. The environment around an autonomous ferry is affected by factors like weather conditions and changes in light. When testing under such circumstances it might be simpler to perform the comparison on a variable scene as opposed to a static one. For this to work a time synchronization of the sensors should be performed. This could be done by using timestamps available by ROS to label the messages sent by both the sensors.

When using the depth estimations in real-time systems an evaluation of the running times of the algorithms should be performed. In the complementary specialization project *Stereo vision using local methods for autonomous ferry* by Trine Ødegård Olsen an evaluation of local matching methods has been conducted. The accuracy and running time of the Semi-Global Matching method should be compared with the local methods presented here.

Bibliography

[1] *About ROS*. https://www.ros.org/about-ros/. Open Source Robotics Foundation.

[2] *Spinnaker SDK driver - ROS wiki*. http://wiki.ros.org/spinnaker_sdk_camera_driver. Open Source Robotics Foundation.

[3] *Velodyne driver - ROS wiki*. http://wiki.ros.org/velodyne?distro=kinetic. Open Source Robotics Foundation.

[4] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992.

[5] D. P. Curtin. Understanding the baseline. http://www.shortcourses.com/stereo/stereo3-14.html, 2011.

[6] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna. LiDAR-Camera Calibration using 3D-3D Point correspondences. *ArXiv e-prints*, May 2017.

[7] Edmund Optics. TECHSPEC Compact Fixed Focal Length Lens #58-000. https://www.edmundoptics.com/p/85mm-c-series-fixed-focal-length-lens/14947, 2019.

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In M. A. Fischler and O. Firschein, editors, *Readings in Computer Vision*, pages 726 – 740. Morgan Kaufmann, San Francisco (CA), 1987.

[9] FLIR. *"Configuring Synchronized Capture with Multiple Cameras"*, 12 2017. https://www.flir.com/globalassets/support/iis/application-notes/tan2016008-configuring-synchronized-capture.pdf.

[10] FLIR. FLIR BLACKFLY S P/N BFS-PGE-50S5. https://flir.app.boxcn.net/s/mj27am7zik371ivyzv352gmt390yqt6z/file/535139104941, 2019.

[11] S. Ghosh. ”Generating Dense Disparity Maps using ORB Descriptors”. `https://sourishghosh.com/2016/dense-disparity-maps-orb-descriptors/`, 2016.

[12] M. Greenspan and M. Yurick. Approximate k-d tree search for efficient icp. In *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pages 442–448, Oct 2003.

[13] C. Guindel, J. Beltrán, D. Martín, and F. García. Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Oct 2017.

[14] T. V. Haavardsholm. A handbook in Visual SLAM. 2019. Compendium in the course TTK21 Introduction to Visual Simultaneous Localization and Mapping - VSLAM at NTNU.

[15] R. Hamzah and H. Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016:1–23, 01 2016.

[16] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision, 2nd edition*. Cambridge University Press, 2003.

[17] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, volume 2, pages 807–814 vol. 2, June 2005.

[18] A. Kaehler and G. Bradski. *Learning OpenCV 3*. O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2016.

[19] D. Kumari and K. Kaur. A survey on stereo matching techniques for 3d vision in image processing. *International Journal of Engineering and Manufacturing*, 6:40–49, 07 2016.

[20] Y. Liu and J. Aggarwal. *Local and Global Stereo Methods*, pages 297–308. Elsevier Inc., United States, 12 2005.

[21] MathWorks. disparityBM. `https://www.mathworks.com/help/vision/ref/disparitybm.html`.

[22] MathWorks. disparitySGM. `https://www.mathworks.com/help/vision/ref/disparitysgm.html`.

[23] MathWorks. What is camera calibration? `https://de.mathworks.com/help/vision/ug/camera-calibration.html`.

[24] MathWorks. Stereo Camera Calibrator App. `https://se.mathworks.com/help/vision/ug/stereo-camera-calibrator-app.html`, October 2016.

[25] MathWorks. Single Camera Calibrator App. https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html, 2019.

[26] K. McCabe. "How to Set Up a Stereo Machine Vision Solution". https://www.qualitymag.com/articles/93543-how-to-set-up-a-stereo-machine-vision-solution, October 2016.

[27] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

[28] Rostam Affendi Hamzah, Rosman Abd Rahim, and Zarina Mohd Noh. Sum of absolute differences algorithm in stereo correspondence problem for stereo matching in computer vision application. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 1, pages 652–657, July 2010.

[29] B. Taati. Click3dPoint. https://se.mathworks.com/matlabcentral/fileexchange/7594-click3dpoint.

[30] B. Templeton. Elon Musk's War On LIDAR: Who Is Right And Why Do They Think That? *Forbes*. Avaliable at: https://www.forbes.com/sites/bradtempleton/2019/05/06/elon-musks-war-on-lidar-who-is-right-and-why-do-they-think-that/#61d36c802a3b.

[31] L. UK. "How does LiDAR work?". http://www.lidar-uk.com/how-lidar-works/, 2019.

[32] Velodyne. Velodyne LiDAR Puck, 2019. 63-9229 Rev-J.

[33] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019.

[34] L. A. Wasser. "The Basics of LiDAR - Light Detection and Ranging - Remote Sensing". https://www.neonscience.org/lidar-basics, 2019.

[35] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.