Master's thesis

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

Lina Charlotte Kristoffersen Theimann
Trine Ødegård Olsen

Stereo vision for autonomous ferry

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke,
Co-supervisor: Annette Stahl, Øystein K. Helgesen.
June 2020

NTNU
Norwegian University of
Science and Technology

Lina Charlotte Kristoffersen Theimann
Trine Ødegård Olsen

Stereo vision for autonomous ferry

Master's thesis in Cybernetics and Robotics
Supervisor: Edmund Førland Brekke,
Co-supervisor: Annette Stahl, Øystein K. Helgesen.
June 2020

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

NTNU
Norwegian University of
Science and Technology

Abstract

The thesis discusses far-range object detection for stereo, calibration, and system implementation for unmanned surface vehicles. The stereo system record with a baseline of 1.80 meters, with a fixation point at 50 meters. For far range distance estimation, a procedure for extrinsic stereo calibration is introduced. Testing the procedure at different distances, show that the selected scene is of higher importance than calibrating at the operating range. The calibration at 20 meters achieves the best overall distance estimates.

The stereo system is designed for the autonomous ferry milliAmpere, and tested in a maritime environment. The system processes raw sensor data and output world coordinates of the detected objects. The disparity map created using Sum of Absolute Difference (SAD) and a Fast Global Image Smoothing based on Weighted Least Square (WLS) filter, is robust and has low computational cost. For object detection purposes, two clustering techniques are implemented. A convolution neural network is applied for classification, and used in combination with the disparity map to extract 3D positions of objects. The method is robust against noise in the disparity map, but appear to be partially inconsistent in the estimates. An alternative detection method based on hierarchical clustering using Euclidean distance yields more reliable detections, but is more prone to noise. The implemented system shows potential for vessel detection in a range of 10 to 200 meters, but it is still not clear that the detection performance is good enough to rely on in an autonomous collision avoidance system.

Sammendrag

Denne oppgaven diskuterer stereosyn for å detetktere objekter på lange distanser, kalibrering og system implementasjon for et førerløst fartøy. Oppsettet innehar en interaksiell avstand på 1.80 meter mellom kameraene for å optimalt detektere objekter på 50 meters avstand. En metode for å kalibrere på lengre avstander er foreslått. Metoden er testet og resultatene viser at valg av kalibrerings scene er viktigere enn avstanden til kalibreringsobjektet. Kalibreringen utført på 20 meter viste å gi mest nøyaktige dybdeestimat.

Systemet er designet for fergen milliAmpere og er testet i et marint miljø. Systemet prosesserer rå sensordata og gir ut verdenskoordinater til detekterte objekter. Dybdekartet er implementert ved bruk av algoritmene Sum of Absolute Differences og Fast Global Image Smoothing filter basert på minste kvadraters metode viser seg å være robust. To metoder er implementert for å detektere objekter i dybdekartet. Ett konvolusjonelt nevralt nettverk (CNN) for klassifisering gir i kombinasjon med dybdekartet verdensposisjonen til objekter av interesse. Metoden viser seg å være robust mot støy, men har noe inkonsekvente estimater. En alernativ metode deteksjonsmetode basert på hierarkisk grupperingsalgoritme som bruker Euklidsk avstand gir mer pålitelige deteksjoner, men er mer utsatt for støy i dybdekartet. Det implementerte systemet viser potensiale for å detektere objekter i avstander mellom 10 og 200 meter, men ytterligere testing må utføres for å kunne integrere sensoren i ett helhetlig system for navigasjon av et autonomt fartøy.

Preface

This is the concluding part of a 5-year Masters's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). We want to thank our supervisor Edmund Førland Brekke for valuable feedback, guidance, and support during this thesis. Also, we would like to thank our two co-supervisors Annette Stahl and Øystein Kaarstad Helgesen, for their input and advice. The choice of the thesis was highly motivated by working with a Unmanned Surface Vehicle (USV) during a summer-internship[1]. The goal was to rebuild a watercraft into a USV, allowing it to be remotely controlled and to execute missions on its own. Even though both the watercraft and the ferry milliAmpere still have some miles to swim before being fully autonomous, we would like to join the journey.

The original plan for this thesis was to cooperate with the Autoferry project, directly implementing the stereo system on the autonomous ferry milliAmpere. Due to circumstances around the COVID-19 pandemic, there was little to no time testing the implementation properly. The only time for testing was the 21st of May, with help from the Department. A huge thank you to Egil Eide, Tobias Rye Torben, and Daniel Andrè Svendsen for helping us test on a public holiday.

Johann Alexander Dirdal and Simen Viken Grini also deserve to be mentioned for lending us the LiDAR and providing weights for YOLO, respectively. For the calibration, we would be nowhere without the technical staff at ITK. Thanks to Glenn Angel and Terje Haugen for magically creating two checkerboards of size 1.5 times 3 meters. Furthermore, Stefano Brevik Bertelli deserves to mentioned for being helpful in times of need. A thank you to the janitor for letting us borrow equipment and helping us with access to NTNU during COVID-19. As well as Erik Wilthil and Bjørn-Olav Holtung Eriksen for explaining the system on the ferry milliAmpere, networking-help, and letting us borrow a handheld GPS.

[1] `https://coastalshark.no/`

Table of Contents

Abbreviations

CCD	=	Charged-coupled device
CNN	=	Convolutional Neural Networks
CPU	=	Central Processing Unit
CT	=	Census Transform
DLT	=	Direct Linear Transformation
FOV	=	Field of view
FP	=	False Positive
GPIO	=	General-purpose input/output
IoU	=	Intersect over Union
LiDAR	=	Light Detection and Ranging
MI	=	Mutual Information
MLE	=	maximum likelihood estimation
MSAC	=	M-estimater SAmple Consensus
NDT	=	Normal-distributions transform
NED	=	North-East-Down
NNS	=	Nearest Neighbor Search
OpenCV	=	Open Source Computer Vision Library
PCL	=	Point Cloud Library
PoE	=	Power over Ethernet
PRC	=	Precision Recall Curve
ptCloud	=	point cloud
RANSAC	=	Random Sample Consensus
RMSE	=	Root Mean Square Error
ROS	=	Robot Operating System
RTK	=	Real Time Kinetic
SAD	=	Sum of Absolute Differences
SDK	=	Software Development Kit
SVD	=	Singular Value Decomposition
TP	=	True Positive
USV	=	Unmanned Surface Vehicle
WLS	=	Weighted Least Squares
YOLO	=	You Only Look Once

Nomenclature

$\mathbf{E}$	Essential matrix
$\mathbf{F}$	Fundamental matrix
$\mathbf{H}$	Homography matrix
$\mathbf{K}$	Intrinsic matrix
$\mathbf{T}$	Transformation matrix
$\mathbf{R}$	Rotation matrix
$\mathbf{t}$	translation vector
$\mathbf{P}$	Camera matrix
b_x	Baseline
$\mathbf{x}$	Pixel position
$\mathbf{x}_{\mathrm{L}}$	Pixel position, left image
$\mathbf{x}_{\mathrm{R}}$	Pixel position, right image
u_{L}	Pixel in u-direction, left image
u_{R}	Pixel in u-direction, right image
v_{L}	Pixel in v-direction, left image
v_{R}	Pixel in v-direction, right image
$\mathbf{x}_{\mathrm{W}}$	World point
X	World position in x-direction
Y	World position in y-direction
Z	World position in z-direction
$\mathbf{x}'$	World point projected to the image plane
$\mathbf{x}'_{\mathrm{R}}$	True pixel position of projected world point, right image
${u_j}'$	True reprojected pixel position in u-direction
${v_j}'$	True reprojected pixel position in v-direction
$\mathbf{x}_{\mathrm{C}}$	Point in camera frame
f	Focal length in millimeters
α_u	Focal length in u-direction in pixel units
α_v	Focal length in v-direction in pixel units
u_0	Principal point in u-direction in pixel units
v_0	Principal point in v-direction in pixel units
$I(u, v)$	Image pixel intensity at location u,v
I_L	Intensity of left image
I_R	Intensity of right image

d	Disparity measured in pixels
$\mathbf{d}$	Displacement vector
Δu	change in disparity u-dir
Δv	change in disparity v-dir
$I(u, v)$	Image pixel intensity at location u,v
W_m	Matching window
erf	Error function
ω	Weighting function
$\mathbf{P}_n$	Precision in PRC
$\mathbf{R}_n$	Recall in PRC
$\boldsymbol{\Sigma}$	Covariance
$\boldsymbol{\mu}$	Mean
p	Pose

Chapter 1

Introduction

Situational awareness for an unmanned vessel requires detection and classification of the surroundings. By mounting sensors on a moving vehicle, the ownship has adequate data to sense, process, and understand its surroundings. For collision avoidance, the sensors must perceive obstacles and require their world positioning to track and predict motion. Today, most unmanned surface vehicles (USVs) make use of IMUs, Radar, LiDARs, and cameras to operate in a dynamic 3-dimensional environment. Radar systems are widely used and efficient for detecting moving objects at very far range, i.e., above 500 meters. However, weaknesses such as relatively long processing time and lack of precision makes it unsuitable for object detection at a closer range (Shin et al., 2018). Stereo cameras and LiDARs are, in principle, able to cover the shorter range and navigate in port areas.

The primary goal of this project is to explore the possibility of using stereo vision for visual vessel detection. While a LiDAR can acquire reliable depth information with object accuracy of centimeters, it is expensive and has no opportunity to classify the specific type of object (Templeton, 2013). A stereo camera has the ability to simultaneously retrieve both the type of object present and its 3D localization in the world. Stereo camera inherent limitations to sensing from stereo matching algorithms and small baselines. The sparse 3D LiDAR and dense stereo camera have complementary characteristics which several researchers use to estimate high-precision depth based on a fusion of the sensors (Park et al., 2018). Leveraging complementary properties by sensor fusion is an extensive research area, but is it possible to achieve centimeter precision with a stereo camera? How can stereo-vision improve visual vessel detection?

The overall motivation behind this work is to design and construct a stereo system suitable to integrate on the autonomous ferry milliAmpere. The project is affiliated with the Autoferry project[1], which aims to develop new concepts and methods which will enable the development of autonomous passenger ferries for urban water transport. milliAmpere is equipped with several sensors for situational awareness, where a LiDAR and a radar are utilized for 3D object detection. As the ferry will operate in a confined environment

[1] https://www.ntnu.edu/autoferry

Figure 1.1: Autonomous ferry milliAmpere

a stereo system may be a suitable choice for redundancy. This describes the development and implementation of a stereo system for proof of concept.

1.1 Background

Passive optical sensors are expected to become indispensable for advanced driver assistance systems (Templeton, 2013). A major reason why several developers often choose LiDARs is the high level of accuracy compared to stereo camera. However, Wang et al. (2019) at Cornell University presented in June 2019 a method for improving the accuracy of visualizing surroundings by stereo cameras. This is only one among others who recently have diminished the gap in accuracy between stereo cameras and LiDARs. With increasing pixel resolution, improved methods, and applications running on GPU, low-price cameras are put more on pair with LiDAR solutions.

Precise stereo depth estimates depend on an accurate calibration. The internal and external calibration parameters together establish the relationship between 2D image content and 3D object data (Schiller et al., 2012). Considering wide baseline and far-range detection, Warren et al. (2013) state that the calibrated rigid body transformation between two cameras impact 3D scene triangulation the most, and affect long-term autonomy object detection. Error in external calibration and external factors directly affecting the setup, such as temperature expansion of steel, will affect the position accuracy the most. As the external camera parameters directly influence the reconstructed 3D points, these will be the main focus when calibrating. This is also supported by both Stretcha et al., (2008) and Marita et al., (2006).

However, the literature diverge in how to best calibrate the rigid body transformation between two cameras. Stretcha et al. (2008) showed that when using a stereo camera, the variance from the ground truth decrease with decreasing distance to the calibration scene. However, Marita et al. (2006) state that calibration minimizing the projection error on near distances is unsuited for far range stereo-vision. They further mention that the most reliable solution is using a calibration scene on distances comparable with the working range of the stereo reconstruction application. Regardless of the chosen calibration distance, Abdullah et al., (2019) confirms this hypothesis by stating that independent of the distance the calibration is performed, the further the objects from the camera, the higher percentage of errors will be recorded.

Stereo depth estimation require the generation of accurate disparity maps in real-time (Wang et al., 2019). Nevertheless, a disparity map includes noise, and post-processing with a clustering technique is therefore preferred to extract the resulting world position of 3D objects. This can be achieved using both classical clustering techniques or convolutional neural networks for object detection and classification. Clustering points into groups to segment the point cloud is a standard step in processing a captured scene (Nguyen and Le, 2013). Graph-based methods such as hierarchical clustering are robust in dealing with complex scenes (Pendleton et al., 2017). Such methods extracts a dynamical number of clusters and are not dependent on seed points. However, recent advances in computing power has made deep learning based methods the standard approach for object detection (Li et al., 2019). Deep learning with convolutional neural networks tends to outperform classical detection methods by recognizing shapes, edges and colors. YOLO is one of the pioneer real-time CNN models, which has demonstrated detection performance improvements over R-CNN (Zhao et al., 2019). YOLOv3 outperforms the state of the art detection algorithm Faster R-CNN in both sensitivity and computation time (Benjdira et al., 2019). The stereo system will be tested using both YOLOv3 and classical clustering techniques.

1.2 Problem description

The situational awareness of USVs depends on sensors that can perceive the surrounding environment. The purpose of this project is to implement an object detection system using a stereo camera for the autonomous ferry milliAmpere. The project builds on the 5th year specialization projects of the two authors (Theimann, 2020)(Ødegård Olsen, 2020) where a stereo system was set up and tested for shorter distances. While Olsen looked into *Stereo vision using local methods for autonomous ferry*, Theimann complemented by focusing on LiDARs with the project *Comparison of depth information from stereo camera and LiDAR*. The projects calibrated a stereo setup, and tested the system inside with a controlled static scene. This thesis incorporate the two projects and continues the research for a far range stereo system. This thesis addresses the following tasks:

1. Design and implement a stereo system.

2. Perform a LiDAR - stereo camera calibration. Use the LiDAR as ground truth to the depth measurements.

3. Propose a far range calibration procedure and evaluate the obtained parameters. Look at important factors when calibrating at far range.
4. Implement a real-time system for object detection. Make a framework compatible with the system running on milliAmpere.
5. Choose a stereo correspondence algorithm suitable for a marine environment. Perform clustering, isolating the object of interest in the scene.
6. Implement an object recognition procedure using CNN and evaluate the reliability of the network.
7. Integrate the system on the autonomous ferry milliAmpere and test it in the operational environment of the vessel. Record relevant data to be used for testing of detectors.
8. Analyze the performance of the proposed system.

1.3 Report Outline

The thesis is divided into two parts. Part I is about stereo vision and the calibration of a stereo camera. The part begins with Chapter 2 which describes the theoretical background of stereo vision. The chosen stereo setup is described along with the cameras in use. Chapter 3 presents the LiDAR which serves as a ground truth for distance estimates. Chapter 4 propose an extrinsic calibration procedure for far range stereo vision. Finally in Chapter 5, the resulting parameters are presented, and evaluated based on their performance on scenes of different depths.

Part II is about the application on the ferry in the marine environment. An overview of the system implemented on milliAmpere is given in Chapter 6. Chapter 7 describes the techniques used for object detection. The implemented algorithms for depth-maps and clustering are presented and an evaluation of the reliability is given. In Chapter 8, the results of testing in coastal and harbour environments is presented and discussed. A conclusion for both parts is given in Chapter 9, along with suggestions for future work.

Part I

Stereo vision and calibration

Chapter 2

Stereo vision

Stereo vision is a computer vision technique used to perceive depth in a scene by combining two or more sensors. With two cameras of known relative position, the 3D position of points projected on the camera plane can be estimated through triangulation. Working with binocular cameras requires an understanding of the calculations and setup, which both influence the 3D points probability distribution. As well, the application area and the choice of algorithms used for computing and solving the correspondence problem will affect the resulting depth-map and the associated point cloud. The chapter rephrases from the two specialization projects written by Theimann (2020) and Olsen (2020).

2.1 Monocular camera

2.1.1 Pinhole model

The pinhole model is one of the most widely used geometric camera models (Hartley and Zisserman, 2004). The mathematical model describes the correspondence between real-world 3D points and their projection onto the image plane. Figure 2.1 demonstrates the concept, where the box represents the camera body. The rear inside of the camera body places the image plane. The light travels from the top of the scene, straight through the aperture, and projects to the bottom of the camera film. The aperture lets only light rays from one direction through at a time. Thus light rays from conflicting directions are filtered out. The camera film, therefore, holds an inverted image of the world.

As there is no lens, the whole image will be in focus. The focus requires an infinity long exposure time to avoid blur and absence of moving objects. In practice, one equips the camera with a lens to produce useful images. The pinhole is mathematically convenient, and the geometry of the model is an adequate approximation of the imaging process.

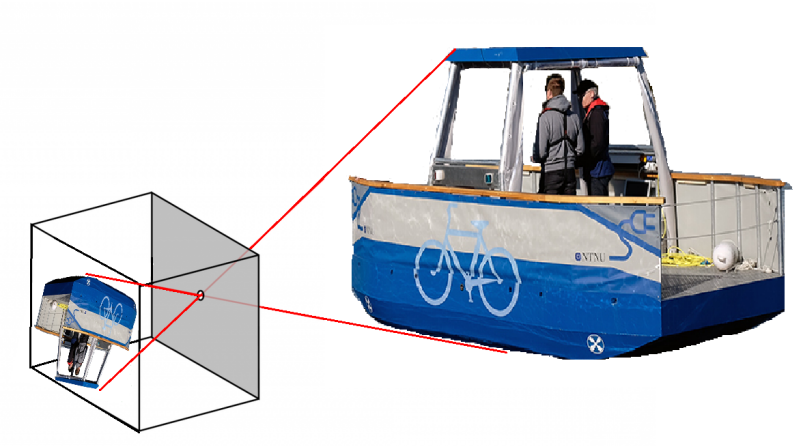


Figure 2.1: Pinhole model

2.1.2 Camera parameters

The camera parameters describe the relationship between the camera coordinate system and real-world coordinates. In image transformations, homogeneous coordinates are convenient for representing geometric transformations. Projective geometry has an extra dimension to describe Cartesian coordinates in Euclidean space. They allow affine transformations and can represent numbers at infinity. Mapping a homogeneous vector to a Cartesian space and Cartesian to homogeneous is given by (2.2) and (2.1) respectively (Hartley and Zisserman, 2004).

$$\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix} \longrightarrow \tilde{\mathbf{x}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.1}$$

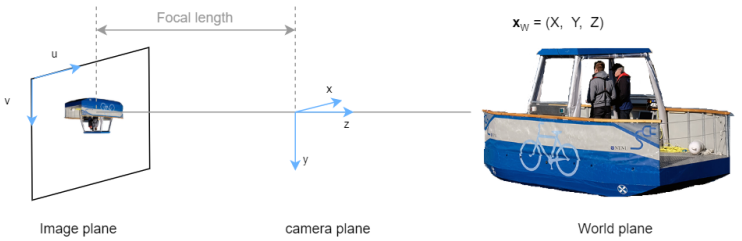


Figure 2.2: Intrinsic and extrinsic geometry

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

The camera parameters are divisible into two groups - internal and external parameters.

Internal parameters

The internal camera parameters comprehend both the intrinsic- and distortion parameters. The intrinsic matrix transforms 3D camera coordinates to 2D homogeneous image coordinates. In Figure 2.2, the image plane is parallel to the camera plane at a fixed distance from the pinhole. The distance is named 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. The perspective projection models the ideal pinhole camera, where each intrinsic parameter describes a geometric property of the camera

The pinhole model assumes pixels to be square, which may not apply for alternative camera models. Commonly, cameras are not exact pinhole cameras, and CCD (Charged-coupled device) is the model generally used in digital cameras. Nevertheless, by a few adjustments, the pinhole model can fit in the CCD model. CCD cameras may have non-square pixels, which can cause an unequal scale factor in each direction. Thus, two additional constants are added, m_u and m_v. The constants defines the number of pixels per unit distance in image coordinates, u and v direction, respectively.

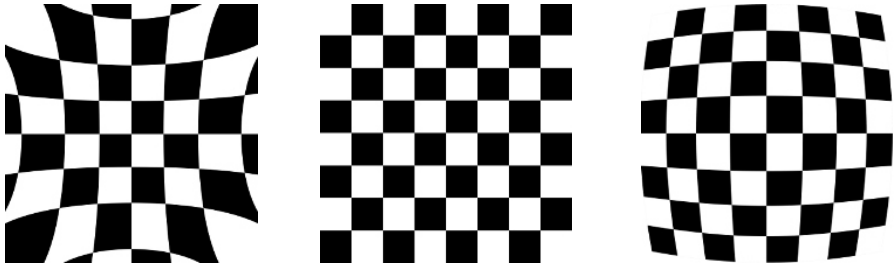
$$\mathbf{K} = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{aligned} \alpha_u &= m_u f \\ \alpha_v &= m_v f \\ u_0 &= m_u p_u \\ v_0 &= m_v p_v \end{aligned} \tag{2.3}$$

The adaptive intrinsic matrix is given in (2.3). The parameters α_u and α_v represents the focal length in pixel units, and u_0 and v_0 symbolize the principal point in pixel units (Hartley and Zisserman, 2004). The s is named the skew parameter. It is non-zero in cameras where the image axes are not perpendicular, usually not the case for today's cameras.

Image distortion makes straight lines in a scene appear bent in the image. The optical distortion is derived in the lens and occurs when lens elements are used to reduce aberrations. An ideal pinhole camera does not have a lens, and thus it is not accounted for in the intrinsic matrix. Radial distortion occurs when light rays are bent more near the edges of the lens than in the center. Figure 2.3 shows how radial distortion can affect an image. Let (u, v) be the ideal points and (u_d, v_d), the radial distortion expressed in (2.4).

$$\begin{aligned} u_d &= u + u\left[k_1(u^2+v^2) + k_2(u^2+v^2)^2 + k_3(u^2+v^2)^3 ...\right] \\ v_d &= v + v\left[k_1(u^2+v^2) + k_2(u^2+v^2)^2 + k_3(u^2+v^2)^3 ...\right] \end{aligned} \tag{2.4}$$

The radial distortion coefficients $k_n = n^{\text{th}}$ express the type of radial distortion (Zhang, 2000).



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

Figure 2.3: Radial distortion

Extrinsic parameters

The extrinsic parameters relate the camera plane with the world reference frame, see Figure 2.2. The two coordinate frames are related by the homogeneous transformation matrix $\mathbf{T}$. The matrix consist of a translation matrix and a rotation vector combined.

When describing the intrinsic parameters, the world coordinates were assumed to be given in the camera frame. To describe the extrinsic parameters, the scene points are expressed in the world coordinate frame (X, Y, Z). A point, $\mathbf{x}_\text{W}$ in the world frame can be expressed as a point, $\mathbf{x}_\text{C}$, in the camera frame by

$$\mathbf{x}_\text{C} = \mathbf{R}\mathbf{x}_\text{W} + \mathbf{t} \quad \Longleftrightarrow \quad \mathbf{x}_\text{C} = \mathbf{T}\mathbf{x}_\text{W}$$

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

The camera matrix is a result of combining the intrinsic parameters $\mathbf{K}$ (2.3) with the extrinsic $[\mathbf{R}, \mathbf{t}]^T$. It relates world coordinates with pixel coordinates and is notaded with $\mathbf{P}$ (2.5).

$$\mathbf{P} = \begin{bmatrix} \mathbf{R} \\ \mathbf{t} \end{bmatrix} \mathbf{K} \tag{2.5}$$

2.1.3 Blackfly S GigE

The thesis utilizes two identical cameras in the stereo setup. They are delivered by FLIR, and the camera model is Blackfly S GigE. The selected lens was bought from Edmund

Optics and is in the C Series. Relevant specifications are written in Table 2.1, for further reading, see the suppliers' website[1] [2].

Camera specification	
Frame rate	24 FPS (min. 1 FPS)
Resolution	2448 x 2048
Pixel Size	$3.45\mu m$
Sensor	Sony IMX264, CMOS, $2/3$"
Readout Method	Global shutter
Interface	GigE
Operating Temperature	$0°C$ to $50°C$
Exposure Range	$13\mu s$ to $30s$
Dimensions (W x H x L)	$29mm$ x $29mm$ x $30mm$
Power Requirements	Power over Ethernet (PoE); or via GPIO

Lens specification	
Focal Length	$8.5mm$
Type	Fixed Focal Length Lens

Table 2.1: Camera and Lens spesicifation

Reading from Table 2.1, the sensor uses the Gigabit Ethernet or "GigE" interface, which is known to combine multiple cameras easily. It is the most common interface for industrial image processing, and offer the possibility for receiving power through the GPIO pins or with Power over Ethernet. The sensor type is CMOS, and it utilizes a global shutter to capture an entire frame all at once. Since the cameras are to be deployed on a moving vessel, this enables capturing all of the motion simultaneously, resulting in no rendering of slanted lines which should have been straight.

The sensor format, combined with the lens specification determines the field of view. From the specifications, each camera calculates to have a field of view of 59.1 degrees. The wide field of view, together with the high resolution, allows the cameras to detect and recognize objects at great distances. The disadvantage is that the number of pixels is tremendous and, therefore, the computational burden when iterating through the pixels in stereo processing. Moreover, the pixel size is microscopic, which gives rise to noisier images in case of little light. Considering that the image processing will have a notable higher runtime than $\frac{1}{24}$ seconds, the frame rate is acceptable even if one decides to lessen the image resolution.

The focus and "iris" of the sensor are manually adjusted by the lens, and can later be tuned in software. The camera supports color images, but considering the stereo matching algorithms is intensity-based, the camera is set to capture greyscale images to maximize the frame rate. Greyscale images are the same as intensity-based images. A color image

[1] https://www.flir.com/products/blackfly-s-gige/?model=BFS-PGE-50S5C-C

[2] https://www.edmundoptics.com/p/85mm-c-series-fixed-focal-length-lens/14947/

transforms into greyscale by summing all the colored layers, $\frac{R+G+B}{3}$.

The cameras are not waterproof, and together with the high operating temperature, it will be necessary to make a waterproof and isolating case for protecting the cameras in the operating environment.

It is preferred to set the manual exposure and focus of the cameras approximately equal to ease the stereo matching. Similar camera parameters will reduce noise when comparing corresponding pixels in the image processing part. Similar, it is essential to synchronize the capturing of the two cameras, so images will appear in the correct order when post-processing.

2.2 Stereo setup

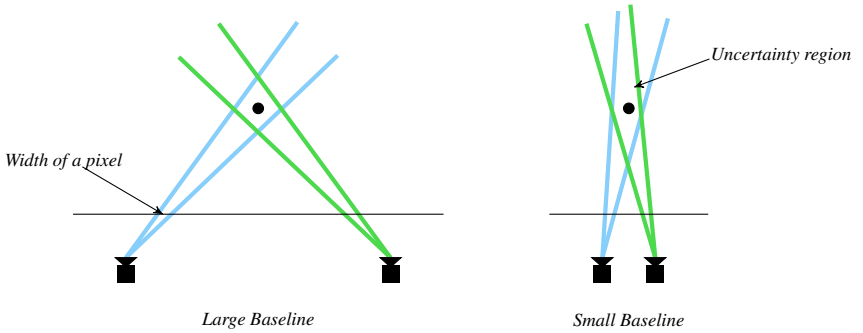


Figure 2.5: Illustrating uncertainty depending on the baseline

Representing the 3D world with a discrete camera plane give rise to uncertainty in the depth estimate. In Figure 2.5 the uncertainty with different baselines is demonstrated. In general, the uncertainty will decrease with expanding baseline. In addition to the baseline, the vergence angle of the cameras will affect the shape of the uncertainty region. Changing the angle between the cameras will result in parallel rays give less precise localization of 3D points. The accuracy of reconstructed 3D points is thus dependent on both the baseline and the angle, where the width of the probability distribution will decrease with increasing baseline and vergence angle.

As the accuracy increase with increasing baseline and vergence angle, the search problem becomes more difficult, and the field of view shrinks. The cameras' angling and baseline directly influence the field of view (McCabe, 2016). Depth estimation is possible only for the overlapping field of view between the two camera views, and in practice, a minimum of $\frac{2}{3}$ of each image should overlap. Three different setups are demonstrated in Figure 2.6. Illustrated in green is the field of view. The overlapping field of view will diminish with a more substantial baseline and concurrently increase the blind spot. The rightmost figure titled *Angled* demonstrates a solution when using a wide baseline for far-range detection. When cameras angle inward, the field of view increases and withal decreases the blind spot.

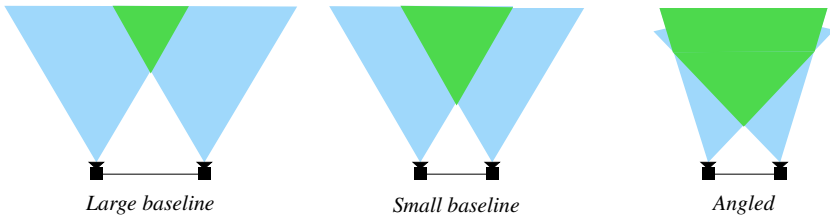


Figure 2.6: Illustrating field of view in a stereo setup

Another benefit of angling the cameras is that the vergence angle determines a particular fixation point in the scene. When the fixation point is near a target object, it can help in separating the target objects from distracting surroundings. The working distance, i.e., how far the setup is from a given object or fixation point, can be calculated depending on the camera optics.

$$\mathcal{WD} = \frac{f * FOV_h}{h}, \qquad \begin{aligned} FOV_h &= \textit{horizontal field of view} \\ h &= \textit{horizontal sensor size} \\ \mathcal{WD} &= \textit{working distance} \\ f &= \textit{focal length in metric units of length} \end{aligned}$$

The baseline is directly proportional to the horizontal distance from the mounting to the object. Thus stereo systems with smaller baseline achieve better results when detecting objects on average distances, whereas broader baseline is preferred with greater distances. As a general rule, the fraction $\frac{1}{30}$ is used to estimate the dimension of the baseline (Curtin, 2011). For a working distance on 30 meters, the baseline would be about one meter, depending on the cameras' angling.

Eventually, the setup is a matter of the distance one wants to detect, the width and blind spots, which are all correlated. The factors all define what level of depth-accuracy the system will achieve.

2.2.1 The chosen stereo setup

The stereo setup is designed for the ferry milliAmpere. Considering restrictions and a user-friendly design, the maximum baseline possible is approximately 1.80 meters. This corresponds to a preferred operating distance at 54 meters when cameras are not angled. As presented in Section 2.2 angling the cameras will minimize the error due to the uncertainty region of a pixel. Thus, the cameras were angled, giving a fixation point at 50 meters. With a baseline of 1.80 meters, the vergence angle of each camera is calculated by Pythagoras to be 1.00 degrees.

Figure 2.7 shows an illustration of the field of view of the stereo camera. This setup

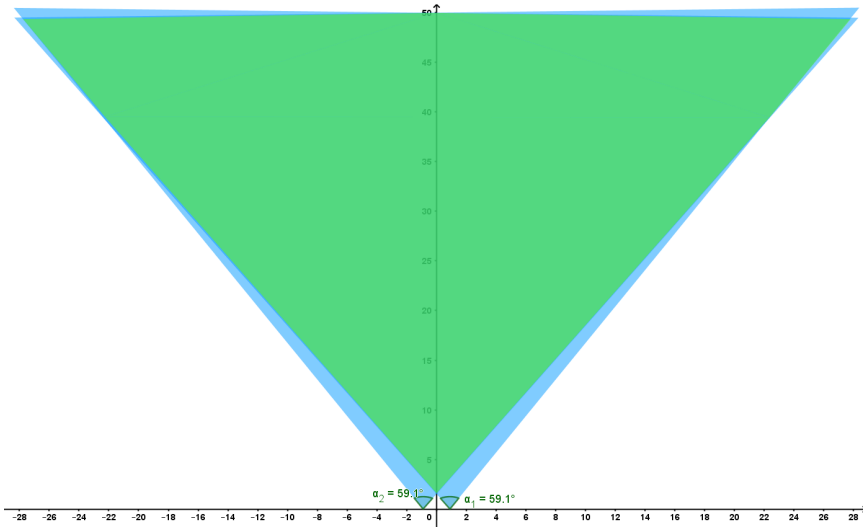


Figure 2.7: Field of view of stereo setup

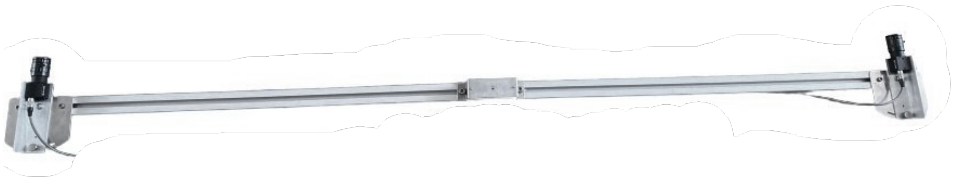


Figure 2.8: Stereo setup

gives a blind spot of approximately 1.6 meters in front of the cameras. At 50 meters, a horizontal FOV of approximately 50 meters is visible.

An attempt of manual measuring the actual baseline was made to get pre-knowledge about the resulting calibration parameters. The baseline was measured to be 1800.25 millimeters. In addition, the cameras are manually angled 1 degree towards each other. Thus the expected rotation in Euler angles and translation of the right camera according to the left would be:

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} x:0 & y:-2 & z:0 \end{bmatrix} \\ \mathbf{t} &= \begin{bmatrix} -1800 & 0 & 0 \end{bmatrix} \end{aligned} \tag{2.6}$$

As the measurements are done by hand, there are uncertainties in the numbers. However, they give a pointer towards the intended extrinsic calibration parameters

2.3 Epipolar geometry

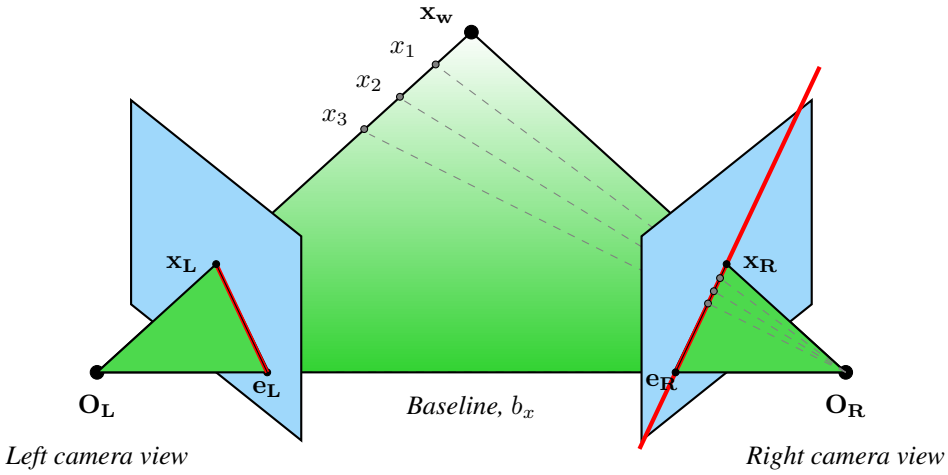


Figure 2.9: Epipolar geometry of two cameras

Observing the same world point $\mathbf{x}_\text{W}$ from two relative views gives a strong geometric constraint on the observed point called the epipolar constraint. By the geometric pinhole model, triangulation retrieves 3D world coordinates. Figure 2.9 gives an illustration of the epipolar geometry between two cameras. Blue illustrates the image planes, and the point $\mathbf{x}_\text{W}$ is the world coordinate desired to reconstruct the depth.

The green triangular in Figure 2.9 connects the two optical centers, $\mathbf{O_L}$ and $\mathbf{O_R}$, and the point $\mathbf{x}_\text{W}$; it is called the epipolar plane. The baseline, a fixed-length depending on the stereo setup, is the line connecting the two optical centers. The epipolar plane intersects through the two blue image planes forming the red epipolar line in each plane. The epipolar line in the left camera-view intersects the epipole $\mathbf{e_L}$ and the point $\mathbf{x}_\text{L} = (u_L, v_L)$.

The epipolar constraint uses the epipolar line to find pixel correspondences. Capturing a point in one of the cameras, the pixel point in the image generates the epipolar line in the second image. The corresponding pixel must lie on the line. Thus, search for correspondences is reduced from a region to a line. The constraint arises by the reason that when two cameras capture the same world object, the pixel points, the world point, and the optical centers are all coplanar. The depth is thus directly proportional to the baseline and the distance between the two pixels capturing the world object.

The epipolar constraint can be represented mathematically by the essential matrix, $\mathbf{E}$, and the fundamental matrix, $\mathbf{F}$. Both the matrices describe the geometric relationship entirely (Hartley and Zisserman, 2004). The essential matrix depends only on the extrinsic parameters, while the fundamental use image coordinates. In other words, the essential matrix represents the constraint in normalized image coordinates, while the fundamental matrix represents parameters in pixel coordinates. It is convenient to describe a normalized image plane as it represents an idealized camera. Setting $z = 1$ in front of the projective center, in Figure 2.2, implies a focal length of 1. This lets us describe the imaging process independently of the camera-specific focal length parameter.

Correspondence between the two homogeneous represented points $\mathbf{x}_\mathrm{L}$ and $\mathbf{x}_\mathrm{R}$ can mathematically be presented in normalized image coordinates (2.7).

$$\mathbf{x}_\mathrm{L}\mathbf{E}\mathbf{x}_\mathrm{R} = 0, \quad \text{where} \quad \mathbf{E} = [\mathbf{t}^\mathrm{L}_\mathrm{LR}]_\times \mathbf{R}_\mathrm{LR} \tag{2.7}$$

The essential matrix is related to the pose between the left and right image planes in Figure 2.9. The corresponding rigid body transformation $\mathbf{T}$ is defined as

$$\mathbf{T}_\mathrm{LR} = \begin{bmatrix} \mathbf{R}_\mathrm{LR} & \mathbf{t}^\mathrm{L}_\mathrm{LR} \\ \mathbf{0} & 1 \end{bmatrix}$$

The fundamental matrix represent the same epipolar constraint in the image plane. It is the algebraic mapping between a pixel in one image and the epipolar line. Corresponding pixels in the left and right image is again established under the epipolar constraint, though via the intrinsic calibration matrices $\mathbf{K}_L$ and $\mathbf{K}_R$ of each camera.

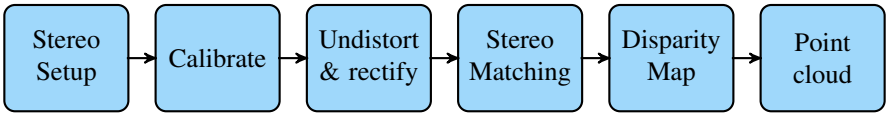
$$\mathbf{F}_\mathrm{LR} = \mathbf{K}_\mathrm{L}^{-\top}\mathbf{E}_\mathrm{LR}\mathbf{K}_\mathrm{R}^{-1} \tag{2.8}$$

The theory is illustrated in Figure 2.9, where the pixel $\mathbf{x_L}$ in the left view is mapped to all the pixels on the epipolar line of the right camera. In practice, every pixel on the epipolar line in the right camera view capturing the world points $\mathbf{x}_\mathrm{W}$, x_1, x_2, or x_3 can correspond to the one pixel $\mathbf{x}_\mathrm{L}$ in the left view. The problem of selecting the correct matching pixel in the right view is termed the correspondence problem.

2.4 Correspondence problem

In stereo images, depth is inversely proportional to the disparity between pixels. The proposition implies that calculating depth is a single trivial equation. However, in order to do the calculation, one must determine for each pixel in one image which pixel in the second originated from the same 3D world point. This issue is known as the correspondence problem and is a central part of binocular vision.

Before stereo matching and solving the correspondence problem, the images should be undistorted and rectified to ease the computation of finding corresponding pixels. The flow chart below illustrates the pipeline for retrieving depth from a stereo camera.



The stereo calibration obtains the rigid body transformation between the two cameras and can be used to undistort and rectify the images. Rectification is the process of transforming the two images onto a common image plane, aligning the red epipolar lines in Figure 2.9. It ensures that the corresponding pixels in the image pair are on the same row.

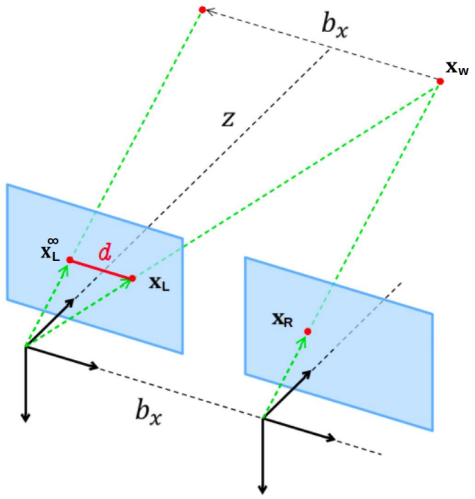


Figure 2.10: Ideal Stereo Geometry

The concept is shown in Figure 2.10. By defining new camera matrices (2.5), the correspondence problem is mapped to ideal stereo geometry. The mapping is accomplished by setting the rotation matrix equal to the identity matrix and expressing the translation along the x-axis. The ideal stereo geometry is a special case of the epipolar geometry where the epipoles are at infinity, meaning $\mathbf{x}_\mathrm{L}^\infty = \mathbf{x}_\mathrm{R}$. The cameras will thus have identical orientation, and baseline b_x along the x-axis. Ideal stereo geometry gives the following pixel correspondence;

$$\mathbf{x}_\mathrm{L} = \mathbf{x}_\mathrm{R} + \begin{bmatrix} \frac{f * b_x}{d} \\ 0 \\ 0 \end{bmatrix} \tag{2.9}$$

The pixel correspondence implies that the epipolar lines become horizontal. Looking for corresponding pixels in the two images boils down to a one-dimensional search.

Various stereo matching algorithms can perform the matching of corresponding pixels. Stereo matching is the task of establishing correspondences between two images of the

same scene. The pixel distance between matching points is used to create a disparity map. Disparity maps are applicable in various applications with a broad range of input data. The depth estimate from the disparity can thus vary remarkably depending upon the properties of the data (Praveen, 2019). Therefore, a broad range of algorithms has been developed for acquisition corresponding points and subsequent for matching the points to estimate depth. In general, the methods are classified into local and global approaches. The local methods consider pixels in a local neighborhood, while the global methods use the information presented in the entire image. Global algorithms are primarily based on the optimization of a well-behaved-energy function created from the hole image (Liu and Aggarwal, 2005). Because the energy function is defined on all the pixels of the image, global methods are less sensitive to local ambiguities than local methods (Liu and Aggarwal, 2005). In short, the global methods may achieve more accurate disparity maps, while the local methods yield a less computational cost.

2.4.1 Disparity map

After matching corresponding pixels, a disparity map expresses the disparities. Disparity map, also commonly named range- or depth-map, is a representation of the displacement between conjugate pixels in the stereo pair image. It represents the motion between the

Figure 2.11: Example of scene with corresponding grey-scale disparity map and Point cloud

two views. An example of a disparity map is illustrated in Figure 2.11. The smaller the disparities, the darker the pixels, therefore less motion and longer distance.

$$X = \frac{u_L - u_R}{f} Z, \qquad Y = \frac{v_L - v_R}{f} Z, \qquad Z = \frac{f * b_x}{d} \tag{2.10}$$

After extracting the disparity map, the 3D world position of a given object only depends on straightforward mathematics. The triangulation between pixels in the left- (u_L, v_L) and right-image (u_R, v_R) is given in (2.10). The position is given in the cameras coordinate frame, where b_x is the baseline of the stereo camera, f, the focal length, and d corresponds to the disparity between the two images.

The 3D world coordinates calculated can further be plotted together, creating a point cloud (Rusu and Cousins, 2011). Point clouds are a means of collating a large number of single spatial measurements into a dataset that can then represent a whole. One can say

that the point cloud displays the back-projection of the stereo image pair; an example is displayed in Figure 2.11.

2.4.2 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 (2005) matches each pixel based on Mutual Information (MI) and the approximation of a global smoothness constraint.

The idea is to perform a line optimization by considering pairs of images with known epipolar geometry. As radiometric differences often occur in stereo image pairs, the dissimilarity of corresponding pixels is modeled using MI. The entropy of the two images defines Mutual Information. It is calculated from the probability distributions of the intensities. The model performs a joint histogram of corresponding intensities over the corresponding parts based on the epipolar geometry. The resulting definition of MI is defined in (2.11).

$$\mathrm{MI}_{L,R} = \sum_{\mathbf{x}_\mathrm{L}} \mathrm{mi}_{L,R}((u_\mathrm{L}, v_\mathrm{L}), (u_\mathrm{R}, v_\mathrm{R})) \tag{2.11}$$

$$\mathrm{mi}_{L,R} = h_L(i) + h_R(k) - h_{L,R}(i,k) \tag{2.12}$$

The probability distributions, h_L, h_R, and $h_{L,R}$ are summed over the corresponding rows and columns in the corresponding parts. The last expression $h_{L,R}$ in (2.12) serves as cost for matching the intensities $I_L(\mathbf{x}_\mathrm{L})$ and $I_R(\mathbf{x}_\mathrm{R})$. In stereo images, some pixels may not have any correspondences, thus the two first expressions in (2.12).

The pixel-wise cost calculation is calculated based on pixel intensity (2.13). The pixel-wise MI matching determines the absolute minimum difference of intensity at $\mathbf{x}_\mathrm{L}$ and $\mathbf{x}_\mathrm{R}$.

$$\mathrm{C}_\mathrm{MI}(\mathbf{x}_\mathrm{L}, d) = -\mathrm{mi}_{L,R}(I_L(\mathbf{x}_\mathrm{L}), I_R(\mathbf{x}_\mathrm{R})), \quad \mathbf{x}_\mathrm{R} = \mathbf{x}_\mathrm{L} + d \tag{2.13}$$

Calculating the cost pixel-wise can often lead to wrong matches, due to, e.g., noise. For this reason, a penalization of changes of neighboring disparities is added (2.14). The implementation is a smoothness constraint, or energy function.

$$\mathrm{E}(D) = \sum_{\mathbf{x}_\mathrm{L}} C(\mathbf{x}_\mathrm{L}, D_{\mathbf{x}_\mathrm{L}}) + \sum_{\mathbf{x}_\mathrm{R} \in N_{\mathbf{x}_\mathrm{L}}} P_1 T(|D_{\mathbf{x}_\mathrm{L}} - D_{\mathbf{x}_\mathrm{R}}| = 1) + \sum_{\mathbf{x}_\mathrm{R} \in N_{\mathbf{x}_\mathrm{L}}} P_2 T(|D_{\mathbf{x}_\mathrm{L}} - D_{\mathbf{x}_\mathrm{R}}| > 1) \tag{2.14}$$

The equation defines the energy function of the disparity image D. The first term sums the pixel-wise matching cost. The two following terms add a penalty cost for all pixels in the right image, which are in the neighborhood $N_{\mathbf{x}_\mathrm{L}}$ of the pixel $\mathbf{x}_\mathrm{L}$ in the right image. The constant penalty is multiplied for unique matching, i.e., to penalize discontinuities. The operator T is defined as the probability distribution of corresponding intensities.

The energy function approximates a global smoothness constraint, and the disparity map D can be found by minimizing the energy (2.14). However, as this global minimization (2D) is computational an NP-complete problem for a frequent number of disparity

maps, the MATLAB implementation uses a slightly modified version. By matching the smoothed cost in only one dimension, all directions are weighting equally (Hirschmüller, 2005). The matching costs are connected by calculating cost paths. Thus the cost of each 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.

Overall, Semi-global matching is calculated for each pixel and is insensitive to illumination changes. It is a semi-global method, which is shown to combine global- and local-concepts successfully (Hirschmüller, 2011).

Chapter 3

Ground truth

Ground truth refers to the accuracy of the data collected. In distance estimation, the ground truth is the real measure and tells how accurate the distance estimation is. When using stereo vision to acquire depth information, a good ground truth can be obtained from a 3D Light Detection and Ranging, LiDAR, sensor. 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 reasonable estimate of how exact the stereo depth estimates are.

3.1 Light Detection and Ranging - LiDAR

Light Detection and Ranging (LiDAR) is a Time of Flight distance sensor. The LiDAR emits a laser pulse, and the reflected light energy is returned and recorded by the sensor. The sensor 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. (Velodyne, 2019)

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

A 3D LiDAR measures azimuth, elevation and range of the reflection. This gives us a 3D point in spherical coordinates, see illustration in Figure 3.1. In the figure, φ is the azimuth angle, θ is the elevation angle and r the distance from the point to the LiDAR. Equation (3.1), expresses a spherical point in Cartesian coordinates in.

$$\begin{aligned} \mathrm{X} &= r\cos(\theta)\sin(\varphi) \\ \mathrm{Y} &= r\cos(\theta)\cos(\varphi) \\ \mathrm{Z} &= r\sin(\theta) \end{aligned} \tag{3.1}$$

The LiDAR outputs a point cloud, and it contains the recorded 3D points given in spherical coordinates.

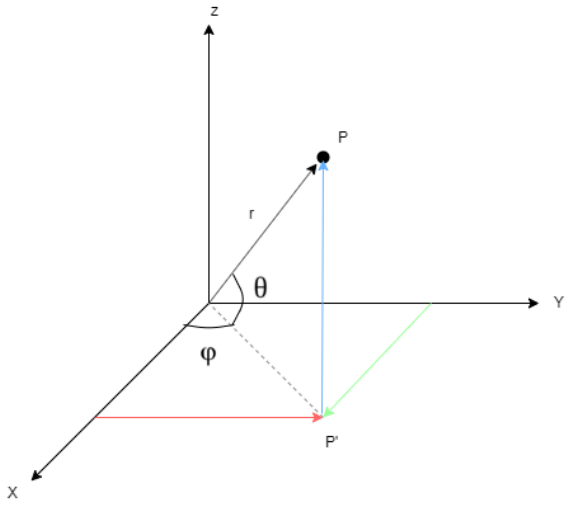


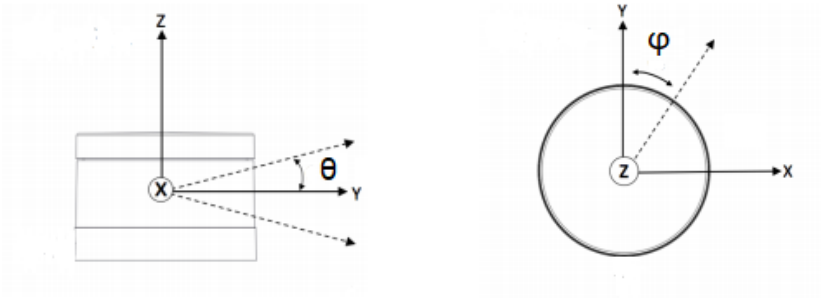
Figure 3.1: Spherical coordinates

Velodyne LiDAR VLP-16

Figure 3.2: Velodyne LiDAR Puck 16

The LiDAR used in this project is the Velodyne LiDAR Puck, VLP 16, see Figure 3.2. Its specifications are given in (Velodyne, 2019). It is a 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 range accuracy of ± 3 cm making it very accurate. The VLP 16 has 16 laser and 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°, depending on the rotation rate. The rotation rate can be chosen as a value between 300 and 1200 RPM.

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 Figure 3.3. Here φ represents the azimuth angle and θ the elevation angle. As described



(a) Side view **(b)** Top view

Figure 3.3: The LiDAR coordinate frame

in (3.1) a 3D point $\mathbf{p}$ from the LiDAR can be represented in the LiDAR frame as:

$$\mathbf{p} = \begin{bmatrix} rcos(\theta)sin(\varphi) \\ rcos(\theta)cos(\varphi) \\ rsin(\theta) \end{bmatrix} \quad \text{where} \quad \mathbf{p} = \begin{bmatrix} \mathrm{X} & \mathrm{Y} & \mathrm{Z} \end{bmatrix}^\top \tag{3.2}$$

3.2 LiDAR - stereo camera calibration

Both the LiDAR and the stereo camera outputs a point cloud relative to the sensor. To get a ground truth from the LiDAR, a direct comparison of the point clouds is necessary. Thus, the rigid body transformation between the sensors needs to be obtained. To obtain a point cloud from the stereo camera a calibration must be performed. The calibration is performed by utilizing MATLAB's stereo calibration app. Wang et al. (2019), among others, justifies MATLAB's stereo calibration method as a high precision calibration method and uses it as ground truth to their research. Thus, the app is used to calibrate the stereo camera.

MATLAB's calibration app requires a checkerboard as a calibration target and states that the optimal checkerboard covers approximately twenty percent of the field of view of the camera. A checkerboard of the size 1.5×3 meters was constructed to obtain an optimal calibration. This is the largest size feasible to make, seeing as it needs to be transportable, and the surface needs to be smooth, flat, and rigid. Placing the checkerboard at a depth of approximately 7.6 meters, it is visible in both cameras, and a proper stereo calibration can be performed.

To find the rigid body transformation between the stereo camera and the LiDAR a comparison of the point clouds is made. First, the LiDAR coordinate frame must be transformed into the camera frame. Comparing the LiDAR coordinate system in Section 3.3 with the camera frame, all the axes must be rotated, and the positive direction of the height must be switched. Now, the rigid body transformation is found by a point cloud registration. Point cloud registration is the process of finding the transformation that aligns two point clouds of the same scene. The technique used, is the Normal-distribution transform.

Figure 3.4: Checkerboard for stereo calibration

3.2.1 Normal-distributions transform

The normal-distributions transform, NDT, was first introduced as a 2D scan registration method by Biber and Strasser (Biber and Straßer, 2003). Magnusson presented a method for registration a point cloud with a reference point cloud using NDT in (2009). NDT uses a probability representation for describing the shape of a surface. Given a set of points of a surface the points are divided into cells of the same size. The probability of a point $\mathbf{p}$ belonging to a cell is given by the normal distribution (3.3).

$$f(\mathbf{p}) = \frac{1}{c_0} e^{-\frac{(\mathbf{p}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{p}-\boldsymbol{\mu})}{2}} \tag{3.3}$$

The mean of a cell is given by $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_{ref_i}$ and the covariance by $\boldsymbol{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{p}_{ref_i} - \boldsymbol{\mu})(\mathbf{p}_{ref_i} - \boldsymbol{\mu})^\top$, where $\mathbf{p}_{ref_i}$ is the points of the reference point cloud within the cell. The constant c_0 is used to scale the function so that its integral is equal to one. Each probability density function is an approximation of the local surface and thus NDT represents the point cloud in a piecewise smooth manner.

The objective for registration is to find the transformation of a point cloud. The solution is the registration that maximises the likelihood function (3.4) of the points lying on the surface of the reference point cloud.

$$\Psi = \prod_{k=1}^{n} f(\mathbf{T}(\mathbf{p}_k)) \tag{3.4}$$

An equivalence to maximizing the likelihood function would be to minimize the log-likelihood of Ψ. Outliers, points far from the mean, cause the negative log-likelihood of a normal distribution to grow boundlessly. To reduce their influence on the result, a uniform

distribution is used to represent outliers. The probability of a point $\mathbf{p}$ belonging to a cell can now be rewritten (3.5).

$$\bar{f}(\mathbf{p}) = c_1 e^{-\frac{(\mathbf{p}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{p}-\boldsymbol{\mu})}{2}} + c_2 f_o \tag{3.5}$$

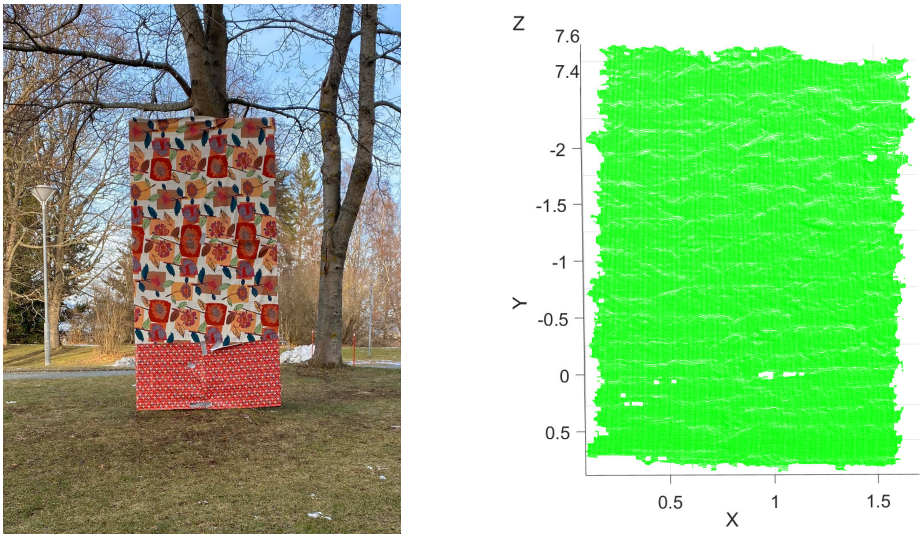
The f_o is the expected ratio of outliers. By demanding that within a cell, the probability mass of $\bar{f}(\mathbf{p})$ has to be equal to one the constants c_1 and c_2, can be determined. Thus, given a point cloud and a transformation $\mathbf{T}(\mathbf{p})$, a cost function is created (3.6).

$$\mathcal{C}(\mathbf{T}(\mathbf{p})) = -\sum_{k=1}^{n} \tilde{f}(\mathbf{T}(\mathbf{p}_k)) \tag{3.6}$$

Where $\tilde{f}(\mathbf{p})$ is the Gaussian approximation of $\bar{f}(\mathbf{p})$. When $\mathbf{p}$ is transformed by $\mathbf{T}$, the cost function corresponds to the likelihood of the point lying on the surface of the reference point cloud. By solving the equation $\mathbf{H}\Delta\mathbf{T}(\mathbf{p}) = -\mathbf{g}$, Newton's method is used to find the optimal $\mathbf{T}(\mathbf{p})$. $\mathbf{H}$ is the Hessian and $\mathbf{g}$ the gradient of (3.6).

3.3 The ground truth

Finding the rigid body transformation between the LiDAR and the stereo camera, makes the LiDAR applicable as ground truth of the accuracy of the stereo camera. The checker-



(a) Object for LiDAR-stereo camera calibration

(b) Stereo point cloud of calibration scene

Figure 3.5: Calibration target and resulting point cloud from the stereo camera

board is suitable as a calibration target in the LiDAR - stereo camera calibration. Due to its size, it is fully visible in both cameras. Covering it with a variable pattern, makes it ideal

for the stereo camera to estimate the most exact depth as possible. The target is placed in the same distance as used for the stereo calibration, to obtain a point cloud that is as exact as possible. The resulting point cloud from the stereo camera and the calibration target is given in Figure 3.5. The result coincides with the measured distance from the camera to the calibration target.

The angling of the laser beams of the LiDAR, causes it to miss parts of the top and the bottom of the object. By angling the object horizontally, the registration of the point clouds is less prone to errors in depth. This helps to make sure that the correct corresponding points between the point clouds are found and used to calculate the rigid body transformation. NDT performs well in the registration of point clouds of different densities. This due to NDTs probability representation of the point cloud. The distance to the target causes the LiDAR point cloud to be sparser, thus making NDT the preferred method for point cloud registration.

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} x : 0.430 & y : 0.025 & z : 0.103 \end{bmatrix} \\ \mathbf{t} &= \begin{bmatrix} -0.940 & 0.132 & 0.067 \end{bmatrix} \end{aligned} \tag{3.7}$$

The registered rotation and translation is given in (3.7). The rotation is given in Euler angles. This is the transformation between the LiDAR coordinate frame and the stereo camera coordinate frame. The camera frame is in the left view, and the LiDAR is placed approximately in the middle of the two cameras. The obtained translation in x-direction places the LiDAR approximately 0.94 meters from the left camera. This seems appropriate due to the cameras being placed approximately 1.80 meters apart. The translation in y-direction corresponds to the height difference between the sensors.

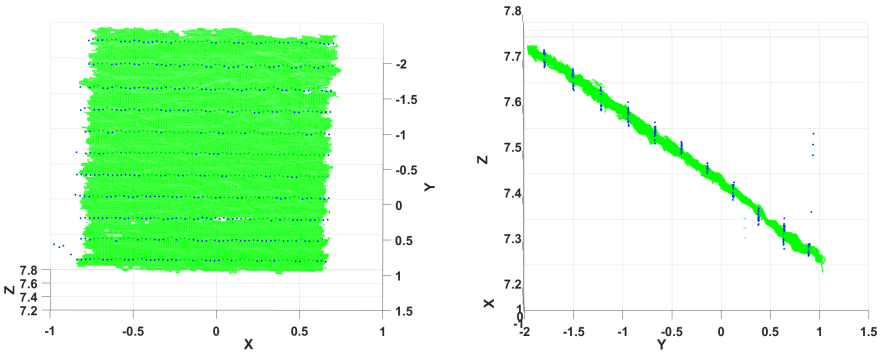


Figure 3.6: Point clouds after transforming the stereo point cloud using the obtained parameters

Figure 3.6 shows the result after the stereo point cloud is transformed using the transformation in (3.7). The LiDAR point cloud is plotted in blue and the stereo point cloud in green. In order to better illustrate the LiDAR point cloud, it is plotted using MATLABS's function `Scatter3()`, using a circle size of 6. The figure shows that the two point clouds aligns.

The root means square error of the point cloud registration is calculated to be

$$\text{RMSE} = 0.0796$$

This means that the Euclidean distance between the two point clouds after they are aligned is approximately 0.08 meters, i.e. 80 millimeters. A stereo system is expected to have meter precision at far distances, making the error negligible. The alignment of the point clouds and the low RMSE show that a valid calibration is made. The LiDAR is used as ground truth by utilizing the obtained parameters throughout the following chapters. It is used to evaluate the accuracy of the stereo camera calibrations.

Chapter 4

Stereo calibration

Calibrating a stereo camera minimizes the measurement uncertainty. It allows for the extraction of precise distance estimations. This chapter presents the theory behind the method utilized for calibrating a stereo system. The intrinsic and extrinsic parameters are estimated in two separate calibration sessions. Before performing the extrinsic calibrations for far range distances, a preliminary work is performed. This is done to give an impression of the influence of the calibration distance, as discussed in Section 1.1. Based on the findings, an extrinsic calibration procedure is proposed.

4.1 Monocular camera calibration

Monocular calibration is the process where intrinsic and extrinsic camera parameters are estimated. As well, lens distortion is modeled and corrected. The intrinsic and extrinsic parameters relate the camera coordinate system to the world coordinate frame. Using these parameters the size of world objects and the location of the camera in the scene can be determined. There exist different techniques, but the two main ones, according to Zhang (2000) 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 moving while the scene remains static. The correspondence between the images is used to estimate the intrinsic and extrinsic parameters.

4.1.1 Zhang's method

Zhang's method is a calibration technique developed by Zhengyou Zhang at Microsoft Research (Zhang, 2000). 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.

A real-world point and its image point is related by a linear transformation between planes, a homography. For every image, an estimation of the homography can be made. Each image imposes two constraints on the intrinsic and extrinsic parameters. Using more than three images a unique solution of the parameters can be computed using Singular Value Decomposition. The parameters are refined by minimizing the reprojection error using Levenberg-Marquardt Algorithm. An estimation of the radial distortion parameters is calculated by comparing the real pixel values and the ideal ones given by the pinhole model.

4.1.2 Intrinsic parameters

The internal parameters for each camera were calibrated indoors in a controlled environment. The parameters were estimated using MATLAB's calibration toolbox. By Zhang's method, the app calibrates and provides tools for evaluating and improving the accuracy. The app outputs an estimation of the intrinsic matrices and the distortion coefficients.

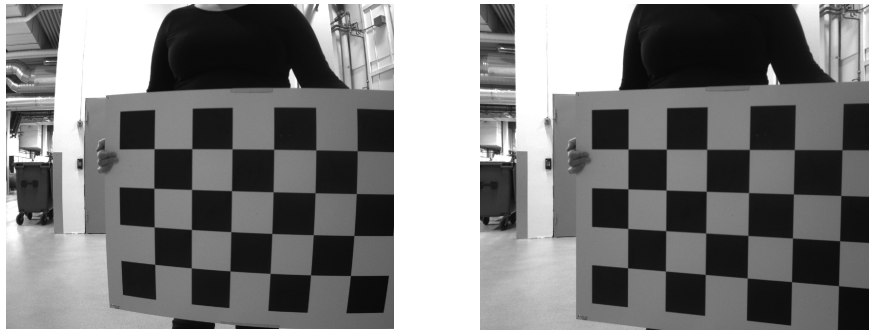


Figure 4.1: Image before and after intrinsic calibration

The resulting intrinsic and distortion parameters are given in (4.1) and (4.2), for each camera respectively.

$$\begin{aligned} \mathbf{K}_{\mathrm{L}} &= \begin{bmatrix} 1233.00 & 0 & 615.64 \\ 0 & 1232.45 & 540.21 \\ 0 & 0 & 1 \end{bmatrix} \\ \mathbf{K}_{\mathrm{R}} &= \begin{bmatrix} 1233.86 & 0 & 636.00 \\ 0 & 1233.37 & 536.00 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{4.1}$$

$$\begin{aligned} \text{Left cam:} \quad & k_1 = -0.3934, \quad k_2 = 0.1816 \\ \text{Right cam:} \quad & k_1 = -0.3910, \quad k_2 = 0.1757 \end{aligned} \tag{4.2}$$

The two matrices are sufficiently equal and the undistorted images, e.g, Figure 4.1, shows that the parameters are reasonable. In theory, the intrinsic- and radial distortion parameters for the two cameras are equal. In practice, this is not feasible due to physical differences, such as manually setting the screw-lenses. The parameters for the focal length are almost identical, but for the principal point on the x-axis, the values are noticeable divergent. Bear in mind that the principal point is defined to be the image position where the optical axis intersects the image plane. Thus, the principal point changes if the camera setting changes, e.g., changing the focus or zoom. Hence, a small distinction in the rotation of the two lenses yields different results for the principal point's parameters.

Both the skew parameters are set to 0 in MATLAB before calibrating. It is possible to allow for a non-zero skew, but with newer cameras, the skew-parameters is usually negligible. An extra free parameter to calibrate in most cases only give rise to more noise in the calibration. Performing the calibration using a non-zero skew did not give any noticeable enhancement to the result.

As the thesis focus on far-range depth estimation, the extrinsic stereo parameters are the most significant and challenging to calibrate (Ødegård Olsen, 2020). The internal parameters will thus remain fixed throughout the rest of the thesis. The internal parameters are easier to re-calibrate in a later stage. The fixed internal parameters makes it easier to compare the different stereo calibrations.

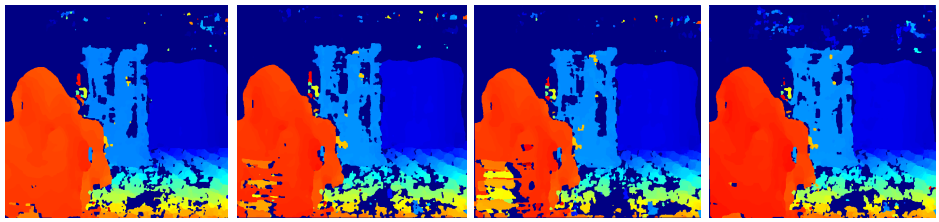
4.2 Preliminary extrinsic stereo calibration

The literature diverges regarding extrinsic camera calibration. In preparation of proposing a method, a preliminary stereo calibration is performed to evaluate the research by Marita et al., (2006), and Stretcha et al. (2008). Stretcha et al. suggests that a higher accuracy is obtained with a decreasing distance to the calibration scene. While Marita et al. found it advantageous to calibrate at the systems operating distance.

As a starting point, an experiment was set up to evaluate the literature regarding far-range stereo calibration. The stereo setup and test scene from the specialization projects were used (Theimann, 2020)(Ødegård Olsen, 2020). Calibrating only the external parameters will give an impression of how the external parameters influences the calibration on different distances. The test scene includes objects placed at a distance of approximately 2, 4, and 6 meters. The fixation point of the stereo camera is 4 meters. Thus, the external parameters are re-calibrated at four distances, 1, 2, 3 and 4 meters respectively. The LiDAR is used as ground truth for the comparison of the distance estimations.

The new external parameters are used to rectify the images before creating a disparity map of the test scene. The disparity algorithm used, Census Transform, is implemented in Olsen (Ødegård Olsen, 2020), and the resulting disparity maps of the four different calibrations are illustrated in Figure 4.3. For each disparity map, a point cloud is reconstructed by triangulation, and compared with the point cloud obtained by the LiDAR. Figure 4.4 presents the four point-clouds, were each subfigure is a result of the different external parameters. Each stereo point cloud, outlined in green, is presented together with the ground

Figure 4.2: Image stereo pair of the test scene



(a) 1m calibration **(b)** 2m calibration **(c)** 3m calibration **(d)** 4m calibration

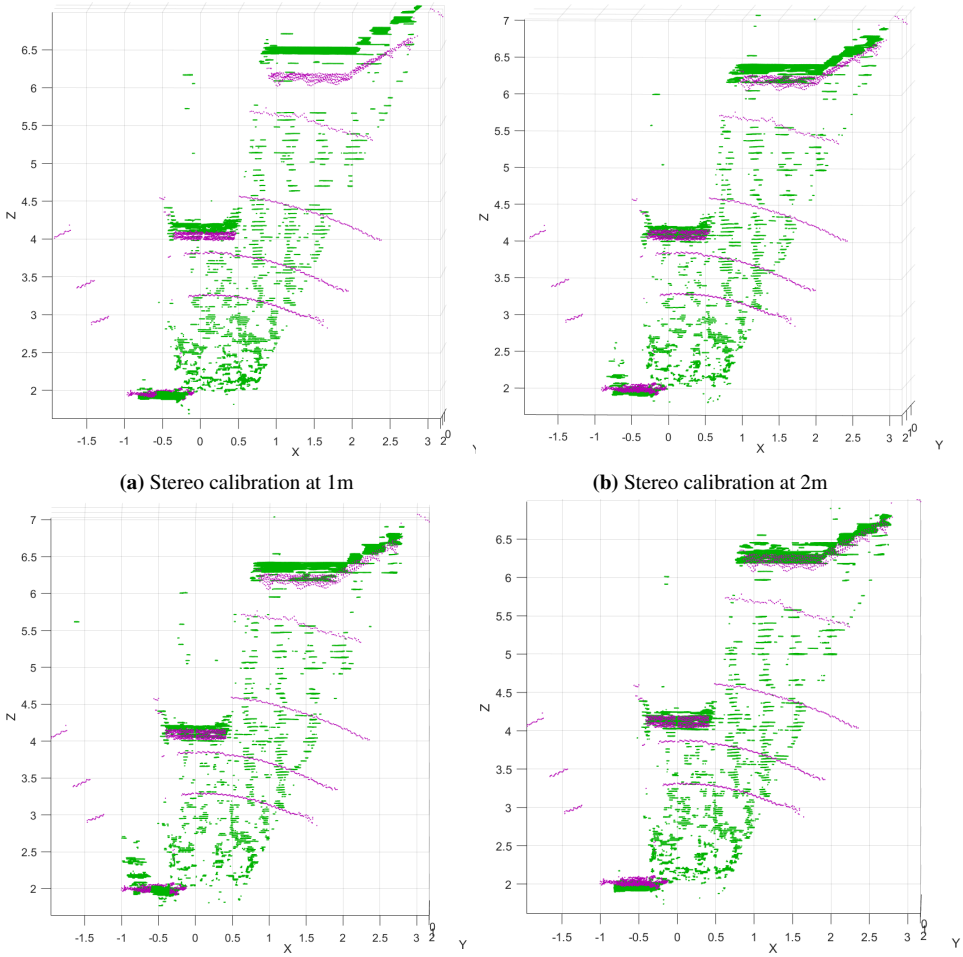
Figure 4.3: Resulting disparity map from the four calibrations

truth illustrated with purple points. As neither of the calibrations is performed at the hindmost object's distance, this is the most applicable object for comparison. The object is placed at approximately 6 meters distance. While the point clouds obtained from the calibration at 2, 3, and 4 meters overall seems rather similar, the calibration at 1 meter diverges the most from the ground truth. As expected, one can observe that the depth estimates for the farthest object deviate the most.

	Average error		
Calibration	**Object 2m**	**Object 4m**	**Object 6m**
1m	0.015 m	0.085 m	0.333 m
2m	0.021 m	0.025 m	0.109 m
3m	0.014 m	0.028 m	0.134 m
4m	0.038 m	0.015 m	0.071 m

Table 4.1: Distance in meters between selected points from LiDAR and points obtained from Semi-Global Matching

For each stereo calibration, the average distance in depth is calculated. A set of points is selected from the LiDAR point cloud, see Figure 4.5, and the corresponding stereo estimates are found. In Table 4.1, the average error between the stereo point clouds and the ground truth is presented for each of the three objects. Observe that all the depth errors



(a) Stereo calibration at 1m

(b) Stereo calibration at 2m

(c) Stereo calibration at 3m

(d) Stereo calibration at 4m

Figure 4.4: Resulting point cloud with LiDAR in purple and stereo camera in green

are positive, indicating that the stereo camera overestimates the depth. This applies to all the objects and the four calibrations.

From the table, the depth errors are relatively small. With the calibration at 1 meter, the object at 6 meters yields an error of 5.6%. At farther distances, an error of 5.6 percent is considered reasonable given the projects operating environment on marine water. However, the calibrations at 1, 2, and 4 meters perform the best on the given calibration distance. Also, calibration 3 and 4 indicate that shorter distances than the calibration distance are easier to estimate than farther distances.

The depth error for each selected point, on the object placed at 6meters, is displayed in Figure 4.6. The four calibrations are given on the x-axis, and the points present the variation in the data. Considering both variation and mean, the calibration performed at 4

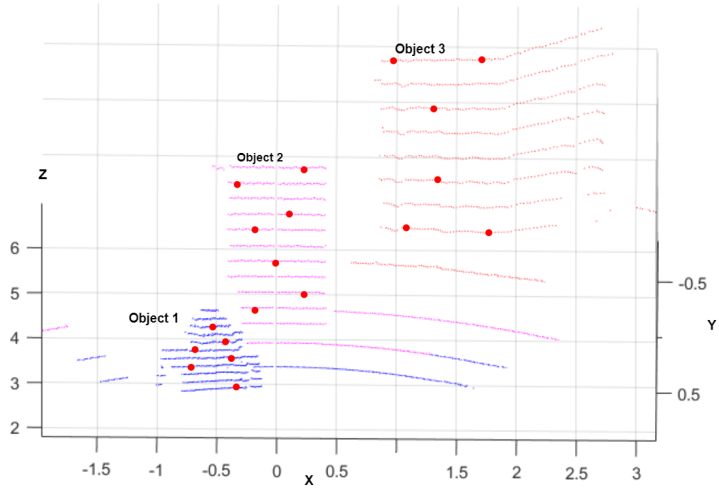


Figure 4.5: Points selected from the LiDAR point cloud.

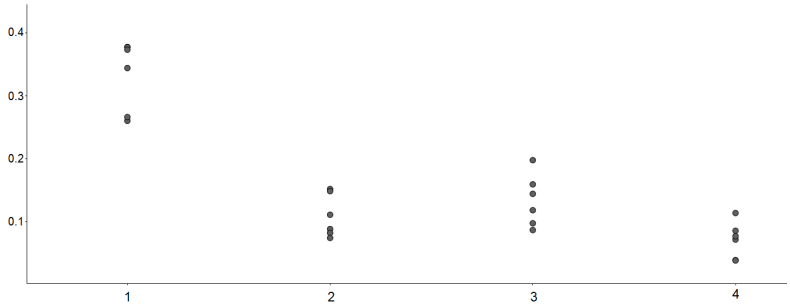


Figure 4.6: The error of the third object plotted

meters undoubtedly perform better than the other in the given scene.

4.2.1 Discussion

The results present that the closer the calibration scene is to the target object, the better depth estimates. Also, more accurate depth estimates are made for targets at shorter distances than the calibrating distances, rather than objects at farther distances. The result seems to confirm what is stated by Marita et al. (2006). The accuracy of the depth estimates is dependent on the distance the calibration is performed at. Thus, it appears that the calibration should be performed at the operating distance of the stereo system. Considering that the stereo camera overestimated the depth of all the objects, it may be beneficial to overestimate the operating distance while calibrating.

The sections gives an overview of considerations taken into account when calibrating

a far range stereo system. It seems that the best result is obtained when performing the calibration at the operating distance. However, the far-range extrinsic calibration should be conducted at various distances to discard or strengthen the hypothesis further.

4.3 Extrinsic stereo calibration method

The precision of the external stereo parameters influence the accuracy of the reconstructed world points (Fooladgar et al., 2013). Wrong estimations of the essential matrix can lead to false correlations or shortcomings of correlated points. This will influence the stereo reconstruction process. Since the objective is to use the stereo setup on an autonomous ferry, which requires far range object detection, the main focus is on estimating the external parameters.

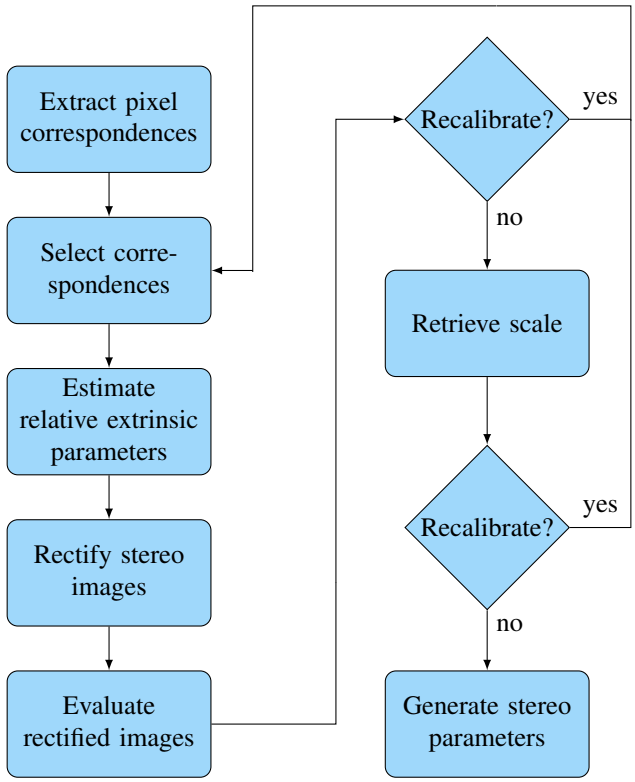


Figure 4.7: Calibration procedure

Calibrating a far range stereo setup using existing programs, such as MATLAB's Stereo Camera Calibration App and Kalibr, requires a predetermined planar calibration target. From the discussion in Section 4.2.1 it was suggested that performing the calibration at operating distance yields the best result. Following MATLABs recommendations, a operating distance at 50 meters requires a checkerboard of size 10×10 meters. It is next

to impossible to create a checkerboard of the preferred size. For this reason, this section proposes a method for calibrating a stereo camera using arbitrary scenes. The following sections presents the theoretical background of the proposed method. A summary of the steps is given in Figure 4.7.

4.3.1 Geometric error

The overall goal of the calibration is to minimize the geometric error. The geometric error, often referred to as the reprojection error, measures the difference between detected and reprojected points. If the world point $\mathbf{x}_W$ is known and detected at position $\mathbf{x}_L$ in the left camera view, the predicted pixel position $\mathbf{x}_R$ in the right camera view can be calculated based on the calibration parameters. Mathematically the extrinsic parameters for a calibrated stereo camera are defined in the essential matrix $\mathbf{E}$.

$$\mathbf{x}_L \mathbf{E} \mathbf{x}_R = 0 \tag{4.3}$$

The disparity between the predicted pixel position $\mathbf{x}_R$ and the actual true 2D world point $\mathbf{x}'_R$ in the right view is termed the geometric error, displayed in red in Figure 4.8. It is commonly measured in Euclidean distance and given in sub-pixel accuracy.

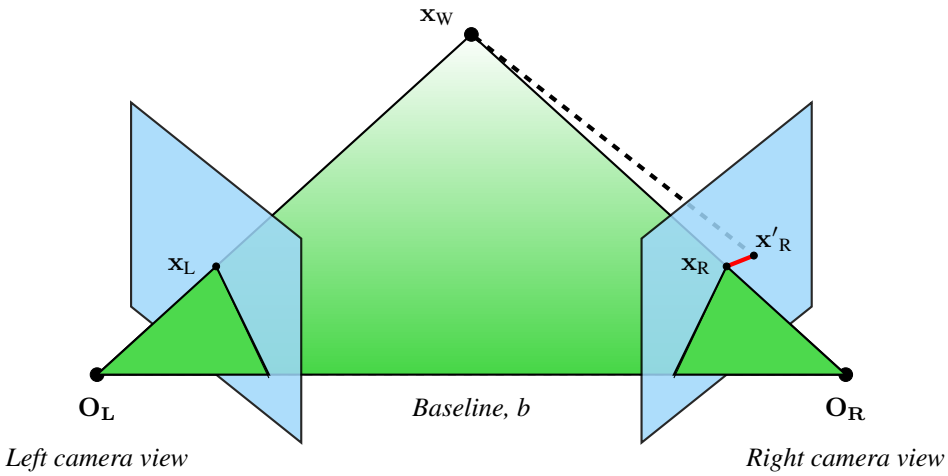


Figure 4.8: Epipolar geometry of two cameras

In an ideal situation, the geometric error is zero, which is desirable to achieve in the calibration. However, in a real stereo camera system, there is no one-to-one correspondence between the points in a world coordinate system and the ones in a camera plane (Fooladgar et al., 2013). Due to constraints in the image resolution and the quantization of world points, the determination of a 3D point is not unique. To deterministic determine the coefficients in the essential matrix two known constrains, (4.4) and (4.5), are used in addition to (4.3) (Kukelova et al., 2008). Hence, the calibration is a compromise between minimizing the geometric error and maintain the constraints.

$$\det(\mathbf{E}) = 0 \tag{4.4}$$

$$2\mathbf{E}\mathbf{E}^T - trace(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0 \tag{4.5}$$

4.3.2 Pixel correspondences

Each corresponding pixel pair defines a 3D world point. Pixel correspondences from a stereo setup usually are extracted by a featured based method, like the Shi-Tomasi Corner Detector. The Shi-Tomasi Corner Detector (Shi and Tomasi, 1994) is based entirely on the Harris corner detector (G. and Stephens, 1988), with an improvement in the last step.

Shi-Tomasi Corner Detector

Harris Corner Detector or Harris Operator is one of the earliest detectors proposed in 1988 by Harris and Stephens (G. and Stephens, 1988). It is an intensity-based mathematical operator finding features using corner descriptors in the extraction of features. A corner is the intersection of two edges, a point where the directions of the two edges change. Hence, a corner can be detected by high variations in the intensity gradient (in both directions). To detect corners and edges, both the Harris corner detection and Shi-Tomasi Corner Detector uses a combination of partial derivatives, Gaussian weighting function, and eigenvalues from the matrix representation of the following equation:

$$\mathrm{erf}(\Delta u, \Delta v) = \sum_{(u,v)\in W_m} \omega(u,v)[I_L(u+\Delta u, v+\Delta v) - I_R(u,v)]^2 \tag{4.6}$$

The equation (4.6) is indirectly searching for corners. By sweeping a window W_m at position (u, v) the variation of intensity is calculated for each pixel between the left image, $I_L(u, v)$, and the displacement pixel $I_R(u+\Delta u, v+\Delta v)$. The window function $\omega(u, v)$ is either a rectangular window or Gaussian window, giving weights to pixels underneath. The rectangle window function gives weight 1 to pixels inside the window and 0 otherwise. In practice, one can get a noisy response with a binary window function.

The objective is to determine the positions which maximize (4.6). By representing the Taylor expansion as a matrix, subsequently calculating the eigenvalues, weak and robust corners are detected with a threshold value. Deriving the mathematical expressions step

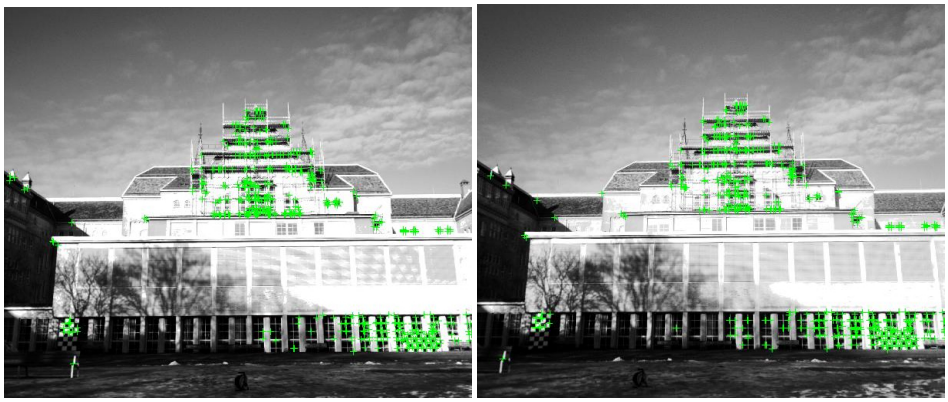
by step:

$$\begin{aligned}
\mathrm{erf}(\Delta u, \Delta v) &\approx \sum_{(u,v)\in W_m} \omega(u,v)[I_L(u,v) + \Delta u I_{Lu} + \Delta v I_{Lv} - I_R(u,v)]^2 \\
&= \sum_{(u,v)\in W_m} \omega(u,v)\left[\Delta u^2 {I_L}_u^2 + 2\Delta u \Delta v I_{Lu} I_{Lv} + \Delta v^2 I_{Lv}\right] \\
&= \begin{bmatrix}\Delta u & \Delta v\end{bmatrix} \left(\sum_{(u,v)\in W_m} \omega(u,v) \begin{bmatrix} {I_L}_u^2 & I_{Lu} I_{Lv} \\ I_{Lu} I_{Lv} & v^2 I_{Lv} \end{bmatrix} \right) \begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix} \\
&= \begin{bmatrix}\Delta u & \Delta v\end{bmatrix} M \begin{bmatrix}\Delta u \\ \Delta v\end{bmatrix}
\end{aligned}$$

For each window a score R is calculated for determining if a window is containing a corner or not;

$$R = min(\lambda_1, \lambda_2) \tag{4.7}$$

Based on (4.7) the algorithm decide whether a region is a corner, edge or flat. If R is greater than a predefined threshold, the region is marked as a corner-point. The maximum error of the corner detection process is ± 1 pixel (Sasiadek and Walker, 2019). Figure 4.9



(a) Left camera view **(b)** Right camera view

Figure 4.9: Corners extracted by Shi-Tomasi Corner Detector

shows an example of the resulting output of the Shi-Tomasi Corner detector. The algorithm marks the extracted features as green crosses.

4.3.3 Estimation of the relative extrinsic parameters

Estimating model parameters requires finding the best pixel correspondences and eliminate outliers. Given a set of correspondences, the most favorable result of the essential

matrix $\mathbf{E}$ minimizes the geometric error. Hence, it minimizes the sum of squares of distances, orthogonal to the variety from each point (Torr and Zisserman, 2000). Estimating the essential matrix often gives rise to nonlinear constraints between parameters, making optimization a difficult task. Choosing the best correspondences is usually done with estimators like Random Sample Consensus (RANSAC) or the modified version M-estimator SAmple Consensus (MSAC) algorithm. While RANSAC tends to give poor estimates when the threshold for inliers is set too high, MSAC is remedied with a new cost function. MSAC was published in 2000 and can robustly estimate the essential matrix from point correspondences, providing a 5-10% improvement compared to RANSAC (Torr and Zisserman, 2000).

Pixel pairs are chosen based on proximity and similarity; hence mismatches can occur. In theory, all the best matches obey the epipolar geometry, and MSAC aims to remove putative pixel pairs inconsistent with the epipolar geometry, i.e., outliers.

$$f(\mathbf{D},\mathbf{E}) = \prod_{i=1,\ldots,n} \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n e^{-\left(\sum_{j=L,R}(u_j{}'_i - u_{j_i})^2 + (v_j{}'_i - v_{j_i})^2\right)/(2\sigma'^2)} \tag{4.8}$$

$$-\sum_{i=1\ldots n} \log\left(f(\mathbf{x}_{(L,R)_i},\mathbf{E},\sigma)\right) = \sum_{i=1\ldots n}\sum_{j=L,R}\left((u_j{}'_i - u_{j_i})^2 + (v_j{}'_i - v_{j_i})^2\right) \tag{4.9}$$

$$e_i^2 = \sum_{j=L,R}(\hat{u_{j_i}} - u_{j_i})^2 + (\hat{v_{j_i}} - v_{j_i})^2 \tag{4.10}$$

For N number of times, a randomly chosen set S_m of pixel pairs are chosen to create the essential matrix $\mathbf{E}$. The support for each essential matrix, $\mathbf{E}$, is determined by the inliers in the set S_m.

The images are assumed to include Gaussian distributed noise with zero mean and a uniform standard deviation. The probability density function is given by (4.8), where n is the number of matches and $\mathbf{D}$ the set of matches. For all the corresponding pixels $\mathbf{x}_{(L,R)_i}$ the negative logarithmic likelihood function is (4.9). The true relationship of the stereo camera minimizes the likelihood function (4.9), and thus, the support for each pixel correspondence is determined based on finding the maximum likelihood estimation error (4.10). This is termed the MLE error e_i, and the estimate $\widehat{\mathbf{x}_{(L,R)_i}}$ of the true position $\mathbf{x}_{(L,R)'_i}$ is minimized for each point i as well as it must satisfy the constraint (4.3).

$$C = \sum_i \rho(e_i^2), \quad \rho(e_i^2) = \begin{cases} e^2, & \text{if } e^2 < T. \\ T^2, & \text{otherwise.} \end{cases} \tag{4.11}$$

In the end, the support for each essential matrix $\mathbf{E}$ is determined by the cost function (4.11). The cost is calculated by summing the MLE error for each inlier in the set S_m, i.e., correspondences with error an e_i below a given threshold T.

The MSAC algorithm for estimating the relative essential matrix can be comprised into the following steps:

1. Repeat n number of times

   (a) Select a random sample subset containing the minimum number of correspondences $S_m = \{\mathbf{x}_{(L,R)_i}\}$
   (b) Estimate the image relation, the essential matrix $\mathbf{E}$
   (c) Calculate the error e_i for each datum
   (d) Calculate the cost function C

2. Select the best solution of the samples, i.e., the one with the lowest C. Store the set of correspondences S_m that gave this solution.

3. Use the solution provided in the last step as the parameterization. Test all correspondences against the parameterization, and add correspondences that fit the constrained optimization.

Summarized, MSAC makes use of a simple redescending M-estimator (Huber, 1985) where inliers are scored on how well they fit the data. The cost function is shown in (4.11), where T is the threshold for considering inliers. The error term e_i is a Sampson distance. Sampson distance is a first-order approximation of a geometric distance, here the squared distance between a point and the epipolar line. MSAC is a nonlinear iterative optimization algorithm, and because the output is an initial estimate of the relation with a corresponding likelihood, the result can not always guarantee the optimal solution (Yu et al., 2018).

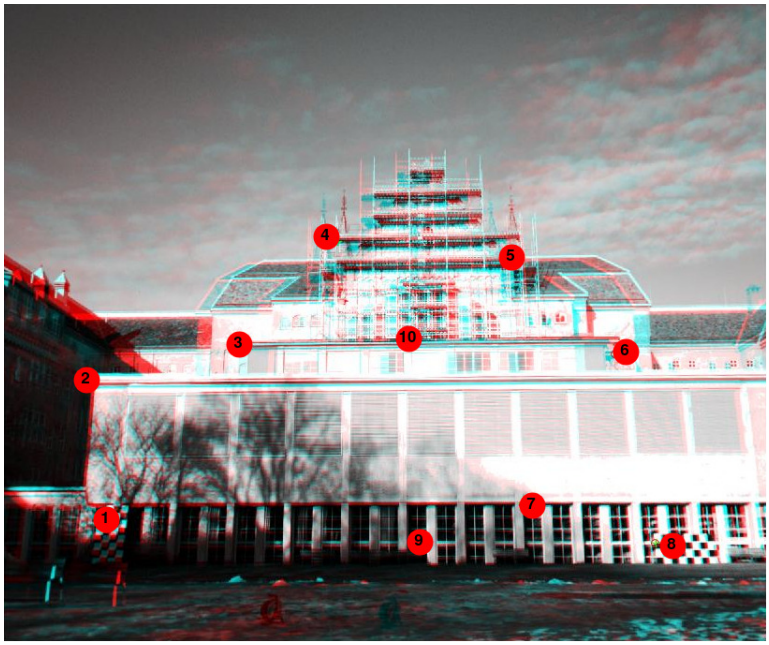


Figure 4.10: Example of calibration points

Due to the inbuilt randomness of the algorithm, it is necessary to select corresponding pixels beneficial for calibration. The estimation of the essential matrix may vary with the

same calibration scene and chosen pixel correspondences. The differences may be large. Thus, it is necessary to select the most suitable points of the ones distributed in Figure 4.9. (Yu et al., 2018). Figure 4.10 shows an example of preferred points sent as input to the MSAC algorithm. Some features may be rejected. The algorithm outputs the estimated relative essential matrix of the system, which may vary depending on which features are rejected and which are not. Comparing and evaluating the relative essential matrices is based on the obtained rectified images. Figure 4.11 shows an example of the resulting

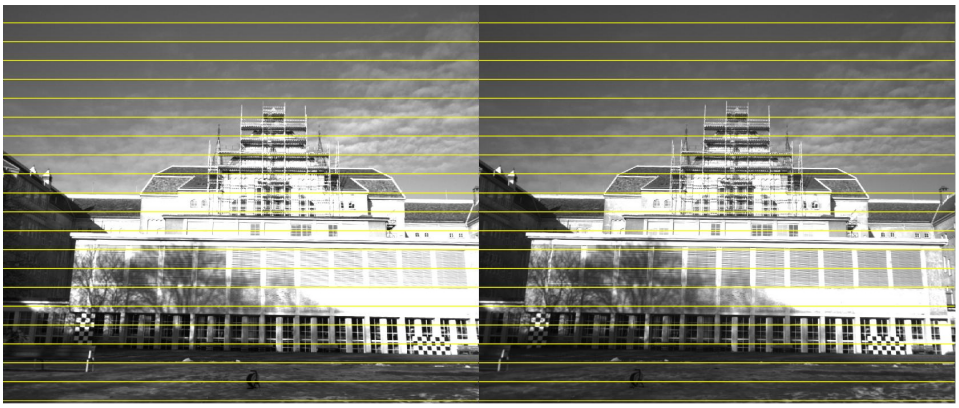


Figure 4.11: Example of rectified images

rectification of a scene. The rectification is obtained by running the MSAC algorithm with the points given in Figure 4.10 as input.

4.3.4 Absolute extrinsic parameters

From the estimated essential matrix a relative transformation between the two cameras can be extracted. The parameters are relative, i.e., not given in scale. The angles are absolute, but the translation lacks scale. The absolute extrinsic parameters link the position of the reconstructed 3D points to the real-world coordinate system. To reconstruct the 3D points, matching features in the two images must be found. The images is rectified using the rotation between the cameras, and extracted features using Shi-Tomasi Corner Detector from the left image can be tracked into to the right using the Kanade-Lucas-Tomasi feature tracker.

Kanade-Lucas-Tomasi feature tracker

The Kanade-Lucas-Tomasi (KLT) feature tracker is a method for tracking extracted features presented by Tomasi and Kanade in (1991). Images referring to the same scene at different viewpoints are strongly related to one another. Thus, it allows for tracking a feature extracted from an image into another image taken at a different angle or time. Consequently, an image sequence obeys the following property

$$I_R(u, v) = I_L(u - \Delta u, v - \Delta v) \tag{4.12}$$

When two cameras capture a scene from two different viewpoints, moving every point in the left image makes it possible to obtain the image captured to the right. The displacement $\mathbf{d} = (\Delta u, \Delta v)$ of a point $\mathbf{x}_\mathrm{R} = (u, v)$, represents the amount of motion between the two frames. Factors such as occlusion, illumination changes, and the disappearance of point make images violate the constraint (4.12). These factors can make the tracking difficult, but at surface markings and non-occluded areas, the property remains invariant.

When tracking a feature from one image to another, a window W_m of pixels is tracked and not a single pixel. Tracking a single pixel is a difficult task due to noise, changes in brightness, and it can be confused with adjacent pixels. Thus, using windows containing sufficient textures is preferable. By discarding windows where the appearance has changed too much, and by modeling the changes as an affine map and not just a single translation, it is possible to make sure that the correct window is tracked. The displacement vector, $\mathbf{d}$, is estimated for small windows. As the images are rectified and taken in the same distance of the target, any discrepancy not due to translation is considered as an error. The local image model is given as

$$I_R(\mathbf{x}_\mathrm{R}) = I_L(\mathbf{x}_\mathrm{R} - \mathbf{d}) + n(\mathbf{x}_\mathrm{R}),$$

where n is the noise. The displacement vector is chosen to minimize the residue error over a given window W_m. The residue error is given by

$$\mathrm{erf} = \int_{W_m} [I_L(\mathbf{x}_\mathrm{R} - \mathbf{d}) - I_R(\mathbf{x}_\mathrm{R})]^2 \omega d\mathbf{x}_\mathrm{R},$$

where, ω is a weighting function.

Approximating the intensity function using Taylor series and shortening it to the linear term yields the following expression.

$$I_L(\mathbf{x}_\mathrm{R} - \mathbf{d}) = I_L(\mathbf{x}_\mathrm{R}) - \mathbf{g} \cdot \mathbf{d}$$

The image gradient is given by $\mathbf{g} = (\frac{\partial I}{\partial u}, \frac{\partial I}{\partial v})$. Using the new representation of the intensity function the residue function can be rewritten to a quadratic function. Thus, the minimization can be done in closed form. After some rearranging, the following equation is obtained.

$$\begin{aligned} \mathbf{d} \int_{W_m} \mathbf{g}\mathbf{g}^\top \omega dA &= \int_{W_m} (I_L - I_R)\mathbf{g}\omega dA \\ G\mathbf{d} &= \mathbf{e} \end{aligned} \tag{4.13}$$

For every corresponding image pairs, the matrix G can be computed by estimating gradients and computing their second-order moments of one frame. The two dimensional vector $\mathbf{e}$ can be calculated from the difference between the two images along with the gradient. Thus, the displacement $\mathbf{d}$ is the solution of (4.13), and feature point can be tracked between frames.

Linear triangulation

After extracting corresponding features, their 3D position can be reconstructed using linear triangulation. Linear triangulation is the process of finding a 3D world point given the

pixel coordinates of the point in two different views (Hartley and Zisserman, 2004). The solution can be found by utilizing Direct Linear Transformation (DLT). Given a set of 2D correspondences, $\mathbf{x}_{\mathrm{L}i} \leftrightarrow \mathbf{x}_{\mathrm{R}i}$, the DLT algorithm determines the relation between the corresponding points.

When using a stereo camera, the corresponding points must satisfy the epipolar constraint presented in (4.3). The points are captured by two cameras which puts a geometric constraint on the points given by

$$\mathbf{x}_{\mathrm{L}} = \mathbf{P}_{\mathrm{L}}\mathbf{x}_{\mathrm{W}}, \quad \mathbf{x}_{\mathrm{R}} = \mathbf{P}_{\mathrm{R}}\mathbf{x}_{\mathrm{W}}$$

where $\mathbf{x}_{\mathrm{L}} = (u_{\mathrm{L}}, v_{\mathrm{L}}, w_{\mathrm{L}})$, $\mathbf{P}$ is the camera matrix and $\mathbf{x}_{\mathrm{W}}$ represents the 3D world coordinates. These equations can be combined in order to represent them on the linear form $\mathbf{A}\mathbf{x}_{\mathrm{W}} = \mathbf{0}$. Using a cross-product, $\mathbf{x}_{\mathrm{L}} \times (\mathbf{P}_{\mathrm{L}}\mathbf{x}_{\mathrm{W}}) = \mathbf{0}$, the homogeneous scale factor can be eliminated and three equations for each image point can be given. This gives the following

$$\begin{aligned} u_{\mathrm{L}}(\mathbf{p}_{\mathrm{L}}^{3\top}\mathbf{x}_{\mathrm{W}}) - (\mathbf{p}_{\mathrm{L}}^{1\top}) &= \mathbf{0} \\ v_{\mathrm{L}}(\mathbf{p}_{\mathrm{L}}^{3\top}\mathbf{x}_{\mathrm{W}}) - (\mathbf{p}_{\mathrm{L}}^{2\top}) &= \mathbf{0} \\ u_{\mathrm{L}}(\mathbf{p}_{\mathrm{L}}^{2\top}\mathbf{x}_{\mathrm{W}}) - v_{\mathrm{L}}(\mathbf{p}_{\mathrm{L}}^{1\top}) &= \mathbf{0} \end{aligned}$$

where $\mathbf{p}_{L}^{i\top}$ are the rows in the matrix $\mathbf{P}_{\mathrm{L}}$. Thus, an equation on the form $\mathbf{A}\mathbf{x}_{\mathrm{W}} = \mathbf{0}$ can be constructed by

$$\mathbf{A} = \begin{bmatrix} u_{\mathrm{L}}\mathbf{p}_{\mathrm{L}}^{3\top} - \mathbf{p}_{\mathrm{L}}^{1\top} \\ v_{\mathrm{L}}\mathbf{p}_{\mathrm{L}}^{3\top} - \mathbf{p}_{\mathrm{L}}^{2\top} \\ u_{\mathrm{R}}\mathbf{p}_{\mathrm{R}}^{3\top} - \mathbf{p}_{\mathrm{R}}^{1\top} \\ v_{\mathrm{R}}\mathbf{p}_{\mathrm{R}}^{3\top} - \mathbf{p}_{\mathrm{R}}^{2\top} \end{bmatrix} \tag{4.14}$$

Using the Shi-Tomasi Corner Detector and Kanade-Lucas-Tomasi feature tracker, more than four pixel correspondences are extracted. This results in the equations extracted from $\mathbf{A}\mathbf{x}_{\mathrm{W}} = \mathbf{0}$ being over-determined. Due to the discrete camera frame, noise will always be present in image coordinates. Hence, there is no exact solution to the system. An approximate solution is found by applying a cost function. Minimizing the norm $\|\mathbf{A}\mathbf{x}_{\mathrm{W}}\|$ subject to the constraint $\|\mathbf{x}_{\mathrm{W}}\| = 1$ the solution is the eigenvector with least eigenvalue of $\mathbf{A}^{\top}\mathbf{A}$.

From relative to scale

Performing linear triangulation on the corresponding feature points in the two images, allows for a point cloud reconstruction of the scene. Figure 4.12 shows the corresponding features found in a scene captured by the stereo camera. The features are extracted in the left frame by the Shi-Tomasi Corner Detector, and tracked into the right frame using the Kanade-Lucas-Tomasi feature tracker. The resulting point cloud is relative and not given in scale. To obtain the scale a measurement from the real-world is required.

Using the LiDAR as ground truth, it provides a real-world measurement of the scene. The calibration of the LiDAR-stereo camera in Section 3.2 provides the transformation necessary to transform the LiDAR into the stereo cameras reference frame. Thus, the LiDAR measures the actual depth of the scene. NDT presented in Section 3.2.1 calculates

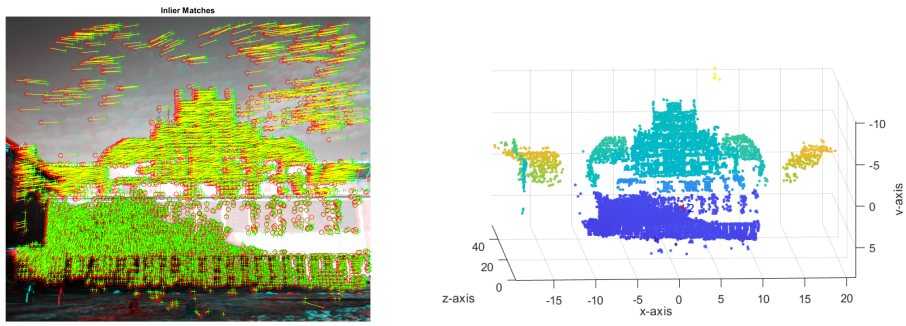


Figure 4.12: Matched features and corresponding point cloud without scale

the rigid body transformation between two point clouds. Finding the transformation between the relative point cloud and the real world LiDAR point cloud provides the actual depth of the points. Thus, the scale is retrieved by dividing the actual depth of a point with the relative one. Multiplying the relative translation with the scale factor provides the absolute stereo parameters. When the scale is retrieved, the rigid body transformation between the two cameras is estimated.

To evaluate the parameters an up to scale point cloud of the calibration scene is reconstructed using a disparity map algorithm. Semi-Global Matching from Section 2.4.2 is used due to its accuracy and tolerance against radiometric changes. The ground truth is used to evaluate if the obtained parameters should be rejected or not.

The calibration method utilized can be summarized in the following steps The steps

Algorithm 1 Extrinsic calibration

1: Undistort the image pair using the pre-computed intrinsic parameters
2: **while** no good parameters **do**
3: Extract pixel correspondences based on Shi-Thomas Corner Detector
4: Estimate the relative extrinsic parameters by the MSAC algorithm
5: Evaluate rectified images
6: **for** each promising calibration **do**
7: Create pixel correspondences by Shi-Thomas Corner Detector and Kanade-Lucas-Tomasi feature tracker
8: Create the stereo point cloud by Direct Linear Transformation
9: Find the actual depth of the scene using the LiDAR point cloud
10: Use NDT to transform the relative point cloud to the actual depth
11: Extract scale from the relative and actual depth
12: Estimate absolute extrinsic parameters
13: Choose the best calibration based on reprojection errors, rectified images and comparing the stereo point cloud against the ground truth obtained from the LiDAR

are implemented using functions made available by MATLAB's Computer Vision Toolbox.

Chapter 5

Calibration results

The far range calibration method described in Chapter 4 is tested on different distances. Considering the operating distance of the stereo camera, it appears that the calibration should be performed at 50 meters. However, to test if the accuracy of the depth estimates is dependent on the distance the calibration is performed at, the calibrations are performed at 10, 20, 30, 40, 50, 60, and 70 meters. To evaluate the resulting parameters, they are tested on scenes of various depths. The LiDAR serves as a ground truth.

5.1 Resulting parameters

(a) Scene 10m **(b)** Scene 20m **(c)** Scene 30m **(d)** Scene 40m

(e) Scene 50m **(f)** Scene 60m **(g)** Scene 70m

Figure 5.1: Calibration scene

The camera's field of view and the required depth of the calibration scene, imposes constraints to the calibration scene. The scene should contain strong edges, and cover most of the cameras field of view. Due to the chosen calibration distances, the calibration is done outside. This gives rise to challenging light conditions. A building including strong edges is selected as the calibration scene. To ensure feature points are present in the outer edges two checkerboards are placed in front of the target.

Transformation at distance	Rotation right camera [Euler angles, deg]			Translation right camera [mm]		
	X (pitch)	Y (yaw)	Z (roll)	X(baseline)	Y(height)	Z(depth)
10m	0.0857	-1.2060	0.0198	-1740.9420	-22.5655	-52.0053
20m	-0.0439	-1.7265	0.0261	-1859.7417	9.5043	-25.2390
30m	-0.0806	-1.6488	0.0147	-1826.3048	23.4460	-20.3536
40m	-0.0443	-1.6558	0.0237	-1876.7839	44.8674	49.8614
50m	0.0012	-1.6058	0.0124	-1789.4666	-7.7865	-40.9851
60m	0.0191	-1.4662	0.0238	-1685.2528	-11.8655	-29.6606
70m	-0.0656	-1.7437	0.0436	-2001.4234	101.9670	50.3563
Average	-0.0183	-1.5790	0.0235	-1825.7022	19.6525	-9.7180

Table 5.1: Stereo parameters

The chosen calibration scene at the selected distances can be seen in Figure 5.1, where the calibration distance is measured to the first wall of the building. The resulting extrinsic parameters are given in Table 5.1. Based one Marita et al. (2006) it is expected that the parameters calibrated at a certain distance give the most accurate depth estimate of that distance. For this reason, an average calibration is made of all the parameters.

Mean reprojection error on all calibrations	
Calibration distance	**Reprojection Error [pixel]**
10	0.3142
20	0.3091
30	0.1113
40	0.1378
50	0.1276
60	0.1026
70	0.1056

Table 5.2: Mean reprojection error

The mean reprojection error of each calibration is given in Table 5.2. The error is calculated based on all the corresponding points selected for the calibration. Because of the randomness of the calibration algorithm, not all points are used to estimate the essential matrix. Thus the extra points are used to ensure the mean reprojection error is still acceptable when it is calculated based on points equally distributed over the image plane. The reconstructed point clouds of each calibration scene are given in Appendix A. Each calibration at the distinct distances is evaluated based on reprojection errors, the

rectified images, and the constructed point cloud of the calibration scene compared to the LiDAR

5.1.1 Evaluation

At 10 meters, the calibration scene is challenging, see Figure 5.1a. There are not enough strong corners to get useful feature points evenly distributed in the scene. Extracting pixel correspondences is only possible around the checkerboard, which is too small of a region of the image frame. Thus, the parameters are expected to deviate some from the others. The same goes for the calibration at 60 and 70 meters. The two scenes contain too much of the sky and the ground, which both are impractical for extracting satisfactory corner points.

Based on the stereo setup described in Section 2.2.1 the parameters are expected to be in the range of

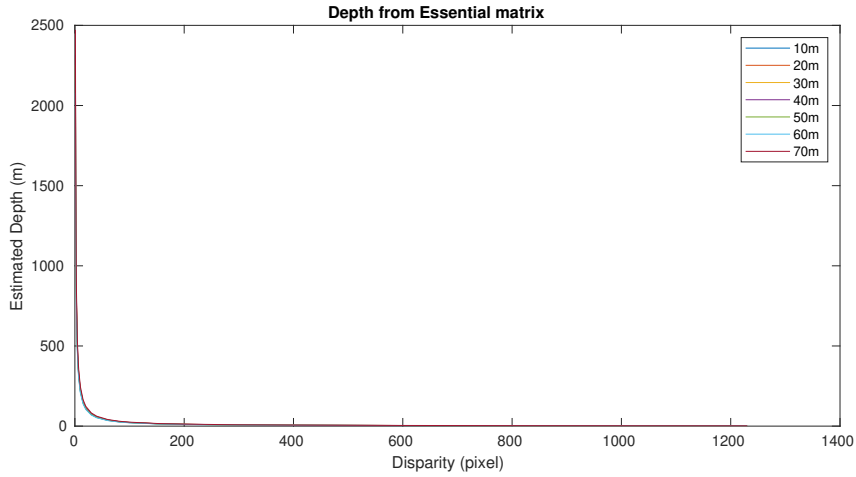
$$\mathbf{R} = \begin{bmatrix} x:0 & y:-2 & z:0 \end{bmatrix}$$
$$\mathbf{t} = \begin{bmatrix} -1800 & 0 & 0 \end{bmatrix}$$

Most of the calibrations are within the range of the expected result, with the 10, 40, and 70 meters calibration showing the most divergence. Considering (2.10) in Section 2.4.1 the baseline i.e., the translation in x-direction directly influences the estimated depth. With this in mind, the 40, 60, and 70 meters calibrations are expected to have a lower depth accuracy than the other parameters.

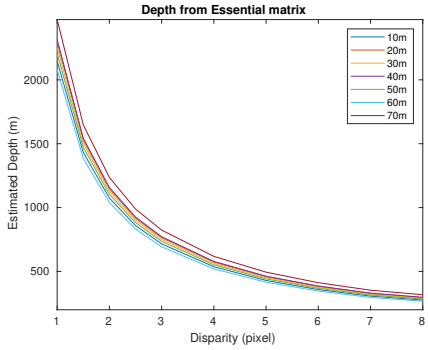
To inspect the extrinsic parameters, all the calibration parameters are plotted together in Figure 5.2. For each disparity on the x-axis, the corresponding depth is calculated. Figure 5.2a shows the entire graph, where 5.2b and 5.2c is zoomed in on the smallest and greatest disparities, respectively. The parameters estimates different depths of the same disparity. The biggest divergence is in the smallest disparities. The disparity is calculated based on the rectified images of a scene, which is dependent on the extrinsic parameters. In a real-world scenario, the same 3D-object corresponds to different disparities depending on the calibration parameters. I.e., the calibrations may give the same depth result even if the disparity is different, see the formula for depth (2.10) in Section 2.4.1.

The corresponding pixels used to estimate the essential matrix are extracted using the Shi-Tomasi Corner Detector, which has a maximum error of ± 1 pixel. In the plot, the gradient decreases with increasing disparity. Hence, at farther distances an error of one pixel corresponds to a higher inaccuracy. When calibrating at farther distances, an error of one pixel is far more critical than for shorter distances. Thus, making calibration on far range targets more sensitive to errors. The graph stabilizes at around 10 pixels, corresponding to 250 meters. To minimize inaccuracies of the depth, it is reasonable to set the minimum disparity around 10 pixels. Given by the plot, the minimum distance is 2 meters. This is due to the wide baseline giving rise to a blind spot in front of the cameras.

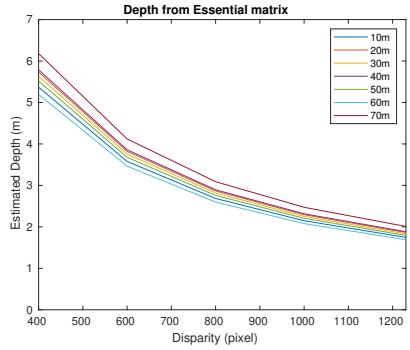
The mean reprojection errors in Table 5.2 are all reasonable low. The reprojection error and the value of the parameters alone do not say much of the quality of the calibrations. Of course, the reprojection error needs to be reasonably low, and the parameters within a given range, but it does not directly express how exact the depth estimates are. From a visual inspection of the point clouds in Appendix A, they all seem to estimate the depth of the calibration scene quite satisfyingly. The stereo parameters at 40 and 20 meters



(a) The graph in total



(b) Smaller disparities



(c) Greater disparities

Figure 5.2: Estimated depth from given disparity

may seem especially promising. But it is next to impossible to predict which parameters perform the best in general. All the parameters are expected to be optimal at the distance they are calibrated on, i.e., their performance must be evaluated at different distances.

5.2 Test scenes

To evaluate the calibrations performance, the stereo parameters are tested on scenes of different distances. The chosen scene is presented in Figure 5.3. The scene is captured approximately 30, 40, 50, 70, and 90 meters from the sensors. The LiDAR point cloud serves as the ground truth. A point cloud of each scene is reconstructed, using the Semi-Global Matching algorithm in Section 2.4.2. It is worth noticing that the scene at 40 meters is captured at a different time and under different weather conditions.

Figure 5.3: Test scene

In addition to a visual inspection, the translation in z-direction between the point cloud and ground truth is registered. To measure the error, the stereo point cloud is mapped onto the LiDAR point cloud. The transformation is registered by NDT. Noise is cut out of the point clouds to calculating the error transformation. The translations in the z-direction are used to evaluate the calibration accuracy in depth.

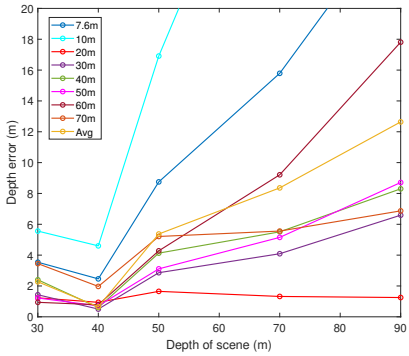
5.2.1 Results

The stereo and LiDAR point clouds of the scenes are plotted together and given in Appendix B. The error in depth of each calibration on the different test scenes is given in

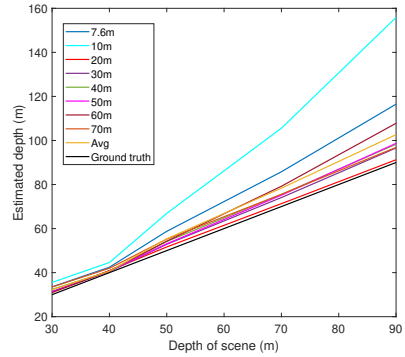
	Average depth error				
Calibration	**Scene 30m**	**Scene 40m**	**Scene 50m**	**Scene 70m**	**Scene 90m**
7.6 m	3.54 m	2.46 m	8.75 m	15.79 m	26.49 m
10 m	5.56 m	4.60 m	16.91 m	35.56 m	65.86 m
20 m	1.23 m	0.94 m	1.65 m	1.32 m	1.25 m
30 m	1.43 m	0.50 m	2.86 m	4.09 m	6.59 m
40 m	2.40 m	0.60 m	4.13 m	5.51 m	8.30 m
50 m	1.22 m	0.71 m	3.11 m	5.15 m	8.71 m
60 m	0.94 m	0.78 m	4.28 m	9.21 m	17.81 m
70 m	3.45 m	1.97 m	5.21 m	5.56 m	6.87 m
Avg	2.27 m	0.65 m	5.37 m	8.36 m	12.64 m

Table 5.3: Distance in meters between stereo point cloud and ground truth

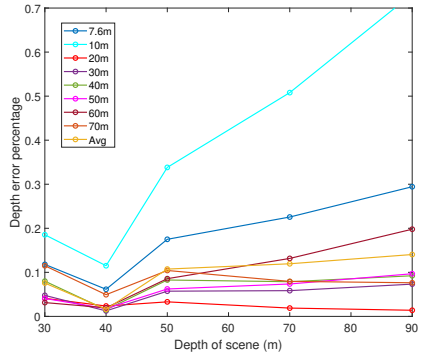
Table 5.3. The calibration obtained using MATLAB's stereo camera calibration app is included to illustrate the far range accuracy of a pre-existing calibration method. It should be noted that all the parameters overestimate the distance to the target. Figure 5.4 shows the resulting plot of the error in the depth of each calibration. Two apparent outliers are



(a) Depth error of scenes



(b) Estimated depth of scenes



(c) Depth error percentage

Figure 5.4: Error plots of test scenes

the calibrations obtained at 10 meters and by MATLAB's Calibration App. With a higher error than 25 percent, both calibrations are rejected.

The scene at 40 meters presents an irregularity in the plots. It was captured a month before the others, under different weather conditions at a different angle. In the other scenes, the stereo system was placed perpendicular to the target building. The different conditions may have caused a scene where disparities in the images are simpler to calculate. Also, due to the time difference, there is no guarantee that external forces has not influenced the stereo setup. For this reason, it is not representative together with the other scenes. To summarize, the 30, 50, 70 and 90 meters scenes are captured under identical conditions.

Figure 5.5 shows the depth error in percentage, where the 40 meters scene is excluded. The plot shows that the errors of the different calibrations are approximately linear. Nevertheless, the 20 meters calibration has a noticeable higher accuracy than the others.

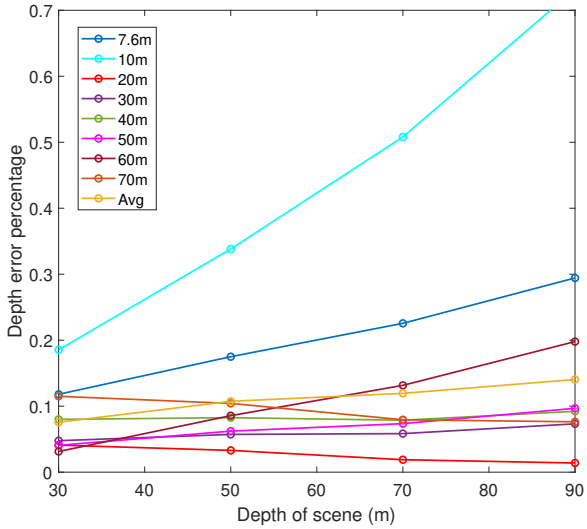


Figure 5.5: Depth error percentage

5.2.2 Evaluation

From Table 5.3 and Figure 5.5, the parameters diverge in their depth estimates. The calibration at 10 meters presents with the lowest accuracy, followed by MATLAB and the 60 meters parameters. Both 10 and 60 meters have a challenging calibration scene. The calibration at 7.6 meters seems to be over-fitted at its calibration distance. As for the rest of the calibrations, from Figure 5.4b an increasing gap between the estimated depth and the ground truth can be observed. However, the 20 meters calibration clearly outperforms the others.

Apart from the 20 meters parameters, the calibration error increase with increasing distance. This is expected as the error of one pixel have a greater impact as the disparities decreases. From Figure 5.4a, the calibrations perform better at shorter distances independent of the calibration distance. The calibrations performed at 20, 30, 40, and 70 meters gives no significant increase in error in percentage. The scenes at 20 and 30 meters are more favourable, as it is easier to extract points evenly distributed in the camera frame. Thus, it seems like the chosen scene influence the results more than the distance. The 20 m calibration is arguably the best. The calibration target covers the entire scene making it easier to extract features evenly distributed in the scene. Despite being calibrated at a shorter distance, the calibration only has an error of 1.4 percent at 90 meters, and an overall error of 2.6 percent.

$$\mathbf{R}_{20} = \begin{bmatrix} -0.0439 & -1.7265 & 0.0261 \end{bmatrix}, \quad \mathbf{t}_{20} = \begin{bmatrix} -1.860 & 0.0095 & -0.025 \end{bmatrix} \tag{5.1}$$

The variations of the results shows that calibrating a stereo camera is a sensitive procedure. To give a sense of how much changes in the parameters influence the accuracy,

small deviations in the parameters a simulated. This is done using the 20 m parameters (5.1) at the scene of 50 meters. The simulated errors is given as changes from the original translation from the ground truth in (5.2).

$$\mathbf{t} = \begin{bmatrix} -2.821 & -0.291 & -1.653 \end{bmatrix} \tag{5.2}$$

The translation (5.2) indicates a significant error in the x-direction. Inspecting the point cloud in Appendix B, it seems as the registration overestimates the translation in x due to some warpings on the left side of the stereo point cloud. The error in x-direction is included to give an estimate of how changes in parameters influence the 3D reconstruction.

Δ Translation [m]		ΔX_{err} [m]	ΔY_{err} [m]	ΔZ_{err} [m]
$\mathbf{t}_x$	-0.010	-0.007	-0.012	-0.286
	-0.005	-0.001	-0.003	-0.143
	-0.001	-0.003	-0.002	-0.026
	+0.001	0.003	0.001	-0.029
	+0.005	0.010	0.008	0.143
	+0.010	-0.009	0.017	0.283
$\mathbf{t}_y$	-0.010	0.189	-0.081	-0.016
	-0.005	0.094	-0.056	-0.008
	-0.001	-0.025	-0.010	-0.007
	+0.001	-0.028	0.015	-0.001
	+0.005	-0.082	0.040	0.005
	+0.010	-0.137	0.080	0.005
$\mathbf{t}_z$	-0.010	-0.352	-0.028	-0.008
	-0.005	-0.202	-0.009	-0.008
	-0.001	-0.044	0.002	-0.010
	+0.001	0.014	0.009	-0.005
	+0.005	0.156	0.014	0.002
	+0.010	0.303	0.032	0.004

Table 5.4: Difference in error in reconstructed points when translation between the cameras changes

Table 5.4 and 5.5 shows how the simulated error is affected when the stereo parameters changes. The rotation of the point cloud is a somewhat inconsistent, as NDT struggles to match the point clouds due the warping in the left side of the stereo point cloud. However, they are presented to give a sense of what parameters influence the 3D reconstruction the most. The parameters that influence 3D reconstruction the most are summarized in Table 5.6. Only the changes in rotation causing a consistent error is included. The table shows how sensitive the resulting accuracy is to wrong estimates of the stereo parameters. Calibration is a delicate procedure; small errors in the parameter estimate can lead to a severe loss of accuracy in 3D reconstruction. It also illustrates how sensitive the system is to external forces, a change of 0.1 degree in yaw yields a difference of 2.5 meters in depth. A robust setup is crucial. The findings show that for depth estimations, accuracy in rotation of y and t_x is crucial.

Δ Rotation [deg]		ΔX_{err}[m]	ΔY_{err} [m]	ΔZ_{err} [m]
X (Pitch)	+0.3	0.285	-1.170	0.113
	+0.2	0.455	-0.734	0.038
	+0.1	-0.067	-0.185	-0.009
	-0.1	0.066	-0.384	-0.087
	-0.2	-0.452	-1.090	-0.039
	-0.3	-0.629	-1,173	-0.045
Y (Yaw)	+0.3	-1.241	-0.595	-9.730
	+0.2	0.068	-0.194	-6.188
	+0.1	-3.235	-1.931	-2.422
	-0.1	-0.000	0.136	2.611
	-0.2	-0.005	0.156	5.000
	-0.3	-0.141	0.152	7.185
Z (Roll)	+0.3	3.224	0.535	-0.088
	+0.2	3.228	0.606	-0.132
	+0.1	1.855	0.359	-0.059
	-0.1	-1.093	-0.222	-0.020
	-0.2	-2.495	-0.242	-0.095
	-0.3	-3.242	-0.102	-0.181

Table 5.5: Difference in error in reconstructed points when orientation between the cameras changes

Extrinsic	**Error of the reconstructed 3D coordinate**		
Parameter Error	**X (lateral offset)**	**Y (height)**	**Z (depth)**
$\mathbf{t}_x$	small error	small error	**critical**
$\mathbf{t}_y$	**error**	small error	≈ 0
$\mathbf{t}_z$	**error**	small error	≈ 0
X (Pitch)	-	-	≈ 0
Y (Yaw)	-	-	**critical**
Z (Roll)	**critical**	small error	≈ 0

Table 5.6: Extrinsic parameters errors' effect on 3D reconstruction

5.3 Discussion

Calibration is a sensitive procedure. Overall, the depth errors were higher than expected. Most parameters has an error that stabilizes around 10 percent when the distance is increasing. However, there are some significant outliers. The accuracy of 3D reconstruction was shown to be sensitive to wrong estimations of rotation of y and t_x.

The calibration procedure is based on a set of assumptions and approximations. First the Shi-Tomasi corner detector has a maximum error of ± 1 pixels. An error of one pixel has a high impact of the accuracy as the distance increases. The extraction of feature points at distances as high as 70 meters makes it challenging to accurately estimate the parameters. In addition, the feature points used are manually extracted. Thus, the result is reliant on how careful the user is in selecting the points. A very careful selection of the used

calibration images and a different placing of the calibration board may also contribute to keep this error smaller. For the extraction of scale, no normalization was performed on the parameters used in the linear triangulation procedure. Due to the numerical estimation in DLT, the entries of **A** (4.14) should preserve to have similar magnitude. Thus, in retrospect the results of the triangulation may benefit from a normalization of the camera matrix. As the intrinsic parameters obtained from the MATLAB calibration are given in millimeters, they are of a higher magnitude than the other parameters; the homogeneous coordinate w and the relative extrinsic estimated matrix given in the range of 0-1.

The calibration at 50 and 60 meters was expected to give the best result for the test scene at the operating distance of the stereo camera, i.e. 50 meters. However, the highest accuracy on the operating distance was obtained from the 20 and 30 meters calibration. Overall, of all the different parameters the ones obtained at 20 and 30 meters gave the highest accuracy. The 20 meters calibration resulted in an average error of only 2.6 percent. Which is noticeable better compared to Wang et al., (2019), which achieved an accuracy of 74% at 30 meters putting their approach on par with LiDAR. The calibration scenes at 20 and 30 meters has the most favorable distribution of feature points for calibration. Thus, the results indicates that using an appropriate calibration scene is of higher importance than the depth of the scene. A scene covering the entire FOV, makes it possible to extract corresponding pixels distributed in the whole frame.

The errors are expected to be higher if testing in the operational environment. In a marine environment, there is less strong corners to match and reflection from the water makes the stereo matching burdensome. Another reason is that the test scenes used, may be too similar to the calibration scene. All the parameters should therefore have been tested in a marine environment to see the actual performance in the operational environment. Due to COVID-19 it was not possible to access the equipment necessary to acquire a more suitable test scene.

From the change of accuracy between the 40 meter test scene and the others scenes, there may have been some external factors influencing the stereo setup. The 40 meter scene was captured in early February, and the other test scenes were captured in late April. In the time between, the physical stereo system has been subjected to wear and tear due to transportation. The camera had to be moved around more than preferable. This may have caused changes to the configurations, including the angling of the cameras and the manually set exposure and zoom. Changes like this will directly influence the uncertainty of the depth estimates.

Part II

Application in marine environment

Chapter 6

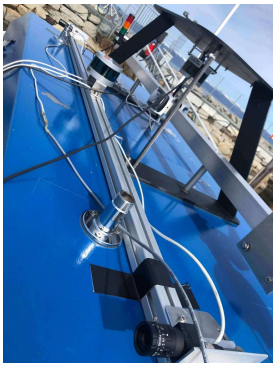
System overview

The intended use of the stereo camera is visual vessel detection for the autonomous ferry milliAmpere. This chapter introduces a stereo system for object detection in a marine environment. Two clustering techniques are implemented; hierarchical clustering using Euclidean distance and a convolution neural network (CNN). The network is used for classification, and its bounding boxes is combined with the disparity map to extract 3D positions of objects. The Euclidean clustering processes the raw point cloud reconstructed from the disparity map. This chapter gives an overview of the complete system while the following chapter goes into detail of the implementation of the techniques. Furthermore, the stereo system is combined with the system running on milliAmpere to output the detected objects in a common world frame. The system is implemented for testing the stereo cameras performance in milliAmperes operating environment.

Figure 6.1: Example of scene captured by the stereo camera

6.1 The operating environment

The performance of the system is evaluated by testing outside the port of Brattøra in Trondheim. The intention is to mount the real physical system on milliAmpere. Due to COVID-19 this was not possible. Instead, the stereo camera was temporarily placed on the ferry using duck tape, see Figure 6.2a. The goal of the system is to detect and estimate the position of boats. For this reason, a Merry Fisher 733 with a length of 7.5 meters was used as the target, see Figure 6.2b. Data is recorded where the target is the main focus in the scene. The scenery was set to the fjord with the coast line and the island Munkholmen present in the background.



(a) The stereo camera set up on milliAmpere

(b) Target boat

Figure 6.2: Stere camera mounted on milliAmpere, and the target boat

6.2 Processing pipeline

This section present the processing pipeline of the system. Figure 6.3 illustrates a simpli-

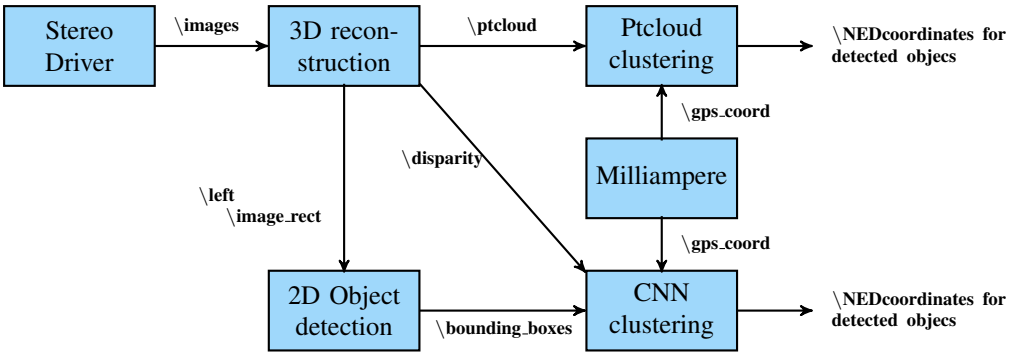


Figure 6.3: System overview

fied version of the overall stereo system. In the following sections the software, the most vital processes and the message distributions are elucidated. The code is stored using the version-control system GitHub. The complementary Readme file is given in Appendix C.

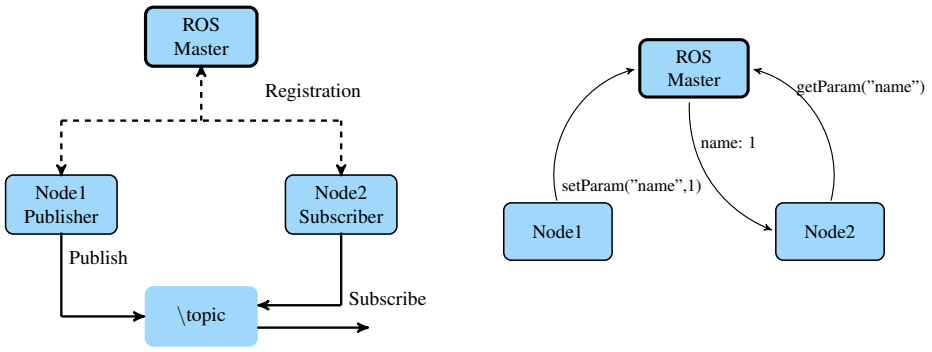
6.2.1 Software

The existing control system on milliAmpere runs the Robot Operating System (ROS)-Kinetic, making Ubuntu 16.04 LTS and ROS a natural choice for the stereo implementation and data acquisition. For processing images and point clouds the libraries Open Source Computer Vision (OpenCV) and Point Cloud Library (PCL) is used. The CUDA toolkit is utilized to speed up the process

Robot Operating System - ROS

The sophisticated robotic middleware used in this thesis is ROS, an open-source framework with collections of tools and libraries. It is defined as a meta-operating system, including low-level device control, hardware abstractions, package management, and message-passing between processes. One of ROS's philosophies is distribution, programs can be run on multiple computers and communicate over the network, making it a suitable choice for peer-to-peer network communication. ROS communication infrastructure provides a layer above the host operating system and is therefore dependent on an underlying operating system (Quigley et al., 2009). ROS Kinetic is primarily targeted at Ubuntu 16.04 (Xenial) release, and thus the chosen operating system for the hosting computer.

ROS is a distributed framework of nodes, i.e., processes, which enable easy communication between multiple machines. Hence, a stereo system can run on a separate machine, talking with the Master (milliAmpere) when necessary. The executables are therefore individually designed and loosely coupled at runtime. The communication is independent of programming language, advantageous as milliAmpere is written in Python, and the stereo system is primarily written in C++.



(a) ROS Topics **(b)** ROS Parameter Server

Figure 6.4: ROS architecture

ROS follows a concurrent computing paradigm, asynchronous communication, where

the main concepts are nodes, messages, services, Master, Parameter Server, and bags. The nodes symbolize processes performing computations, and message passing between nodes is published and subscribed via topics. The messages are routed via topics, which is a namespace specifying the content of a message. It implements a concurrent computing paradigm, a many-to-many relationship, where a node, in general, is not aware of whom it is communicating with. ROS also includes a one-to-one connection, services, where a node sends a message and waits for a response. This is synchronous communication, opposite of using topics.

The ROS Master manages the communication between nodes. It also provides name registration, registering every node at startup. Thus the Master can reanimate nodes. Inside the ROS master, the ROS Parameter Server is running. The server is a multi-variate dictionary that is accessible via network APIs. Best performance using static data, thus only used for tuning parameters in the nodes. Finally, ROS provides the package Rosbag. A rosbag records and plays published messages, making it possible to store data from experiments. With the package, the output of a sensor can be simulated without sensors being present. Tuning of parameters and plotting results therefore utilized bagfiles subsequent of the final-experiment.

In the flow chart, in Figure 6.3, each box represents a ROS package, which each contains nodes, nodelets, and topics for a given purpose.

OpenCV

Open Source Computer Vision (OpenCV) is a software library for computer vision and machine learning. The library has an interface for C++, and provides more than 2500 algorithms for processing images. Implementations of both classic and state of the art computer vision and machine learning algorithms are made available by the library. Throughout the system, the nodes include the library for reading and processing images. Especially, creation of the disparity map inherits from their stereo matching objects.

Point Cloud Library

Point Cloud Library (PCL) (Rusu and Cousins, 2011) is an open-source library providing tools for processing point clouds and 3D geometry. The library implements algorithms for three-dimensional computer vision, and it contains functions for among others filtering, feature extraction, and segmentation. It is written in C++. In the stereo system, the clustering package heavily depends on this library.

CUDA

Compute Unified Device Architecture, CUDA[1] is a parallel computing platform and programming model. The NVIDIA CUDA Toolkit lets the developer create high-performance applications by use of GPU-accelerated libraries. The GPU-acceleration is used for high-performance computing and optimized for parallel computing and multi-threading. The

[0] https://opencv.org/
[1] https://developer.nvidia.com/cuda-zone

convolutional neural network is utilizing the toolkit, as well as the stereo matching algorithm is to be accelerated using CUDA.

6.2.2 Stereo driver

The stereo package acquisition images and processes the raw data into a digitally encoded representation of the scene. The driver `spinnaker_sdk_camera_driver` is provided in ROS and is downloaded from (robotics). The cameras are supported by Spin-

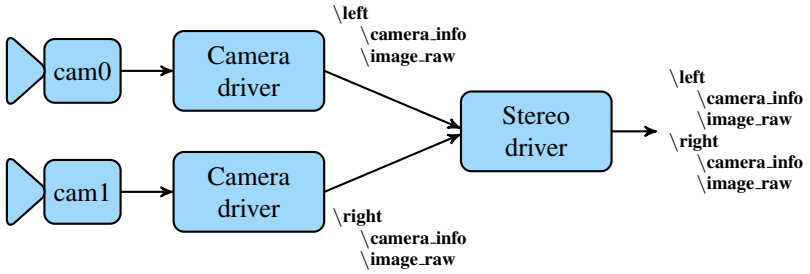


Figure 6.5: Overview of the stereo driver

naker SDK, an API built for machine vision developers. Each camera driver runs the camera using the Spinnaker SDK. The camera driver collects the data sent through the Gigabit Ethernet port and publishes the raw image data on the topic \image_raw. For two cameras to work as a stereo camera, the captured images need to be synchronized. This is achieved by using a master-slave setup. The master camera and each of the camera IDs need to be specified in the code. The master camera is software triggered, and it externally triggers the slave through the GPIO connector.

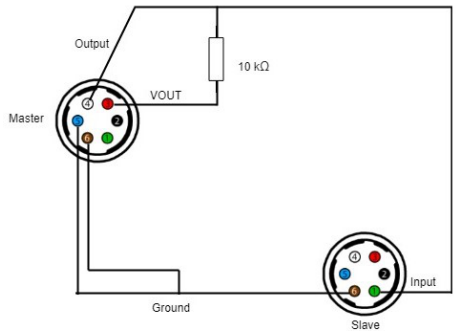


Figure 6.6: GPIO connections between master and slave

Figure 6.6 shows the wire soldering between the GPIO pins of the master- and slave. The GPIO pins are configured using the demo program SpinView made available with the

Spinnaker SDK. The camera driver labels each image with a frame ID and a timestamp according to ROS's built-in clock `ROS::Time`. The timestamp and camera ID are passed to the stereo driver together with the calibration parameters on the topic \camera_info. In the stereo driver node, two images are combined into a stereo pair. By synchronizing the timestamps with a new identical timestamp, the stereo driver passes the images on for further processing.

6.2.3 3D reconstruction

While the stereo driver mainly acquisition images and set the stereo time stamp, the 3D_reconstruction package creates three-dimensional models from the image pair. The package directly utilizes the theory in Chapter 2. The node Rectify is continuously run-

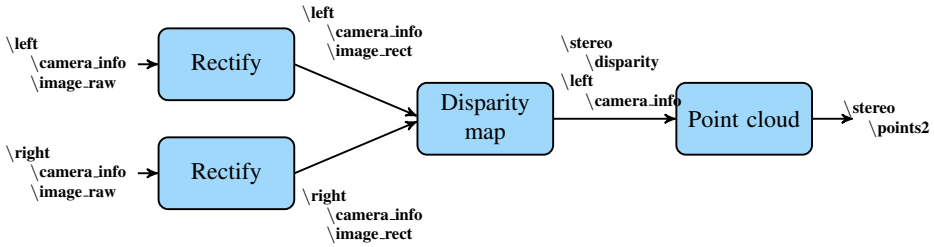


Figure 6.7: Overview of 3D Reconstruction

ning, concurrently subscribing to the topic image_raw from the left and right camera, respectively. The node uses the metadata assigned on the topic camera_info to remove distortion and for rectifying the co-calibrated cameras. Both the processes continuously publish rectified images, and the disparity nodelet subscribes to the topics. The disparity nodelet explicitly uses the timestamps from the stereo driver to correctly synchronize the image pairs. The stereo matching algorithm produces a depth map that further is mapped to a point cloud. The nodelet Point Cloud combines the disparity map with the calibration parameters to produce the point cloud.

6.2.4 Point cloud clustering

The package ptcloud clustering contains only one node, namely the PCL obstacle detector. With the name-twin class, it filters and clusters the raw point cloud published from the 3D_reconstruction package. For the ease of implementation, recording bagfiles, and testing, it is included in a separate package. However, the node can advantageously be included in 3D reconstruction as a nodelet (inherit from the nodelet class). A nodelet will minimize memory usage, running multiple algorithms in the same process with zero-copy transport between algorithms.

6.2.5 2D Object detection

A convolutional neural network recognizes and locates objects on the image plane. The node receives the left rectified image, and processes it to locates the 2D position of objects. The objects position in the image, the network's confidence, and the type of object is published for acquiring the world position. The package utilizes CUDA for speeding up the detection process.

6.2.6 CNN clustering

CNN-Clustering clusters the disparity map using the output from the 2D object detection package. The disparity map is created from the left view of the stereo camera, which is the same image fed to the convolutional neural network. The bounding box position in the image extracts the area of interest in the disparity map. Message filter in ROS ensures time synchronization between the disparity map and the bounding box predictions. By reconstruction, the world coordinates are extracted and published together with the prediction of the object type. The package publishes a message for each disparity map containing a list of objects.

6.3 Communication with milliAmpere

The system in use on milliAmpere can be compiled in a graph, giving an overview of the code. The system overview is presented in Appendix F. The tree consists of nodes, topics, hardware, python modules, services, and namespaces used, and the communication is illustrated with edges. This is the code currently running on the ferry, where for this project the navigation data is the most pertinent. Due to the processing power, and memory usage, the stereo system is running on a separate computer communicating with the master on milliAmpere. The stereo system is connected through a Ethernet cable, for stable communication through topics.

6.3.1 Common world frame

The navigation data provided by milliAmpere is used to determine the world position of objects detected by the stereo camera. The system subscribes to the navigation data of the ferry, which provides the data needed to transform the detected object into the set world frame. This places the vessel and the detected object in the same reference frame. The navigation data from milliAmpere and the detected objects by the stereo camera are synchronized by their ROS-timestamps. Subsequently, the clustering packages transforms the detected objects position into a North-East-Down (NED) reference frame. The stereo camera thus places objects in the local reference frame used by milliAmpere. The data flow is illustrated in Figure 6.3.

As mentioned, the stereo camera was temporary placed on the ferry for testing purposes, making a calibrating of the stereo camera and the BODY frame unachievable. Instead, the translation was measured by hand to the GPS sensors position on the ferry. The measured translation is $\mathbf{t}_c^{GPS} = [-1.86 \quad 0 \quad 0.29]^\top$, given in meters. The translation of

the GPS to BODY frame is already known; $\mathbf{t}^b_{GPS} = [0 \quad -0.975 \quad 2.33]$. Due to the intended use of the system, meter precision of the stereo camera measurement is sufficient. The error due to hand measurement can be thus neglected. The axes of the coordinate frame of the stereo camera is defined differently, but is approximately aligned. Thus the rotation of the stereo camera into the ferrys BODY frame is set to.

$$\mathbf{R}^b_c = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The transformation of the stereo cameras measurements is given by the matrices above, and can further be rotated into the NED frame. This is done by the rotation $\mathbf{R}^n_b$. The rotation is given by

$$\begin{bmatrix} \cos\psi cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\theta + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}$$

where ϕ, θ and ψ is the roll, pitch, and yaw of the vessel, respectively (Fossen, 2011). As testing on milliAmpere is performed under calm sea, three DOF is assumed, i.e., pitch and roll is negligible. The rotation is thus simplified to

$$\mathbf{R}^n_b = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where ψ is the heading. The heading is provided by the navigation system on milliAmpere. The detected objects position is know given in NED coordinates in the ferrys BODY frame.

To give the detected objects in a world frame, the geodetic coordinates of milliAmpere are transformed into the NED frame set in the operational area. The world position of milliAmpere is given in geodetic coordinates using latitude as the World Geodetic System 84 reference ellipsoid (WGS-84). The coordinates is thus given in latitude μ, longitude l and elevation h. Before transforming to NED coordinates a transformation to Earth-Centered, Earth-Fixed (ECEF) coordinates is performed by

$$\begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} = \begin{bmatrix} (N+h)\cos\mu\cos l \\ (N+h)\cos\mu\sin l \\ (\frac{r_p^2}{r_e^2}N + h)\sin\mu \end{bmatrix}$$

where r_e and r_p is the equatorial and polar axis radius of the ellipsoid (Fossen, 2011). Their values are given by WGS-84. N is the radius of the curvature in the prime vertical of the ellipsoid and can be calculated by

$$N = \frac{r_e^2}{\sqrt{r_e^2\cos^2\mu + r_e^2\sin^2\mu}}$$

The NED frame can now be found by

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} -\sin\mu_0 \cos l_0 & -\sin\mu_0 \sin l_0 & \cos\mu_0 \\ -\sin l_0 & \cos l_0 & 0 \\ -\cos\mu_0 \cos l_0 & -\cos\mu_0 \sin l_0 & -\sin\mu_0 \end{bmatrix} \begin{bmatrix} x_{e,b} - x_{e,0} \\ y_{e,b} - y_{e,0} \\ z_{e,b} - z_{e,0} \end{bmatrix}$$

where $x_{e,0}$, $y_{e,0}$ and $z_{e,0}$ and l_0 and μ_0 are the ECEF and geodetic coordinates of the origin of the NED frame. $x_{e,b}$, $y_{e,b}$ and $z_{e,b}$ are the ECEF coordinates of the origin of the BODY frame. Points in the body frame can now be transformed to NED frame by $\mathbf{R}_b^n$ and $\mathbf{t}_b^n$. The translation $\mathbf{t}_b^n$ is the world position of milliAmpere in the NED frame.

Chapter 7

Object detection

Object detection and localization are essential parts of a tracking system. The classification of objects is also helpful, especially for a ship that needs to follow legislation at sea. For a stereo camera, this means that the captured images need to be post-processed to obtain world coordinates of objects in the scene. In this chapter, the methods and algorithms used in the proposed system are presented. As well as an overview of some uncertainties present in the stereo system.

7.1 Uncertainty in the stereo system

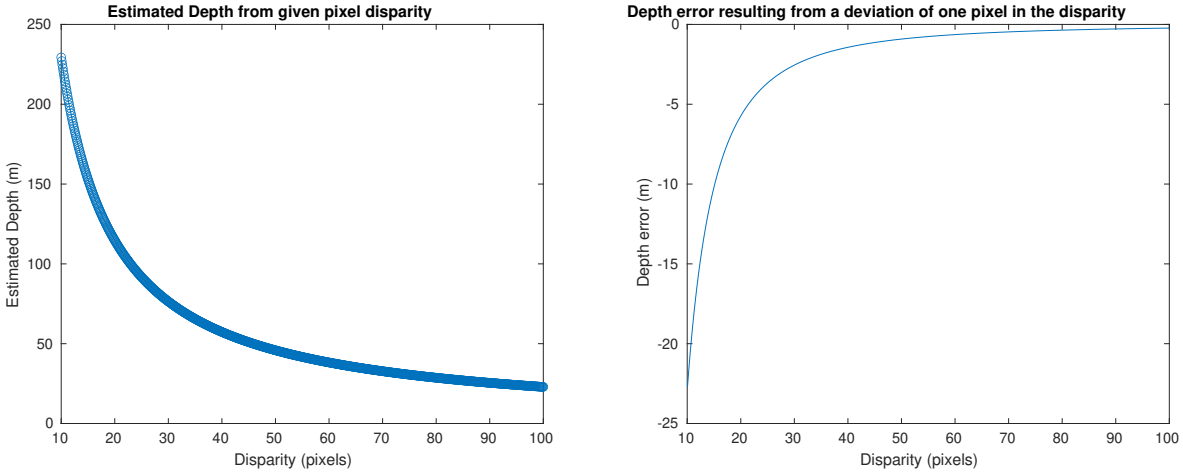
For the system to perform a correct localization of objects, it has to rely on an accurate calibration. The most accurate calibration in Chapter 5 is the 20meters calibration. The parameters performed overall with the highest accuracy. However, as they were determined on solid ground, with large scale buildings, a re-estimate of the accuracy for use at sea is desirable. By analyzing the obtained parameters, an estimate of the expected accuracy is given based on the operating environment. The extrinsic parameters of the calibration at 20 meters are given in equation (7.1).

$$\mathbf{R}_{20} = \begin{bmatrix} -0.0439 & -1.7265 & 0.0261 \end{bmatrix}, \quad \mathbf{t}_{20} = \begin{bmatrix} -1.860 & 0.0095 & -0.025 \end{bmatrix} \tag{7.1}$$

7.1.1 Reprojection error

The obtained calibration parameters (7.1) had an average reprojection error of 0.3091 pixels in the calibration process. This is the mean accuracy; hence the error can be higher at some parts of the image. The erroneousness means the 3D reconstruction will be slightly biased, but typically one assumes that these small errors do not cause problems for stereo matching (Hirschmüller and Gehrig, 2009). However, most literature uses and evaluate stereo accuracy on shorter distances. Figure 7.1a illustrate the relationship of the disparity and its corresponding depth estimate for the 20 meter-graph, extracted from Figure 5.2.

The depth error one expects from an error of one pixel in the disparity is shown in figure 7.1b. Table 7.1 summarizes the depth error on different distances from a given pixel-error. The pixel error is determined by computing in the slope of one pixel.



(a) Depth on given pixel disparity

(b) Depth uncertainty on given pixel disparity

Figure 7.1: Depth estimate for a given disparity and the error estimate for a one pixel deviation in the disparity (Baseline = 1.860m)

Real world-depth	Disparity	Depth error [1 pixel error]
10 m	229.3	± 0.044m
20 m	114.7	± 0.175m
30 m	76,5	± 0.392m
40 m	57.3	± 0.699m
50 m	45.9	± 1.091m
60 m	38.2	± 1.572m
70 m	32.8	± 2.138m
80 m	28.7	± 2.794m
90 m	25.5	± 3.541m
100 m	22.9	± 4.373m
110 m	20.9	± 5.275m
120 m	19.1	± 6.287m
130 m	17.7	± 7.362m
140 m	16.4	± 8.527m
150 m	15.3	± 9.797m

Table 7.1: Expected error based on pixels and baseline

The shown depth error indicates the expected reprojection error under the assumption that the disparity error is ± 1 pixel. Based on the challenging operating environment some higher inaccuracies can be expected. There are challenges in the lighting conditions, and usually some parts of boats are in general texture-less. This makes the stereo matching burdensome. Combining the mean reprojection error with sub-pixel accuracy in the stereo matching, expecting an error of around one pixel appears reasonable for us.

7.1.2 Stereo setup

In addition to the theoretical reprojection error, the stereo setup directly influences the accuracy of the reconstructed 3D points. An evaluation of what to expect in the operating environment is conducted by looking at the thesis's real physical stereo camera. For the discussed setup, the baseline measures 1.860 meters, and the symmetric vergence angles measures $\psi = 90 - 1.7265/2$. With these parameters the determined fixation point is around 61 meters (7.3).

$$Z = \frac{b_x}{2} \tan \psi \tag{7.2}$$

$$61.0145 = \frac{1.860}{2} \tan(90 - \frac{1.7265}{2}) \tag{7.3}$$

The fixation point diverges 10 meters from the original or intended fixation point on 50 meters.

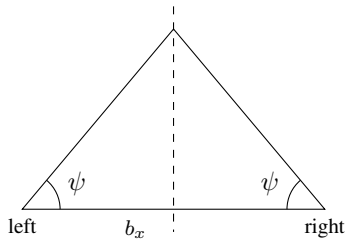


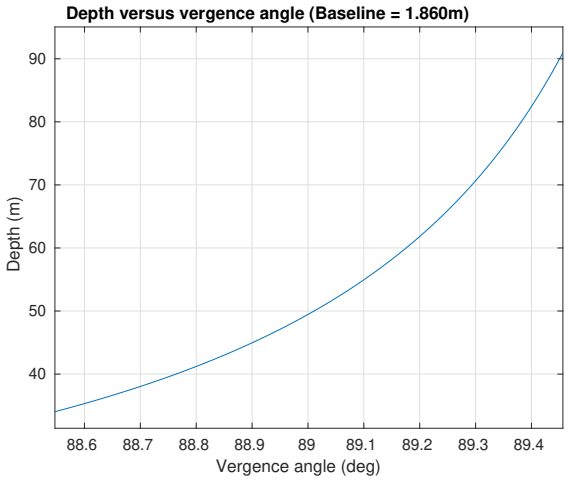
Figure 7.2: Symmetric vergence angle

The best possible accuracy for vergence stereo is obtained with symmetric camera vergence angles (Christensen et al., 1993). The thesis' stereo system utilizes symmetric vergence angles; however, the angles are of significant size. The accuracy of the rotation matrix, rotation around the y-axis, highly depends on the size of the angles. Consider the partial derivative of equation (7.2):

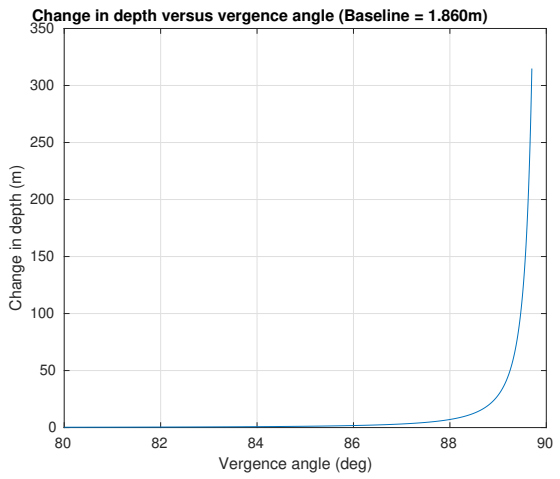
$$\frac{\partial Z}{\partial \psi} = \frac{b_x}{2}(1 + \tan^2 \psi) \tag{7.4}$$

As a result of the tangent-function, the vergence angle sensitivity approaches infinity near 90 degrees (parallel cameras). Hence, smaller angles are preferred considering the accuracy of the resulting parameters from a calibration. In theory, a wider baseline with

a smaller vergence angle would enhance the calibration accuracy on the given fixation point. However, the baseline in use is the largest baseline available to implement on the ferry, thus an additional source of error.



(a) Depth estimate on given vergence angle



(b) Depth uncertainty given a change in vergence angle of 0.01 degrees

Figure 7.3: Depth estimate for a given vergence angle, and the error estimate for a deviation of 0.01 degrees in vergence angle

From the plots in Figure 7.3, one can see how sensitive the setup is to changes in angling. Error in angling can arise either from wrong calibration estimates or from external sources that reluctantly change the angling during the system's lifetime.

Irrespective of the angling, the baseline and focal length directly influence the uncertainty of the depth estimates. From equation 2.10, the uncertainty of the baseline and focal length will introduce approximately the same amount of uncertainty in the depth estimates. The translation in x-direction diverges 6 centimeters from the expected value, while no validity exists for the focal length. Therefore the average error from Chapter 5 is used as a foundation for uncertainty. The depth error from the test scenes yields to be approximately constant for the 20m calibration. Subtracting the expected one pixel error from Table 7.1 can be interpreted as an estimate of the overall uncertainties of the real stereo setup.

Distance	**Actual test-scene error**	**Expected reprojection error**
30 m	1.23 m	0.392 m
40 m	0.94 m	0.699 m
50 m	1.65 m	1.091 m
70 m	1.32 m	2.138 m
90 m	1.25 m	3.541 m

Table 7.2: Error from the real physical stereo camera, with the expected theoretical error

In Table 7.2, the reprojection errors are shown that were extracted from the test scenarios. By comparing the measured errors with the expected theoretical error, the experimental error turned out to be higher than in our theoretical consideration. The theoretical assumptions were more optimistic about the possible setup and measurement deviations. It can be caused by several reasons which finally accumulate to a higher error. This includes higher errors in the estimation of the baseline and the relative orientation of the cameras, but also higher errors in the estimation of the matching position of corresponding points.

The uncertainty of the physical stereo camera is expected to yield an even higher error for testing in the operating environment than in Table 7.2. The test scene is a bit unreliable as it contains an enormous building similar to the calibration scene. Besides, it utilizes semi-global matching which is shown to be more precise than our chosen real-time disparity-algorithm (described in the subsequent section). Thus, assuming an error of about 2-3 pixels seems to be more appropriate for the real physical stereo camera.

The finding from this subsection estimates the reprojection error to be highly dependent on the operating distance. As well, uncertainties in estimating the stereo setup have a negative impact on the overall accuracy. Therefore, higher uncertainty is to be expected. A more precise estimation of the error is to be determined by trial and error in the operating environment.

7.2 The disparity map algorithm

When processing images from a stereo camera, the corresponding pixels in the two images need to be matched to obtain depth information. Based on the evaluation of matching algorithms presented in the specialization project by Olsen (2020), a simple correlation-based method is preferred. Due to real-time computation, the chosen algorithm categorizes as a local method with a low computational cost.

7.2.1 Sum of Absolute Difference

Sum of Absolute Difference (SAD) is known for its simplicity and low execution time. It is one of the simplest methods for matching corresponding pixels but still has an advantage due to few memory requirements. (Thaher and Hussein, 2014)

The main idea is to find matching pixels by exploiting the neighborhood around a pixel. The neighborhood is a predefined block, and when used on rectified images, a suitable match is found on the same horizontal axis in the second image. The search starts in the same location as the reference block and moves along the horizontal line. A match is defined as the block with the most similarity. Similarity is computed by a cost function, measuring the absolute intensity difference between two selected neighborhoods.

$$C_{SAD}(d) = \sum_{(u,v)\in W_m} |I_L(u,v) - I_R(u-d,v)| \tag{7.5}$$

The cost function (7.5) measures absolute similarity, and the block with the minimum cost is considered a match. The intensity of pixel $(u-d, v)$ in the right image I_R is subtracted from the reference intensity in the left image I_L. The cost is summed over the

neighborhood W_m. For each pixel in the reference image, the cost function is calculated on the horizontal line to find the most suitable match. The cost function outputs a disparity for a given reference pixel, which is directly placed in the resulting disparity map. An example of a simple disparity map using default values is shown in Figure 7.5d.

The algorithm calculates the absolute differences between two selected window sizes W_m. The execution time and precision are dependent on the size of the window. A smaller window implies shorter execution time and is more prone to noise. Vice versa, a large window gives a smoother disparity map, but takes time to compute and may remove essential information. The preferred window size is dependent on the scene, the size of objects to be detected, and the computational constraints (Thaher and Hussein, 2014).

7.2.2 Filtering

SAD is mathematically a simple matcher, which is proven to be prone to errors. It is highly vulnerable to errors in the rectification parameters, also changes in illumination between the two stereo images (Nguyen and Ahn, 2019). A more robust algorithm can be implemented using filters. A filter can improve the resulting disparity map significantly with a relatively small computational cost.

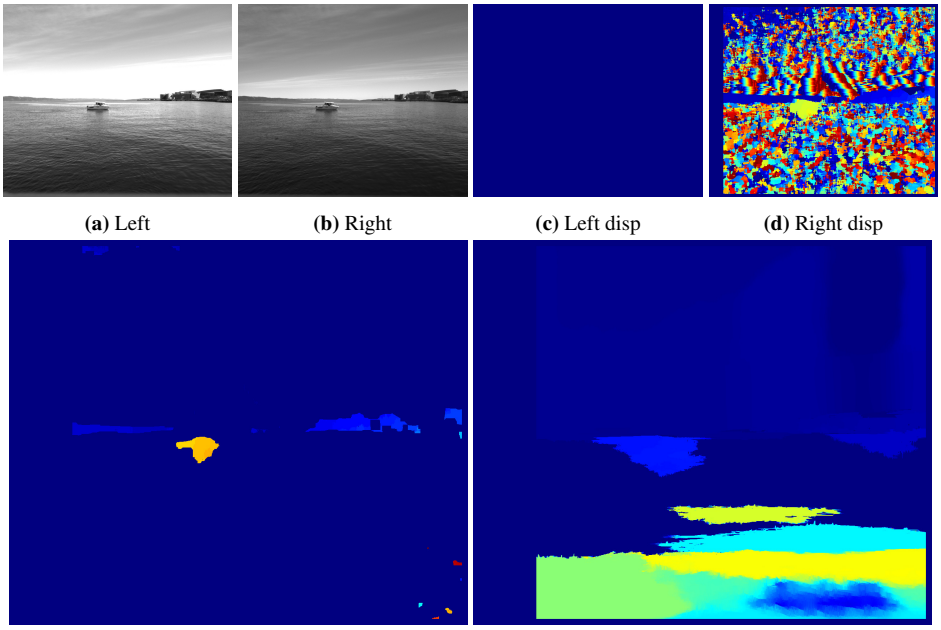
(a) Left stereo image **(b)** Right stereo image

Figure 7.4: Stereo image pair

In Figure 7.4, an example of a left and right stereo image is shown. The scene is challenging due to noise in the sea, and lack of texture above the horizon. Besides, the stereo images have different lighting, which makes it challenging for SAD to find corresponding matches. However, there are strong edges in both the pictures which a filter can preserve. Thus when filtering a disparity map from the two example images, an edge-preserving filter is desirable.

7.2.3 Fast Global Image Smoothing Based on Weighted Least Squares

Fast Global Image Smoothing Based on Weighted Least Squares is a method combining efficient edge-preserving filters and optimization-based smoothing (Min et al., 2014). Min



(a) Left **(b)** Right **(c)** Left disp **(d)** Right disp

(e) Resulting disparity map, small filtering values **(f)** Resulting disparity map, larger filtering values

Figure 7.5: FGS-WLS filtering using the Disparity Tuner

et al. proposed the algorithm in 2014 to enhance the quality of depth maps. The filter implements spatially inhomogeneous edge-preserving smoothing, commonly referred to as Fast Global Smoother (FGS). The proposed algorithm uses two disparity maps and the corresponding rectified grey-scale images for filter guidance and consistency. The first disparity map is created using the left image as a reference, and the second disparity map is computed with the right image as a reference.

$$\mathrm{E}(D_{out}) = \sum_{\mathbf{x}_\mathrm{L}} \left((D_{out}(\mathbf{x}_\mathrm{L}) - D(\mathbf{x}_\mathrm{L}))^2 + \lambda \sum_{\mathbf{x}_{\mathrm{L}2} \in \mathcal{N}(\mathbf{x}_\mathrm{L})} w_{\mathbf{x}_\mathrm{L}, \mathbf{x}_{\mathrm{L}2}}(L)(D_{out}(\mathbf{x}_\mathrm{L}) - D_{out}(\mathbf{x}_{\mathrm{L}2}))^2 \right) \tag{7.6}$$

In a Weighted Least Square smoothing, the output D_{out} is obtained by minimizing the energy function (7.6). The input disparity map D uses the left reference image L as guidance. The pixel $\mathbf{x}_{\mathrm{L}2}$ is contained from the pixel neighborhood of $\mathcal{N}(\mathbf{x}_\mathrm{L})$. The varying weighting function $w_{\mathbf{x}_\mathrm{L}, \mathbf{x}_{\mathrm{L}2}}(L)$ represent similarities in the two pixels and enforce the smoothness constraint. The energy function can be written in matrix form, and as it is strictly convex, the output D_{out} is solved by setting the gradient to zero. The solution can thus be computed by solving the linear system (7.7) (Li et al., 2016).

$$(\mathbf{I} + \lambda \mathbf{A}) D_{out} = L \tag{7.7}$$

The matrix A is a large sparse matrix, usually referred to as a Laplacian matrix. In the linear system, the input disparity map D and reference image L is written in vector format.

For efficiently solving the equation (7.7), one-dimensional Fast Global Smoothing is utilized. The key idea is that the matrix $\mathbf{A}$ is a tridiagonal matrix in a one-dimensional system. For images in grey-scale, one dimension means horizontal or vertical inputs. By solving a series of multiple linear equations over the total image, the exact solution of equation (7.6) can be solved. The one-dimensional linear equations are solved by the Gaussian elimination algorithm (Golub and Van Loan, 2013) in a recursive manner. By repeating the linear equation over horizontal lines in the image, the result is the minimum solution of (7.6).

In Figure 7.5, the two guidance images are displayed together with the two disparity images by SAD. Figure 7.5e and 7.5f shows the resulting disparity map from executing the Fast Global Smoothing based on WLS filtering.

7.2.4 Implementation

```
1  //Creating a disparity map from two rectified images
2  void DisparityNodelet::imageCallBack(left_rectified_image,
       right_rectified_image, calibration_params)
3
4      convert images to opencv lib
5      downscale images //speed up the execution, optional
6
7      compute left disparity map
8      compute right disparity map
9      WLSfilter(left image, left disparitymap, right image,
       right disparity map)
10
11     upscale image //optional
12
13     Adjust offset in principal point
14
15     Convert to ros msg
16     publish(disparityMap)
```

Listing 7.1: Create disparity map

The algorithm is notated in pseudo-code in Listing 7.1. The image callback is a ROS-function which is executed every time a new image is published to a rectification-topic. It listens to camera calibration parameters and the left- and right rectified images. Whenever two images are published, the function is called, and it checks that the timestamp of the images is equal. The stereo image pairs are further converted to use the OpenCV image library. The first disparity map is computed using SAD with the left image as the reference. The second disparity map uses the right image as a reference. Both matching pixels with the fast and efficient stereo matching algorithm based on Konolige (Konolige, 1998). Fast Global Smoothing combines the two disparity maps with both the rectified images for preserving edges and smoothing the result. The resulting disparity map is subsequently published to a topic for later creating the world position for regions of interest.

The disparity map is calculated using standard stereo vision techniques. However, a tricky part of creating a depth map is tuning all the parameters. For an easier tuning of the disparity map, a GUI for dynamically tuning rectified images is implemented.

7.2.5 Disparity Tuning

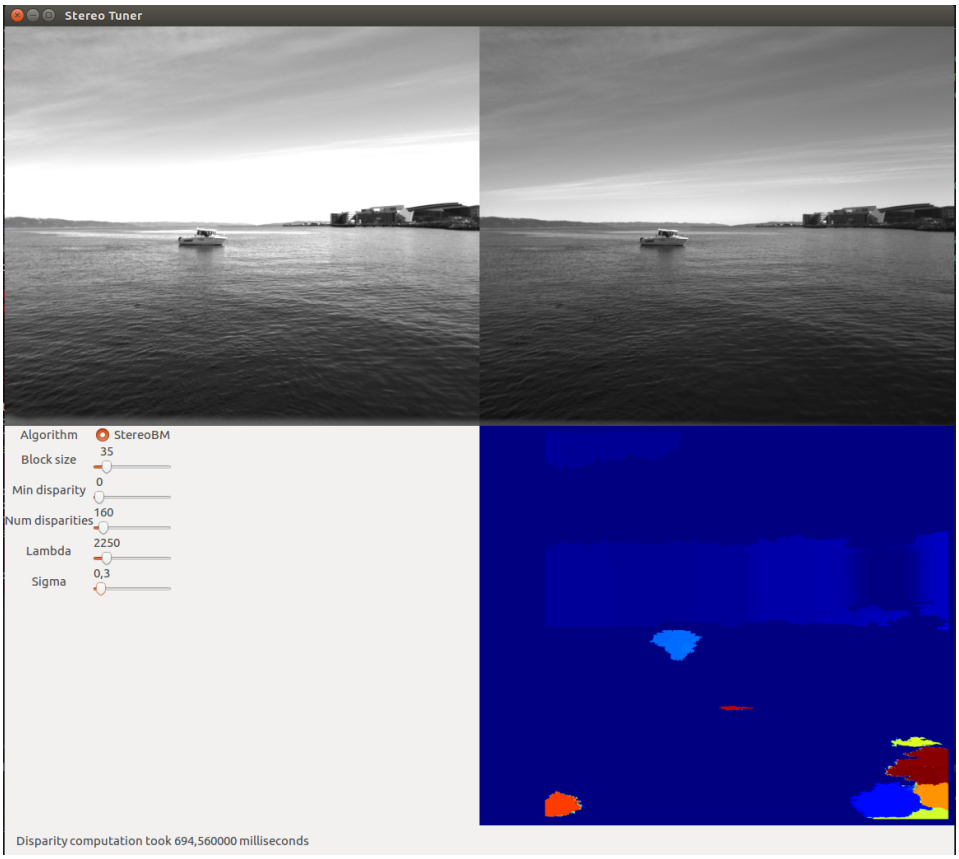


Figure 7.6: The stereo tuner application

For the overall system to perform, it is dependent on a sufficiently tuned disparity map. Tuning parameters is demanding, especially finding parameters suitable for different depths and lighting conditions. A GTK application is created to tune parameters on the stereo matching algorithm implemented. It is written in C++ using openCV, and is a combination and modified version of the stereo-tuner[1] and the opencv_contrib[2] repository, both can be found on Github.

[1] `https://github.com/guimeira/stereo-tuner`

[2] `https://github.com/opencv/opencv_contrib/blob/master/modules/ximgproc/samples/disparity_filtering.cpp`

The application has a graphical interface where parameters can be changed dynamically. A screenshot of the application is displayed in Figure 7.6. The two upper images are the rectified stereo images from the left and right camera, respectively, while the lower right image is the resulting disparity map. Only the parameters influencing the disparity map is chosen in the application. This is the block size for SAD and the disparity range for searching for matches on the horizontal line. For the filter, sigma and lambda are included for tuning. Lambda, λ in (7.6), defines the amount of regularization during filtering. Large values will adhere to the disparity map edges closer to the guidance image. Sigma defines the sensitivity of the filtering. Larger values smooth the resulting image but may miss some of the corners.

7.3 2D Object Detection with YOLO

The detection of objects in an image is twofold. It consists of locating the exact position of objects in an image, and subsequently classify each object. YOLOv3 (Redmon and Farhadi, 2018) is a fast and accurate real-time detection system. In the overall system, YOLOv3 is implemented to locate objects of interest in the left stereo image. By combining the disparity map found in Section 7.2 with a trained network, it can output positions of detected objects in world coordinates.

YOLO is a single convolutional neural network, engineered to detect patterns in images. It is a type of neural network, a special kind of computer algorithms.

Neural Networks

A neural network is inspired by the knowledge of the inner workings of the human brain. It consists of perceptrons, which is a mathematical model of a biological neuron. The nodes are connected to one another to model pattern recognition capabilities. A network is made up of layers, and they can be trained to perform specific tasks like image recognition, image classification, and image captioning.

A neural network comprises one input layer, one output layer, and at least one hidden layer. Figure 7.7 illustrates a simple three-layer neural network consisting of only one hidden layer. Each input of a neuron is multiplied with a weight w_i. The weighted sum, a, of a neuron is given in (7.8).

$$a = \sum_{i=1}^{n} w_i x + b \tag{7.8}$$

The signal or value x_j represents the input, and b is applied to allow for a constant bias to the neuron. The neuron generates an output when the weighted sum exceeds an activation threshold set by an activation function. The parameters of the weights and biases are set during training. When training a neural network, the parameters are tuned to find the desired output. (Krenker et al., 2011)

Convolutional Neural Networks

Convolutional neural networks, CNN, is a special case of neural networks. CNN's are made of mostly convolutional layers, which are designed for having images as inputs. The

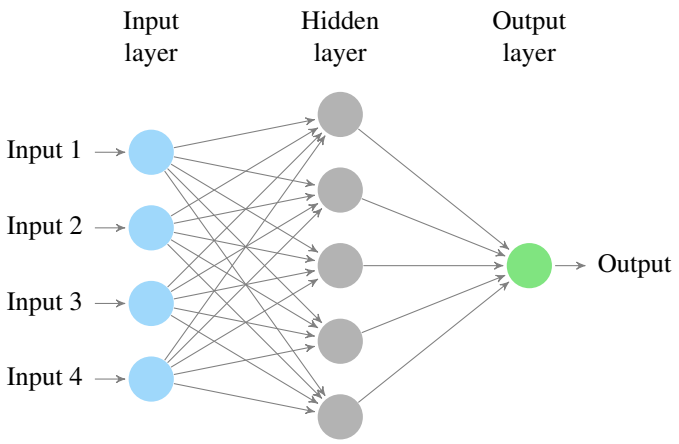


Figure 7.7: Neural netwrok with three layers

dimensions of a color image consist of the width W, height H, and three channels. Inputs of such a high dimension would require a large number of neurons when using neural networks. CNN takes advantage of the assumption that the input is an image and uses this to process the input.

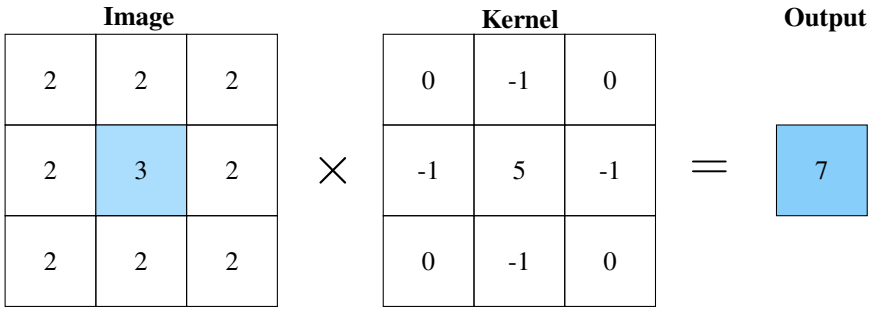


Figure 7.8: Caption

CNN consists of three types of layers; convolutional, pooling, and fully-connected layers (O'Shea and Nash, 2015). The input layer holds the pixel values of the input image. The convolutional layers consist of a kernel that is slid over the image. Every time the kernel is moved, matrix multiplications are performed over the image region. Figure 7.8 illustrates the operation. The objective of the operation is to extract features such as edges or corners. The features extracted depends on the value of the kernel. The pooling layers performs a downsampling of the input. The dimension of the input is reduced in order to decrease the computational power necessary to process it. Convolutional layers and pooling layers are similar, and usually placed after one another. Fully-connected layers perform the same operations as in neural networks and are used for classification.

7.3.1 YOLOv3

YOLO - You Only Look once is one among other open-source neural networks. It is written in C, and a CUDA interface is available for GPU computation, making it suitable for real-time detection. The main idea is that the system only looks at an image once, framing object detection to a single regression problem (Redmon et al., 2015). A single convolutional neural network is applied to the full image. Thus, from looking at an image once, the system can predict what objects are present and their location in the image. An

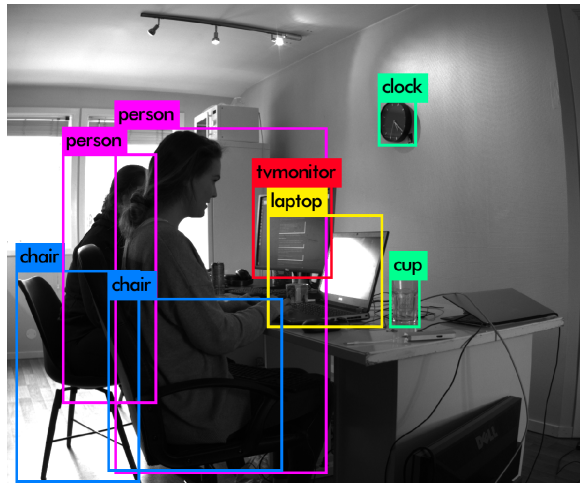


Figure 7.9: Output from YOLO

example of the output given by a pre-trained YOLO network is presented in Figure 7.9. The bounding boxes tell what objects are present. The presented image is run through a pre-trained network for classification and object recognition. It is a pre-trained model, made available in the download of the YOLOv3 system.

By using two fully connected layers, YOLO performs a linear regression to make boundary box predictions. The logistic regression calculates a probability, which is returned as a binary value given a predefined threshold. Thus, YOLO keeps predictions with a high box confidence score. To measure the confidence in both the classification and the objects positioning the class confidence score is computed as:

$$\textit{Class confidence score} = \textit{Box confidence score} \times \textit{Conditional class probability}$$

The networks confidence and quality varies with the chosen threshold, and is further evaluated in the next subsection.

The network of neurons can be trained on custom datasets. The pre-trained weights, which are made available online, is only an appreciated starting point for creating a network suitable for more specific tasks of own interest. Using a custom dataset allows the user to specify the detection classes and to improve the classification in a specific operating environment. YOLOv3 is a network of 106 layers, and it allows for each layer to be customized. The chosen network was used due to the availability of pre-trained weights

developed during a Master thesis project by Grini (2019) at NTNU. The framework is trained for detecting boats in images. The dataset consists of images captured in and around the fjords in Trondheim, combined with some online datasets.

The network is implemented using a pre-exisiting ros-package[3], and combined with the new weighs. To accelerate the execution the package is utilized by installing the Nvidia CUDA Toolkit 10.2. This boosts the network to use parallel computing for real-time performance. A pre-trained network requires a threshold in percentage for classifying objects. The training dataset in its whole and the confidence threshold of the application is unknown. For this reason, an analysis of the quality is conducted before taking it to use.

7.3.2 Precision recall curve

Precision-Recall metric is a useful measure to evaluate the classifiers output quality. By using different probability thresholds, the curve summarizes the trade-off between the true positive rate and the positive predictive value for the network. Precision $\mathbf{P}_n$ (7.9) defines the true positive rate for the threshold n, and recall $\mathbf{R}_n$ (7.10) the positive predicted value for the threshold n. With the given weights in the thesis, the classifier is binary, i.e., boat or not boat.

$$\mathbf{P}_n = \frac{TP}{TP + FP} \tag{7.9}$$

$$\mathbf{R}_n = \frac{TP}{TP + FN} \tag{7.10}$$

Both the functions consist of the abbreviations in (7.11).

$$\begin{aligned} TP &= \text{True positive:} \quad \textit{boats correctly classified} \\ FP &= \text{False positive:} \quad \textit{miss-classifications} \\ FN &= \text{False negative:} \quad \textit{boats not classified} \end{aligned} \tag{7.11}$$

High precision shows that the classifier is accurate and does not label wrong objects as boats. The recall score is the ability of the classifier to find all the boats in the picture. Lowering the threshold of the neural network may increase the precision's denominator by increasing the number of results returned. The denominator of the recall does not depend on the classifiers threshold, implying that lowering the threshold will mostly increase the recall. If the recall leaves unchanged while changing the threshold, the precision may fluctuate.

The values are usually comprised in a confusion matrix, defined in Table 7.3 The last

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Table 7.3: Confusion matrix

[3] `https://github.com/leggedrobotics/darknet_ros`

value, TN, is defined as no boats present nor detected. When detecting and classifying objects in a picture, it makes no to little sense using TN, because a negative class does not exist. The negative class is defined as parts of the image not including boats.

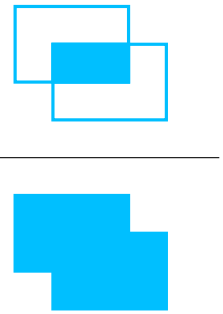
Computing the curve

To evaluate the accuracy of the network a ground truth dataset was labeled. The images are taken in the test area outside the port of Brattøra. The dataset and the graphical user interface are presented in Appendix D. The labeled images are the ground truth, implying a squared PRC with both precision and recall equal to zero. It is the same boat, just captured at various distances and with different headings. In theory, a perfectly trained network should be able to detect the same boat with about the same confidence and similar sized boxes.

		Predicted	
		Positive	Negative
Actual	Positive	483	0
	Negative	0	-

Table 7.4: Ground-truth Confusion matrix

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} =$$



The ground truth bounding boxes are compared with the bounding boxes predicted by YOLO. To get a comparison, the Intersect over Union, IoU, is calculated. The predicted bounding box is a match if it shares the same label as the ground-truth and has confidence- and IoU- value above the given thresholds. Only one predicted bounding box can be assigned to the ground truth to avoid multiple detections of the same object. The pseudocode of the implementation is given in Listing 7.2. The implementation is given in the GitHub-repository explained in Appendix C. The PRC is calculated for each IoU- and YOLO-threshold, of all the images in the dataset.

```
for each IoU-threshold
    for each YOLO-threshold
```

```

        sort detection-results by decreasing confidence
        for each detection confidence >= YOLO-threshold
            for each class
                assign detected box to ground-truth object
                if IoU >= IoU-threshold
                    TP ++
                else
                    FP ++

            Precision = TP/(TP+FP)
            Recall = TP/ (number of ground truth objects)
Plot curve
```

Listing 7.2: Pseudocode plot PRC

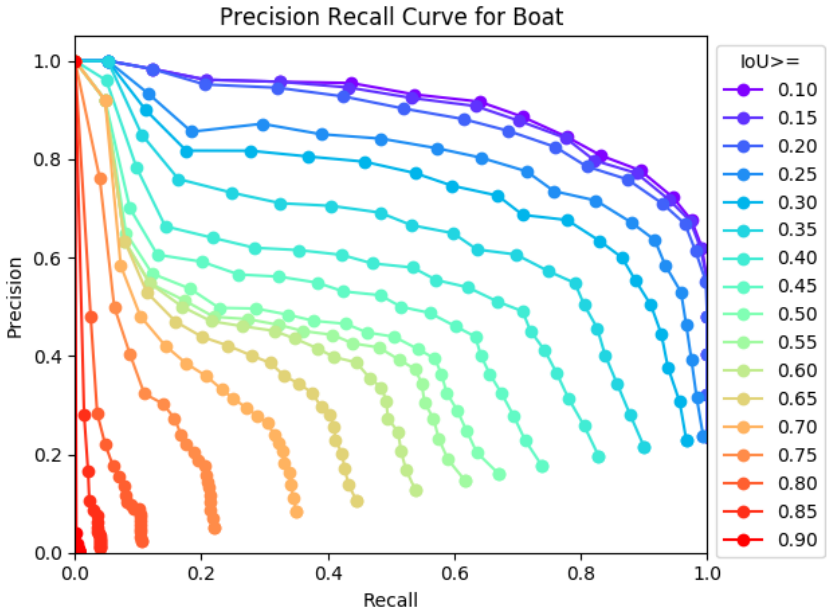


Figure 7.10: Precision Recall Curve

The resulting plot is presented in Figure 7.10. Each colored line represents the given IoU-threshold used for counting the True Positive detections. The dots represent a given threshold for the YOLO-network. The YOLO-threshold is plotted from 0 to 1 with a step size of 0.05. The YOLO-threshold of 0.95 and 0.005 of each curve are given in the top left and lower right corners, respectively. The graph with the largest area under the curve has IoU= 0.10%. The node giving the largest possible area is computed with a YOLO-threshold of 0.45. The associated confusion matrix is given below and corresponds to a

precision of 0.78 and a recall at 0.89.

		Predicted	
		Positive	Negative
Actual	Positive	432	51
	Negative	124	-

Table 7.5: Confusion matrix. IoU-threshold: 0.10, YOLO-threshold: 0.45

However, what IoU is acceptable to evaluate the network depends on the intended use. The less area of overlap acceptable, the better the resulting curve. To get a visual overview of what IoU-threshold is acceptable, some examples are given in Figure 7.11. The blue boxes are the ground truth boxes manually labeled, and red is YOLOs predictions. The intersect over union is given in the caption.

In theory, the area of overlap should be as high as possible, but from the figure, it is observed that 80% overlap is close to perfect. With an IoU-threshold of 0.80, the YOLO-threshold with the best results is 0.94. This corresponds to the precision of 0.5, recall of 0.03, and an area of 0.016. The results are given in the following confusion matrix: A precision of only 0.5 means that if an object is detected, there is only a 50% chance

		Predicted	
		Positive	Negative
Actual	Positive	15	468
	Negative	15	-

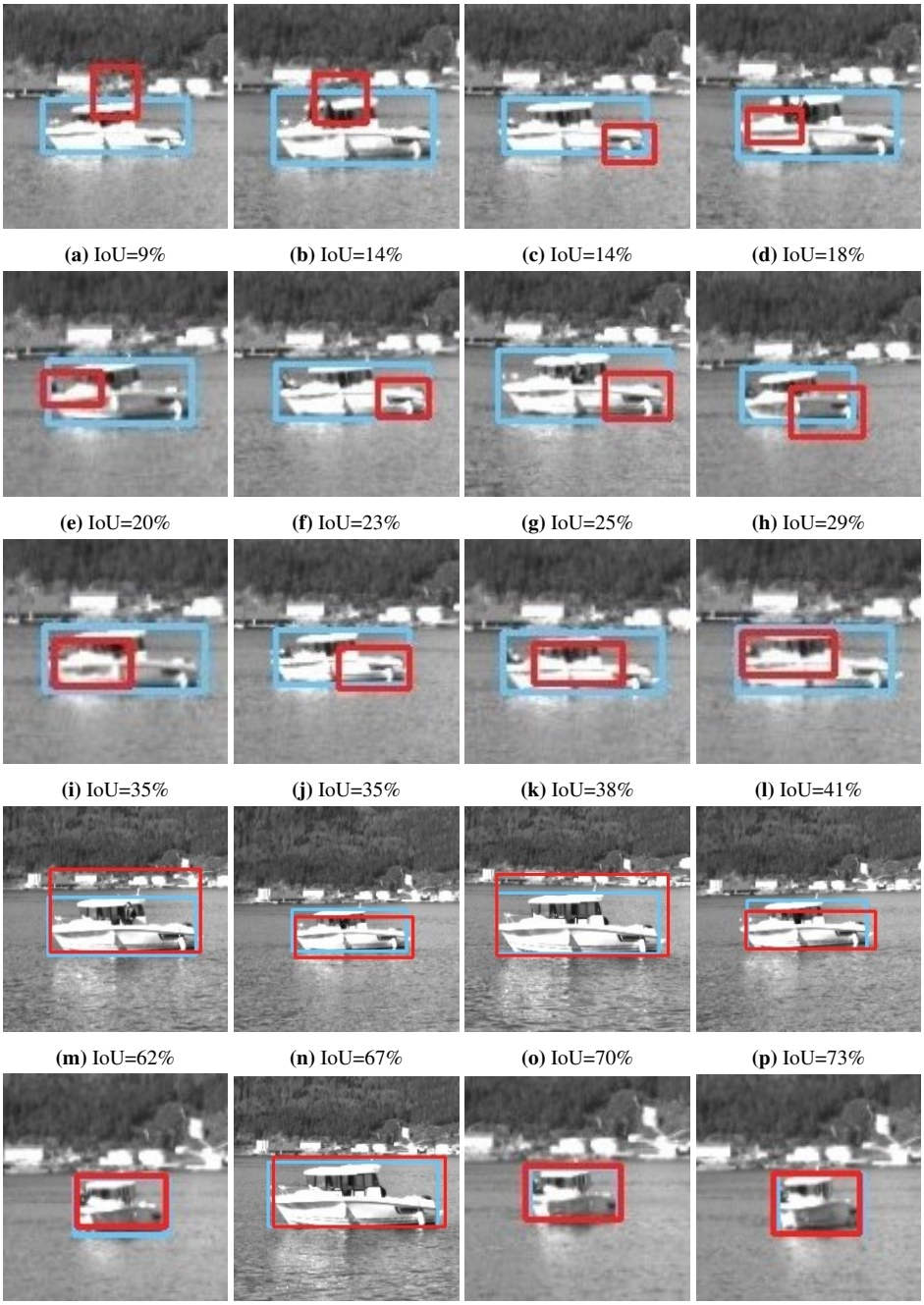
(a) YOLO-threshold: 0.94

		Predicted	
		Positive	Negative
	Positive	50	433
	Negative	687	-

(b) YOLO-threshold: 0.32

Table 7.6: Confusion matrix. IoU-threshold: 0.80

that it is a boat. Likewise, a recall of only 0.03 entails that 3.0% of all the boats are detected. Reducing the YOLO-threshold will result in a higher recall but do not give any considerable improvements, as seen in Table 7.6b. Setting the threshold closer to zero results only in small improvements in TPs, which indicates that most of the predicted boxes are too small compared with the ground truth.



(a) IoU=9% **(b)** IoU=14% **(c)** IoU=14% **(d)** IoU=18%

(e) IoU=20% **(f)** IoU=23% **(g)** IoU=25% **(h)** IoU=29%

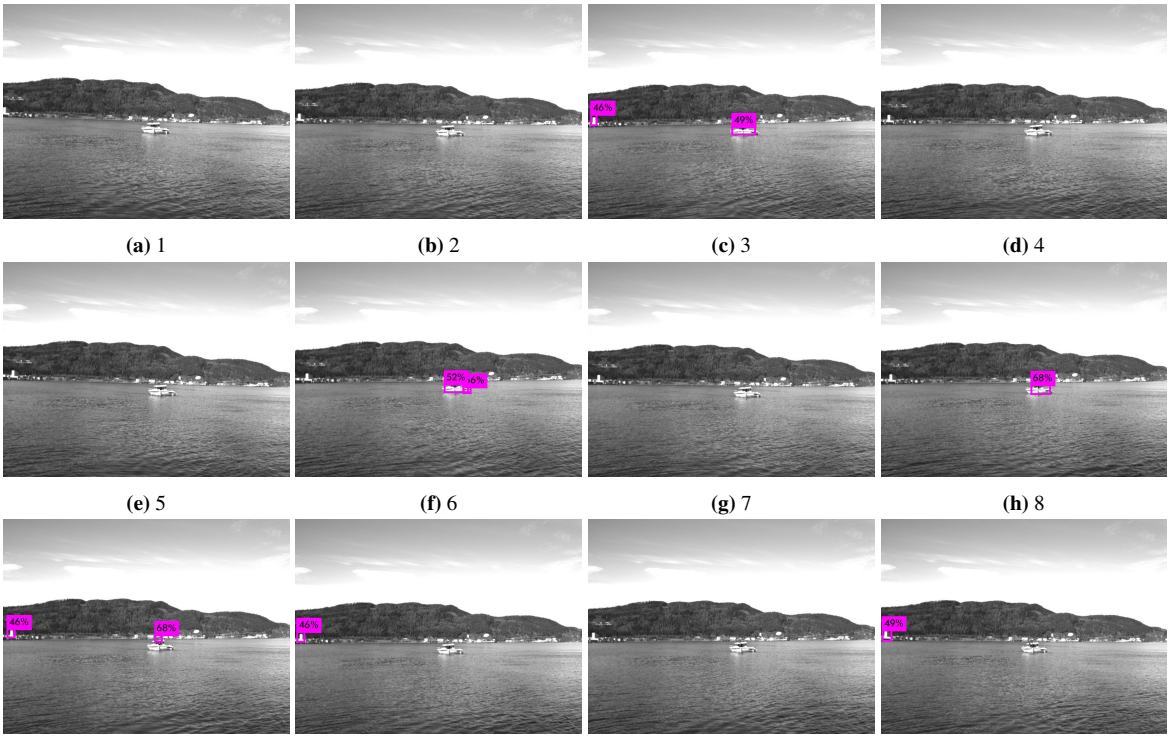
(i) IoU=35% **(j)** IoU=35% **(k)** IoU=38% **(l)** IoU=41%

(m) IoU=62% **(n)** IoU=67% **(o)** IoU=70% **(p)** IoU=73%

(q) IoU=80% **(r)** IoU=86% **(s)** IoU=90% **(t)** IoU=91%

Figure 7.11: Example of the IoU with ground truth in blue and YOLO predictions in red

Looking at the pictures in Figure 7.11, one can estimate the preferred IoU-threshold for the application. With IoU less than 0.2, the predicted bounding boxes seem to detect too much of the surroundings. However, increasing the threshold above 0.2, most of the predicted bounding boxes primarily detect the boat. The bounding boxes are too small compared to the ground truth, but parts of the boat are detected. The network is implemented together with the disparity map to get depth estimates. By using a disparity with some smoothing, the same depth values will yield for every pixel where the boat is present. Thus, YOLO only needs to detect a small part of the boat for the system to extract it from the disparity map and reconstruct its 3D position.



(a) 1 **(b)** 2 **(c)** 3 **(d)** 4

(e) 5 **(f)** 6 **(g)** 7 **(h)** 8

(i) 9 **(j)** 10 **(k)** 11 **(l)** 12

Figure 7.12: Example of predictions in a time sequence, images in chronological order. YOLO-threshold: 0.44

For IoU-threshold of 0.2, the results from differing the YOLO-thresholds is written in the table in Appendix E. From the table, the YOLO-threshold giving the largest area is 0.44. The confusion matrix is given in Table 7.7. With a recall of 89%, the network still misses out on 52 out of 483 boats in the images. Not detecting 11% of the boats can be critical for an autonomous navigation system, increasing the likelihood of a collision. However, if the probability for detecting a boat is independent and 89%, setting the frame rate to about 10 will ensure enough TPs to establish the presents of a boat. Looking at the dataset in Figure 7.12 and 7.13 comparing two time-sequences of images, one can

observe that the probability of detecting a boat is indeed dependent on the scene. In the first time-sequence, YOLO detects 25% of the boats, while in the second it detects 100% of the boats. Thus, as the system is meant for collision avoidance, the YOLO-threshold

		Predicted	
		Positive	Negative
Actual	Positive	431	52
	Negative	146	-

Table 7.7: Confusion matrix. IoU-threshold: 0.20. YOLO-threshold: 0.44

is decreased to increase the recall. Even though this implies decreasing the precision, detecting all boats is of higher importance. From the table in Appendix E, a recall of 1 can be obtained with a threshold of 0.2. The confusion matrix is given in Table 7.8, with three additional thresholds of interest.

Predicted	
Positive	Negative
483	0
525	-

(a) YOLO-threshold: 0.20

Predicted	
Positive	Negative
482	1
368	-

(b) YOLO-threshold: 0.26

Predicted	
Positive	Negative
480	3
350	-

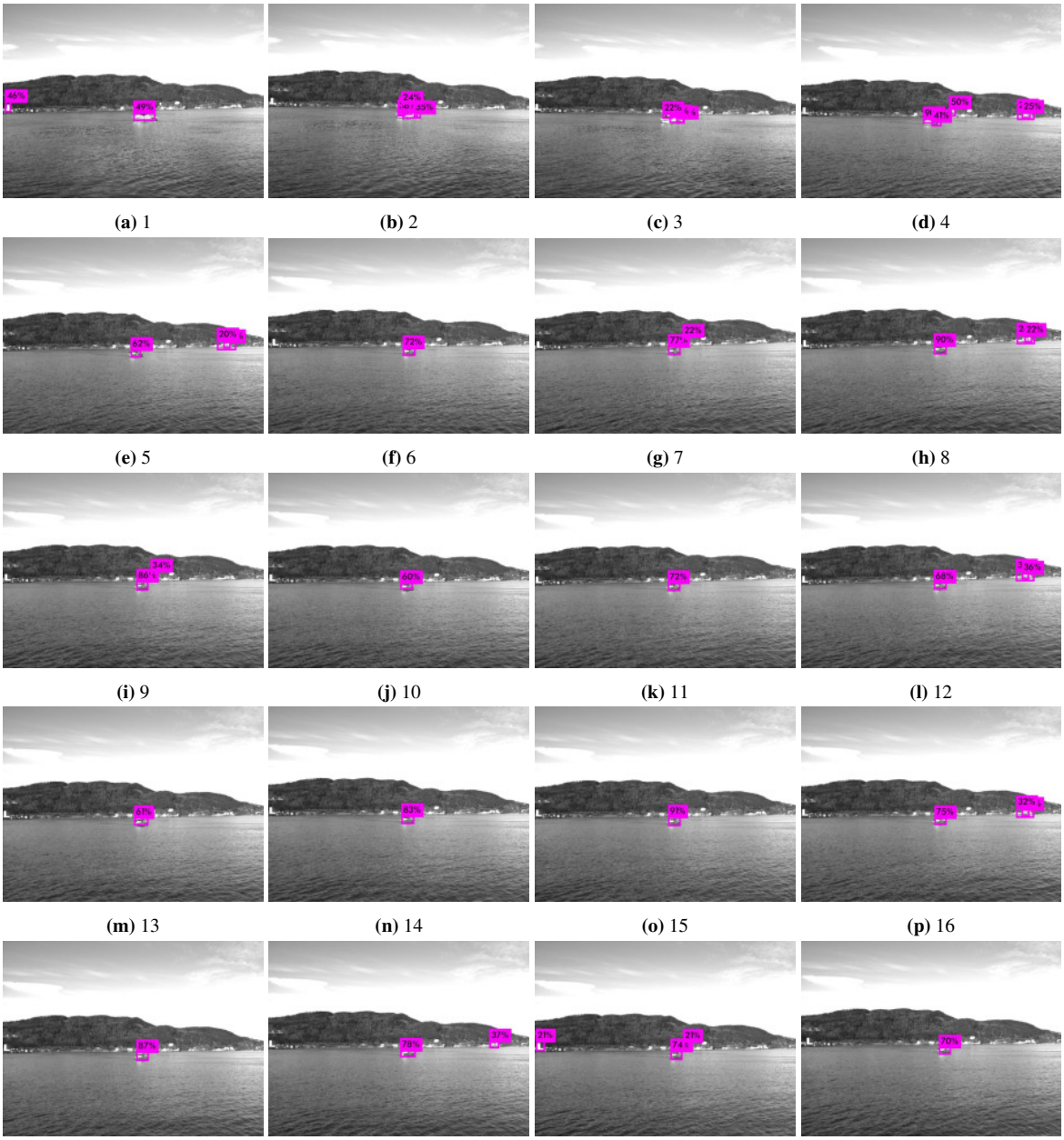
(c) YOLO-threshold: 0.27

Predicted	
Positive	Negative
471	12
256	-

(d) YOLO-threshold: 0.33

Table 7.8: Confusion matrix. IoU-threshold: 0.20

The optimal threshold is a matter of the intended use of the system. In the case of object detection, to be able to trust the output, a well-trained network would be preferred. As the available YOLO-weights struggle with detecting the same boat, a higher recall is preferred to detect as many boats as possible. A higher recall will increase the number of FPs. A miss-classification of 525 images is not that bad considering the wrong match in Figure 7.13. In the case of multiple matches, the one with the lowest confidence score is chosen as the TP, the other as FP. In total, around half of the FPs are because of multiple matches on the same boat. However, YOLO tends to miss-classify the same buildings as boats, which implies that the probabilities are not independent. Therefore FPs should be minimized. Regardless, the thresholds of 0.26, 0.27, and 0.33 are added to see the change in FPs and FNs. Increasing the threshold from 0.20 to 0.26 shows quite an improvement. The FP is decreased with 157 boats, and the network only misses one boat.



(a) 1 **(b)** 2 **(c)** 3 **(d)** 4

(e) 5 **(f)** 6 **(g)** 7 **(h)** 8

(i) 9 **(j)** 10 **(k)** 11 **(l)** 12

(m) 13 **(n)** 14 **(o)** 15 **(p)** 16

(q) 17 **(r)** 18 **(s)** 19 **(t)** 20

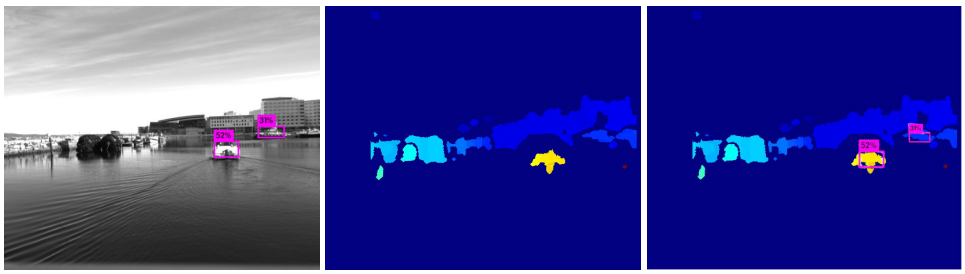
Figure 7.13: Example of predictions in a time sequence, images in chronological order. YOLO-threshold 0.20

The threshold chosen in the thesis is 0.26. In the ground truth dataset, the network only

misses one boat, but keep in mind that on a different scene more FP and FN can occur. This can be considered as acceptable with a higher frame rate and a well-tuned disparity map. As the network can process about ten images per second, depending on the GPU, a missing boat in one image can be resolve by combining several images.

7.3.3 Using CNN for clustering

Figure 7.14 illustrates the achievement of the world position by combining YOLO 2D object detection with the corresponding disparity map. Each bounding box in Image 7.14c



(a) Result from YOLO **(b)** Corresponding disparity map **(c)** Combining the two

Figure 7.14: Require 3D position of detected objects

extracts the median disparity value for depth estimates. The bounding box's center position, $\mathbf{x} = (u_{\mathrm{L}}, v_{\mathrm{L}})$, with the depth is extracted to calculate the world positioning of each object. The characteristics of the clustering technique makes a denser disparity map more optimal as the bounding boxes in some cases overestimates the targets size. Thus a large matching window, with more smoothing is chosen. The advantage of using a neural network for clustering is clearly that a lot of the noise in the disparity map is easily filtered out. As well, the values in the disparity map can remove falsely detected boats. Figure 7.15 illustrates this. Grey color defines infinity or invalid, and hence the four miss-classified boxes will not be considered valid objects. Deep learning is shown to be efficient, and YOLO is running concurrently with the creation of the depth map, making it time-efficient.

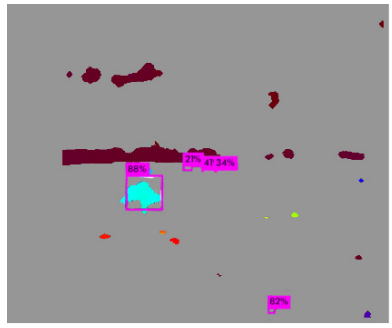


Figure 7.15: Example of dispairty map improving YOLO

7.4 Point Cloud Clustering

The images captured by a stereo camera are rectified and utilized in the creation of a disparity map. A disparity map contains depth information needed to reconstruct the captured scene in 3D. In both the disparity map and the reconstructed point cloud, objects in the scene appear as a tight bundle of points. This makes clustering a suitable method for object detection and localization. As the point cloud is a direct product of the disparity map, it is important to preserve edges for acquiring the correct localization of objects. This is achieved by the disparity map, by using a smaller matching window with larger lambda values. This makes the point cloud more vulnerable to noise, making filtering procedures important before the clustering.

Clustering categorizes data points such that similar points belong to the same category, and unalike points belong to other categories. Various algorithms can achieve this. The algorithms have different definitions of what a cluster is and how to find points belonging to them. The number of categories can be both defined and unknown, depending on the algorithm. The appropriate algorithm must be chosen depending on the intended use of the result.

7.4.1 Hierarchical clustering

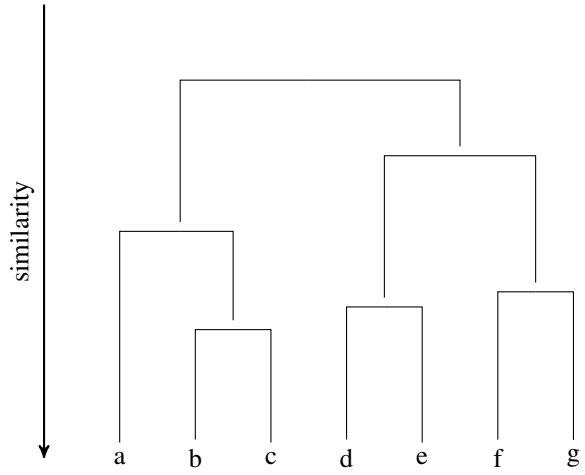


Figure 7.16: Dendrogram from single-link algorithm

Hierarchical clustering creates a dynamical number of clusters making it suitable for the variable environment the stereo camera operates in. Hierarchical clustering produces a final set of clusters by successively merging or splitting clusters based on similarity. Hierarchical clustering can thus be divided into two types; agglomerative and divisive. Agglomerative clustering uses a bottom-up approach, while divisive clustering uses a top-down approach. Agglomerative clustering starts by specifying each data point as separate clusters. The goal is to meet a similarity measure by merging clusters. Divisive cluster-

ing begins by defining all data points as one cluster and splits them until the similarity threshold is met.

Hierarchical clustering algorithms can further be divided by their similarity calculations. The three algorithms are the single-link, the complete-link, and the average-link algorithm. Single-link uses the minimum distance between two clusters, while complete-link uses the maximum distance. In average-link, the similarity is a measure of the average distance between the members of the clusters. There are drawbacks and advantages of each method.

The hierarchical clustering methods produce a dendrogram. In a dendrogram, the categorization of the points and the similarity levels where the categories change are represented. Figure 7.16 shows a dendrogram obtained from a single-link algorithm.

7.4.2 Implementation of Euclidean clustering

The system implements a Euclidean clustering extraction for segmenting point clouds into clusters. The method is a single-link hierarchical clustering procedure and extracts clusters based on Euclidean distance as similarity measurement. The code is implemented with the Point Cloud Library and uses a simple data clustering approach summarized in Listing 7.3. The code makes use of nearest neighbors approach.

```
1  //segmentation class for cluster extraction in an Euclidean
       sense
2  PointCloud::Ptr PclObstacleDetector::cluster_cloud(const
       PointCloud::ConstPtr &cloud_input)
3      Create a Kd-tree for  //search method
4      Create empty list of clusters, and an empty queue
5
6      for every point in cloud_input
7          add point to queue
8
9          for point in queue
10             search for neighbors within threshold
11             if not in queue
12                 add to queue
13         add queue to clusters, set queue to empty
14
15     create empty centroid_cloud
16     for each cluster
17         add centroid to centroid_cloud
18     publish(centroid_cloud)
19
20 void cluster_callback(centroid_cloud, position milliAmpere)
21     for each centroid
22         calculate NED coordinates
23     publish(detected objects)
```

Listing 7.3: Create disparity map

The algorithm falls under the category agglomerative clustering. All points start in separate clusters, and based on the Euclidean distance points are merged into the same cluster.

The raw stereo point cloud is too dense and noisy for segmenting out objects of interest. Thus, the unorganized point cloud is pre-processed using several filtering techniques. The written class *PclObstacleDetector* implements both the clustering and filtering, and is comprised in the cloud callback in Listing 7.4. Figure 7.17 illustrates the output from each function.

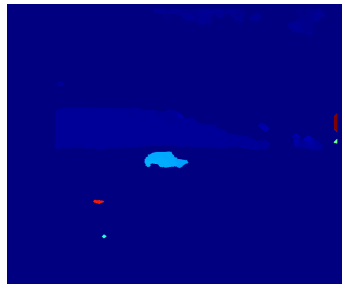
```
1  /***
2  ** Point cloud callback
3  ** - PassThrough filter
4  ** - Statistical Outlier Removal
5  ** - Voxel Grid filter
6  ** - Combine point clouds
7  ** - Euclidean clustering
8  ***/
9  void PclObstacleDetector::cloudcb(const PointCloud::
       ConstPtr &cloud_input){}
10     &cloud_cut = cut_cloud(cloud_input);
11     &cloud_sor = sor_cloud(cloud_cut);
12     &cloud_vg  = vg_cloud(cloud_sor);
13     &cloud_acc = accumulate_cloud(cloud_vg);
14     &cloud_clustered = cluster_cloud(cloud_acc);
15
16     pub_cluster.publish(cloud_clustered);
17 }
```

Listing 7.4: Clustering callback

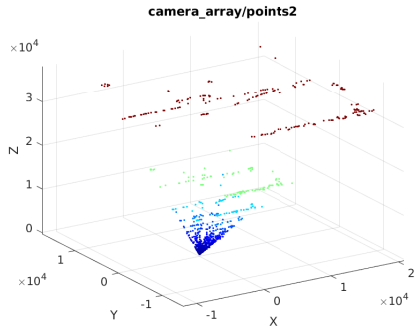
The function in line 10, cut_cloud, utilizes a pass-through filter. It is in all simple manners just cutting of the point cloud for values over a given threshold in the z-dimension. The threshold is obtained from the calibration chapter concluding that the stereo setup is nowhere capable of acquiring accurate depth estimates above 250 meters. The threshold will also remove the numbers at infinity, making it easier to visualize and post-process.

Subsequently, sor_cloud is removing noisy measurements using statistical analysis techniques. The algorithm takes the mean distance from a query point to each point in a neighborhood. The points whose distance is larger than a standard deviation multiplier is considered as outliers and removed from the point cloud. Figure 7.17e displays an example output from the function. As one can observe, the point cloud is, in theory, ready for clustering. Directly clustering the output with Euclidean clustering, a computationally efficient computer takes about 130 milliseconds to output the clustered centroids.

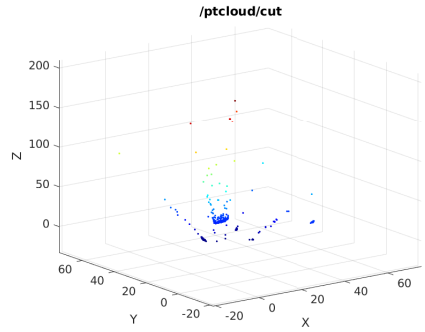
(a) Left image



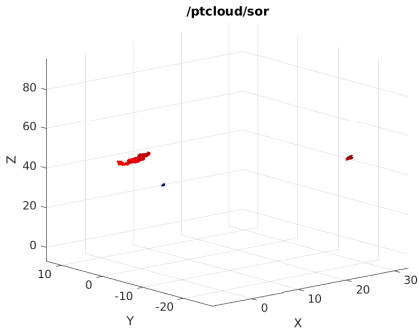
(b) Corresponding disparity map



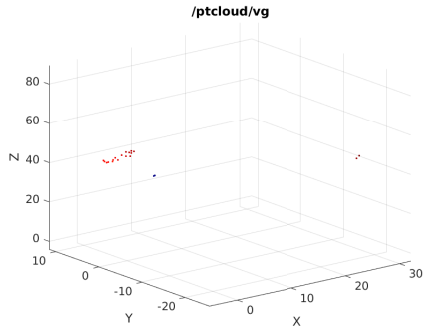
(c) Raw point cloud from disparity map



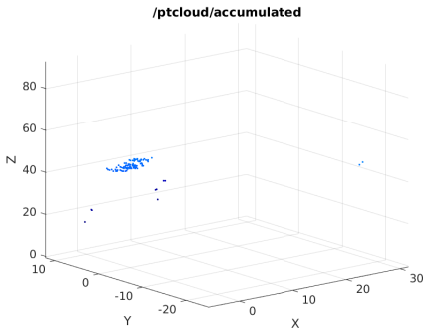
(d) Point cloud after cropping



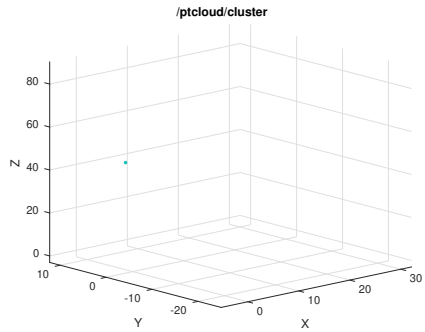
(e) Point cloud after Statistical Outlier Removal



(f) Point cloud after Voxel grid



(g) Point cloud accumulated 6 times



(h) Point cloud after clustering

Figure 7.17: Point cloud processing

The implementation of the voxel grid filter is for real-time considerations. The voxel grid downsamples the point cloud, approximating a neighborhood of points to the centroid. Using the filter enhances the computational time more than ten times! The last preprocessing function in line 13 accumulates the point clouds. It assembles subsequent point clouds for redundancy, increasing the reliability of the system. The function anticipates shortcomings in the stereo matching or filtering. If the stereo matching algorithm fails to match the objects of interest in one image, the accumulate function will include the missing parts from the already processed point clouds. Also, objects of interest will be denser, while noisier parts like sea waves remain sparse. The complete callback in Listing 7.4 takes about ten milliseconds to compute accumulating 6 point clouds, by all means depending on the computer.

Both of the presented clustering techniques are used for object detection on milliAmpere. They are tested in the operational environment, and their performance is evaluated and compared.

Chapter 8

Test results in marine environment

The experiment and data collection took place outside the port of Brattøra. A handheld Global Positioning System (GPS), placed on the target boat, serves as the ground truth. The test area and the GPS route are shown in Figure 8.1. The origin of the NED coordinate frame is set inside the harbour. The performance of the stereo parameters obtained at 20

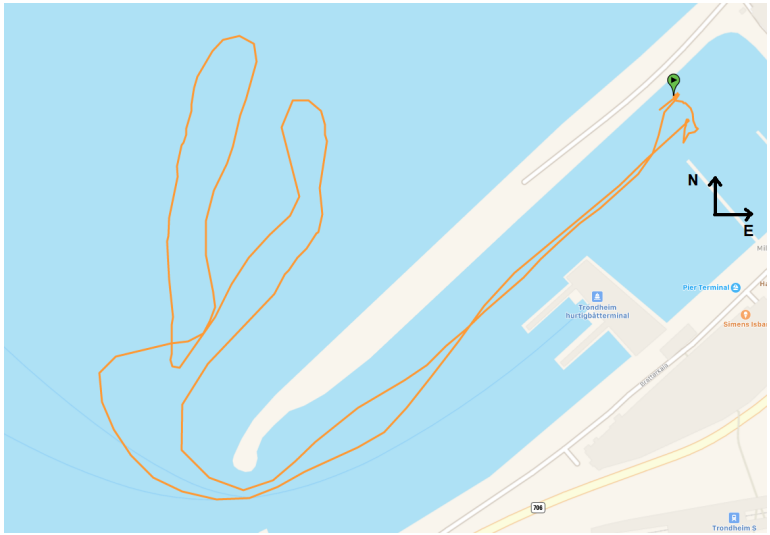


Figure 8.1: Test area and route of the target

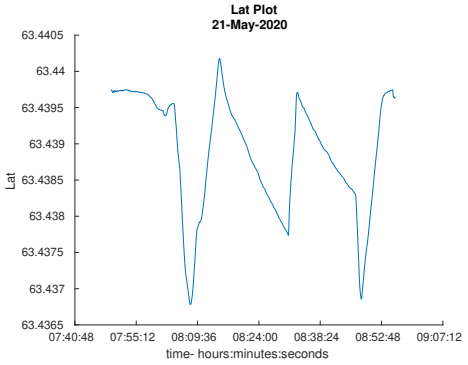
meters, and the algorithms described in Chapter 7 are tested and evaluated.

The experiment is divided into seven different scenarios, capturing the target boat at different distances. The scenarios are numbered in chronological order by the time they were recorded. The distance to the target varies between 10 and 160 meters. Scenario two, three, four, five and six are captured along the coast. The target is the main focus in

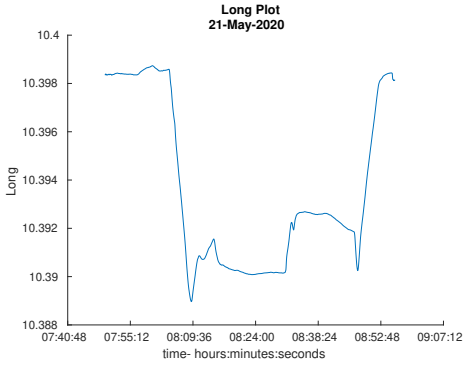
the scene, with the coast line and the island Munkholmen present in the background. In scenario one and seven milliAmpere is following the target in and out of the harbour. This is to give an impression how the disparity map and the two clustering algorithms perform in a confined environment. The parameters are not tuned for such an environment.

8.1 Ground truth

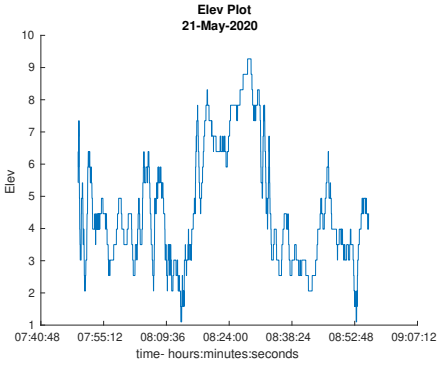
A Garmin eTrex 10 GPS is placed right in front of the wheelhouse of the target boat, recording its positioning once per second. The GPS device has an accuracy of about 10 meters (Luoto et al., 2010). Figure 8.2 shows the recorded GPS data during the experiment.



(a) GPS latitude



(b) GPS longitude



(c) GPS elevation

Figure 8.2: GPS plots

The test scenarios are all captured along the coast, thus no signal-blocking from buildings is expected. During the data collection, there was little to no wind and a smooth sea. The elevation plot in Figure 8.2c shows that the elevation varies between 1 and 10 meters. The varying elevation signal indicates that the GPS signal strength varies throughout the experiment. In Figure G.2 in Appendix G the longitude, latitude and elevation plots are divided into the different scenarios. Inspecting the plots, the GPS has smooth curves for

most of the scenarios. For scenario three, the longitude curve is mostly smooth but is somewhat noisy towards the end. In scenario five and six, more noise is present. This may indicate a weaker GPS signal. For this reason, the GPS accuracy is evaluated by the LiDAR.

The LiDAR's precision compared to the stereo camera makes it suitable to evaluate the GPS. The LiDAR has a range of 100 meters, but due to its sparse output it struggles to consistently detect the target. This is due to the angling of the laser beams. However, when the object is present, the precision is still in centimeters (Kidd, 2017). The GPS's position in the boat does not in general coincide with the reflecting points seen by the LiDAR. Thus, an uncertainty of approximately ± 1 meters in distance is expected from the extracted points of the LiDAR. Due to the uncertainty in the stereo system, discussed in Section 7.1, only meter precision is reasonable to expect.

The LiDAR and the GPS are transformed to the same coordinate system to correlate the depth measurements. The resulting plots for the different scenarios are given in Appendix G. LiDAR data was not recorded for scenario one, thus scenario two to seven are used to evaluate the GPS. Figure 8.3 shows the depth error between the LiDAR and the GPS.

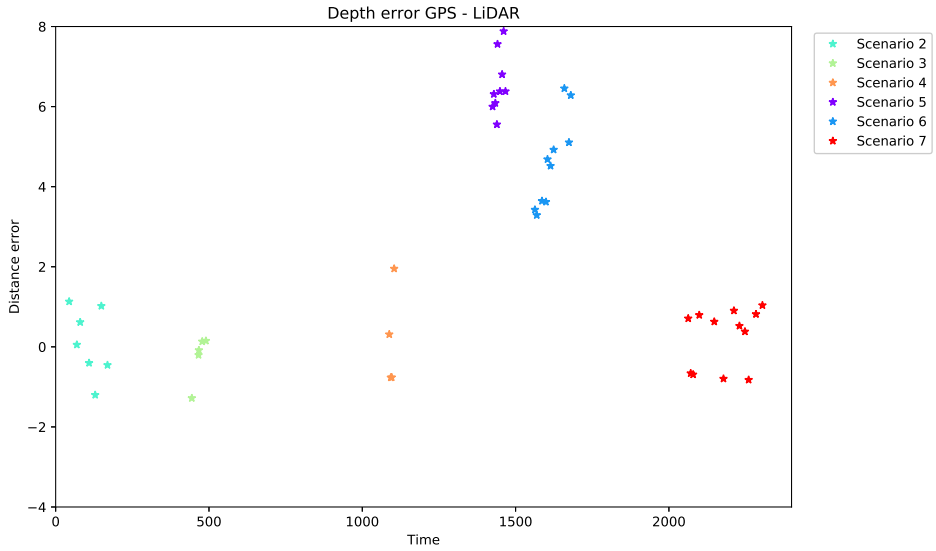


Figure 8.3: Depth error in GPS according to LiDAR

Scenario two, three, four, and seven show small deviations between the LiDAR and the GPS. Based on these four scenarios, the GPS has an average error of ± 0.689 meters. For scenarios five and six, there is a more significant error. On average, the GPS overestimates the depth with 5.537 meters compared with the LiDAR. Together with the noise in the corresponding figures in Appendix G, this indicates a weaker GPS signal during scenario five and six. The accuracy of the LiDAR point cloud is approximately equal in scenario three, five and six. The positioning of the target is similar, thus it is fair to assume that there is a difference in GPS accuracy between the different scenarios.

The average error of scenario five is 6.551 meters while for six it is 4.625. Trusting the LiDAR's high accuracy, the mean error of scenario five and six is subtracted from the GPS. The resulting errors are illustrated in Figure 8.4. The GPS measures are assumed

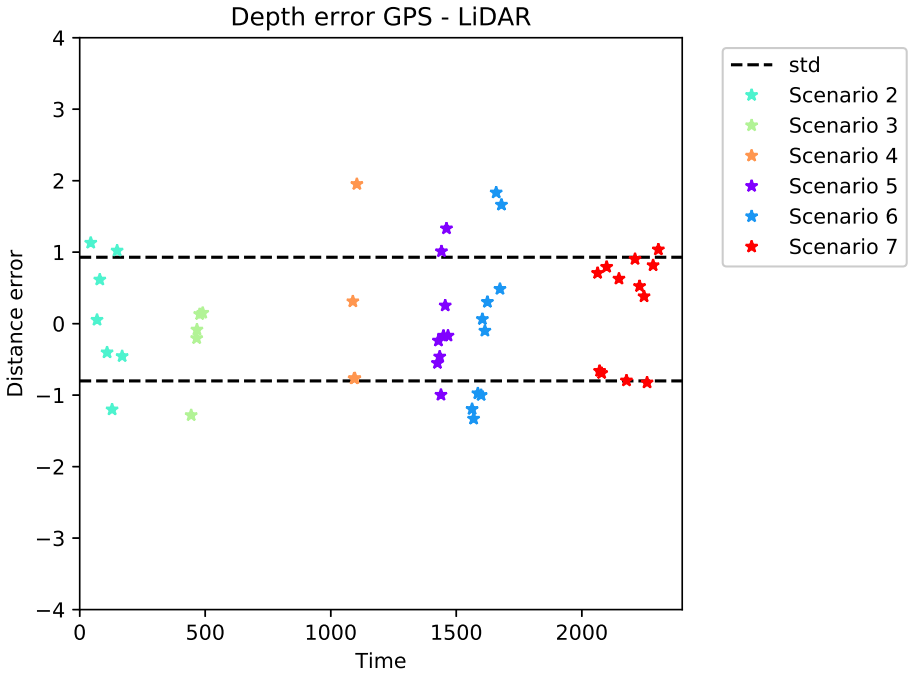


Figure 8.4: Depth error in GPS adjusted according to LiDAR measurements

to include Gaussian distributed noise with mean $\bar{e} = 0.064$, and a standard deviation of $\sigma = 0.865$. The variance is therefore $\sigma^2 = 0.748$. This seems reasonable as the LiDAR is expected to have a standard deviation of around 1. Finally, it should be noted that the error of the last point in scenario four deviates more than the other points in the same scenario. This may indicate that the GPS has a offset from the true position in the end of scenario four.

Both the measurements by the GPS and the LiDAR are transformed into the NED reference frame providing as the ground truth for evaluating the measurements of the stereo camera.

8.2 Results

This section presents the five test scenarios along the coast of Trondheim. The target is captured in a range of different distances to determine the stereo systems accuracy. The targets position is given in NED coordinates, and the distance to the target is calculated relative to milliAmpere. The resulting plots of the two clustering techniques; point cloud

clustering and CNN clustering, are presented for each of the test scenarios. The estimated distance to the target, the resulting error, and the NED position of the targets is given. The position in north and east direction depends on the time, and the plots of the directions are given in Appendix H together with the corresponding errors. In each plot, the ground truth is given in black, and the stereo systems estimates in blue. For all plots, distances are given in meters and time in seconds.

As already mentioned in Chapter 5, there may have been added a constant error to the stereo setup. This will have a high influence on the overall results. However, the comparison between the algorithms, and distance estimates is still valid. From Section 7.1, the stereo system is expected to have a theoretical reprojection error of more than one pixel. For this reason, the theoretical reprojection error of one pixel is given in purple and plotted together with the distance error of the stereo cameras. The expected distance based on the theoretical error is included to illustrate what one pixel error corresponds to in distance for each scenario. The value of the reprojection error is given in Table 7.1, and plotted with the same sign as the error of the depth estimates.

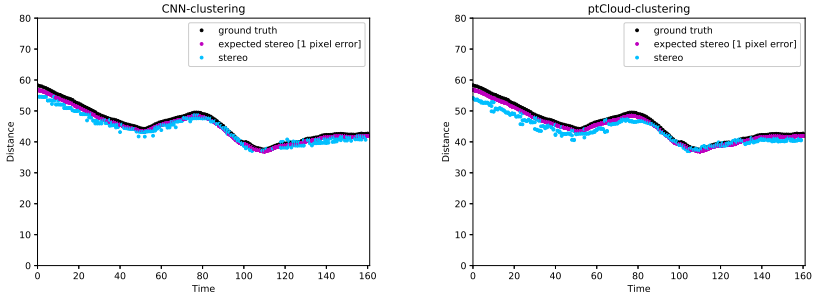
8.2.1 Scenario 2

(a) Start of scenario 2 **(b)** End of scenario 2

Figure 8.5: Scenario 2

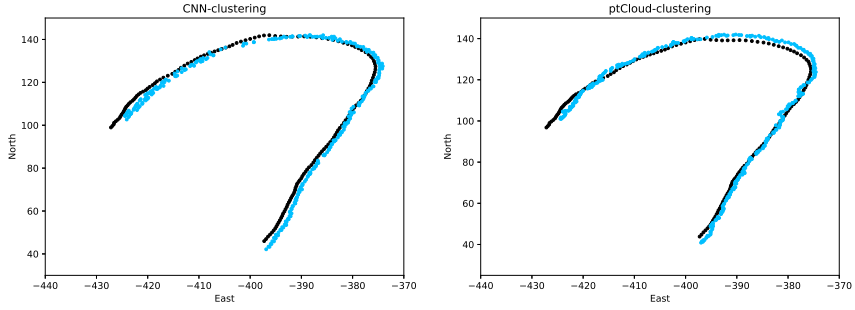
In the beginning of scenario 2, the stereo camera observes the aft of the target. In the middle of the scenario the target makes a turn and places itself perpendicular to Milliamepre, see Figure 8.5. As the GPS is placed in front of the wheelhouse, a larger error is expected in the first half of the scenario.

From the distance estimations in Figure 8.6, both clustering seems quite accurate. The distance is the optimal depth for the stereo system, and the result are therefore expected to be the most satisfactory of all the scenes. However, Figure 8.8 show some deviations from the theoretical error. As expected, the estimates are more accurate in the second half of the scenario.



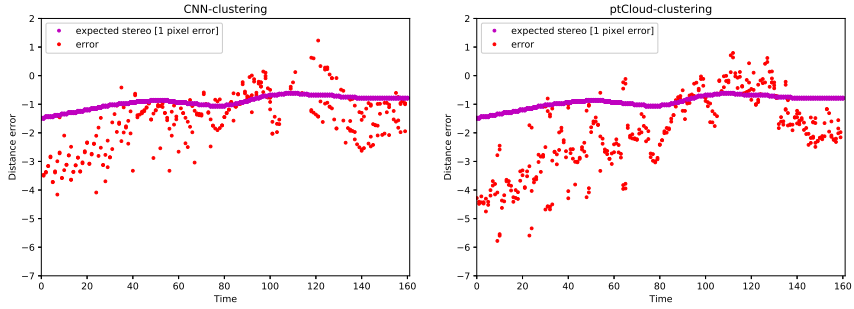
(a) Estimated distance **(b)** Estimated distance

Figure 8.6: Distance estimate of scenario 2



(a) Position in North-East **(b)** Position in North-East

Figure 8.7: Position estimate of scenario 2

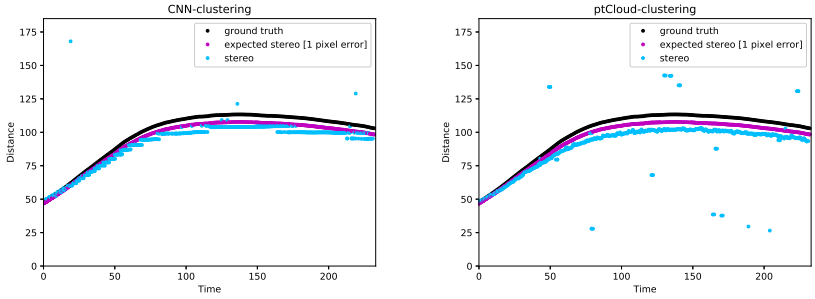


(a) Error of estimated distance **(b)** Error of estimated distance

Figure 8.8: Error of distance estimate of scenario 2

8.2.2 Scenario 3

Figure 8.9: Scenario 3



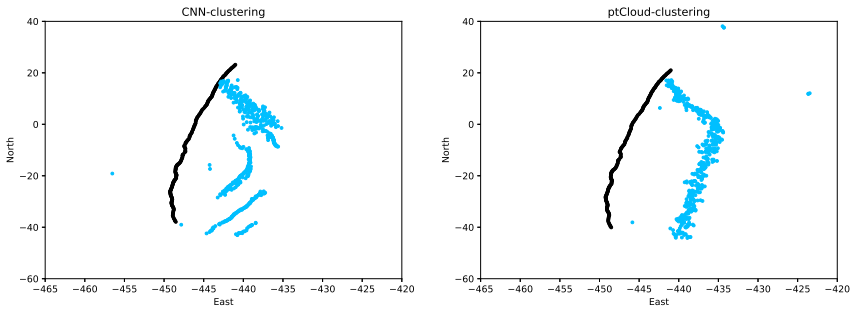
(a) Estimated distance

(b) Estimated distance

Figure 8.10: Distance estimate of scenario 3

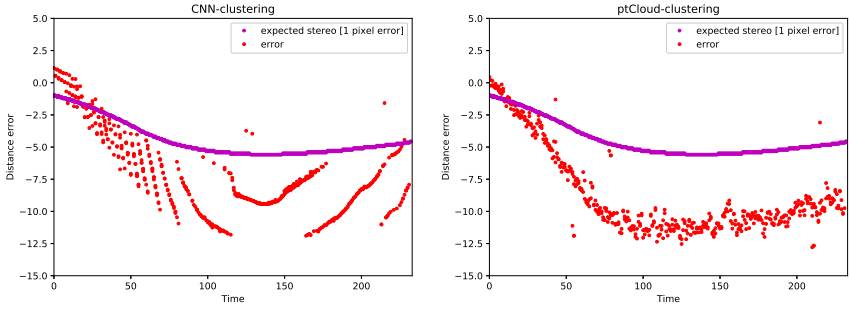
In scenario 3, milliAmpere is moving away from the target, see Figure 8.9. The estimated position in north-east, Figure 8.11, shows clear deviations from the ground truth. The distance to the target is increasing throughout the scenario.

In the early stages of the scene, the stereo estimates follows the theoretically expected error. The deviation increases more than expected as the depth increases, see Figure 8.12. At 100 meters the magnitude of the actual distance error is around twice the theoretical one pixel error. The graph in Figure 8.10 is smoother for ptCloud-clustering than CNN-clustering. The discrete result from CNN-clustering points to a network that struggles to detect the boat consistently in similar scenes.



(a) Position in North-East **(b)** Position in North-East

Figure 8.11: Position estimate of scenario 3



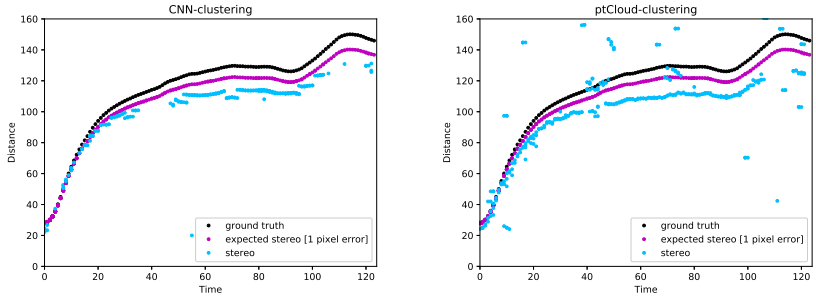
(a) Error of estimated distance **(b)** Error of estimated distance

Figure 8.12: Error of distance estimate of scenario 3

8.2.3 Scenario 4

Figure 8.13: Scenario 4

In scenario 4, the target is moving away from milliAmpere making only the rear end visible for the stereo camera, see Figure 8.13. The stereo camera will therefore have an

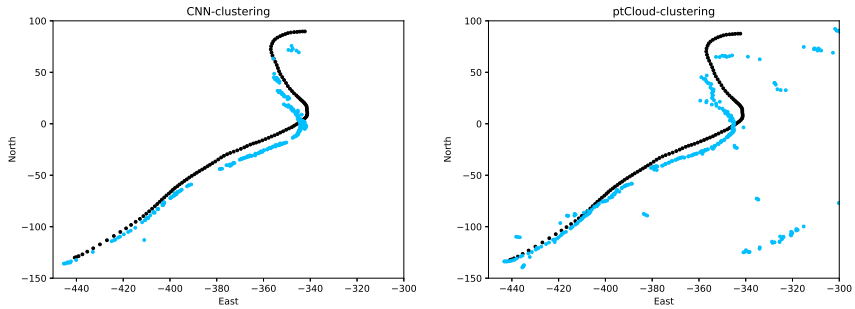


(a) Estimated distance **(b)** Estimated distance

Figure 8.14: Distance estimate of scenario 4

offset from ground truth. The distance to the target is increasing but continues to a distance of 150 meters.

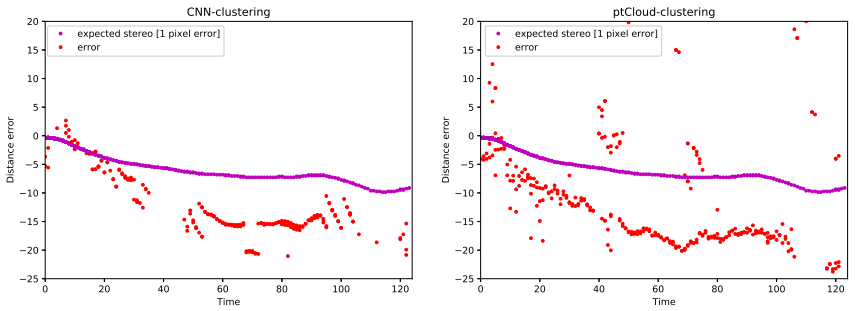
Similar to scenario 3, for distances below 80 meters the estimates are within an acceptable range, see Figure 8.14. At farther distances stronger deviations are present, and the error exceeds 20 meters. During the last twenty second both algorithms fails to extract clusters, which implies the clusters from the disparity map are too small and considered as noise for the algorithms. The estimates follows the shape of the curve, but with an increasing error. From Figure 8.15 CNN-clustering has a lower error, but point cloud clustering has a more consistent detection of the target throughout the scenario. The noise from the point cloud clustering originates from a noisy disparity map. In the last part of the scene there is a significant large error. As mentioned in 8.1, the GPS might have an offset in the measurement towards the end of scene 4.



(a) Position in North-East **(b)** Position in North-East

Figure 8.15: Position estimate of scenario 4

For clarification, for a couple of seconds the target moves out of the stereo cameras field of view. This occurs at position -50 North, -380 east in Figure 8.15, which correspond to time 40 (seconds).



(a) Error of estimated distance

(b) Error of estimated distance

Figure 8.16: Error of distance estimate of scenario 4

8.2.4 Scenario 5

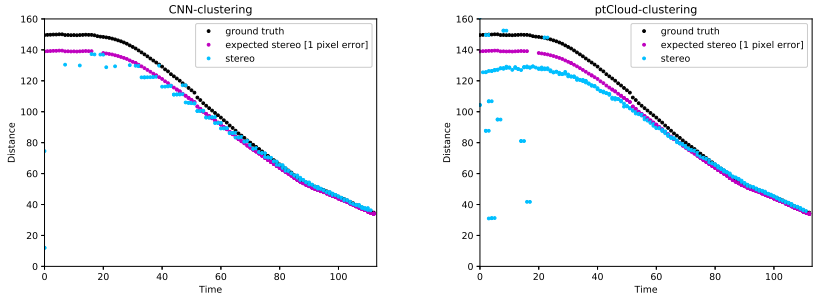
(a) Start of scenario 5

(b) End of scenario 5

Figure 8.17: Scenario 5

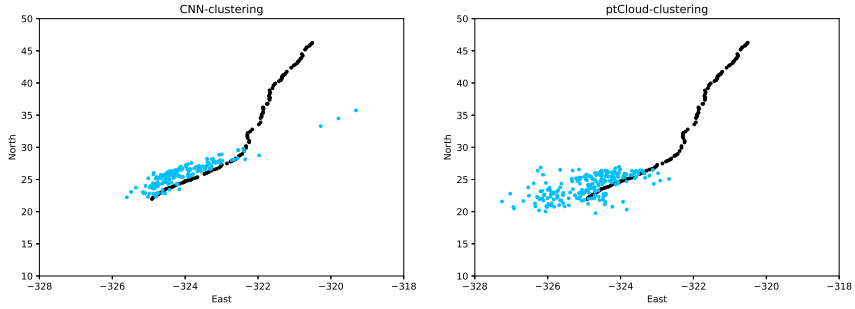
In scenario 5, milliAmpere is moving towards the target. Figure 8.17, shows the target captured at the beginning of the scene with a distance of 157 meters. At this distance CNN-clustering fails to detect the target, while the point cloud clustering actually manages to cluster the boat, see Figure 8.18. Again, the distance estimates follows the shape of the curve. However, the error of the estimated distance is large, from Figure 8.20 it measures up to 25 meters. At distances lower than 90 meters, the system performs better than the estimated theoretical error.

The resulting north-east plots, in Figure 8.19, looks quite odd, but seeing each of the coordinates plotted against time it makes more sense. Referring to Figure H.4 and H.9 in Appendix.



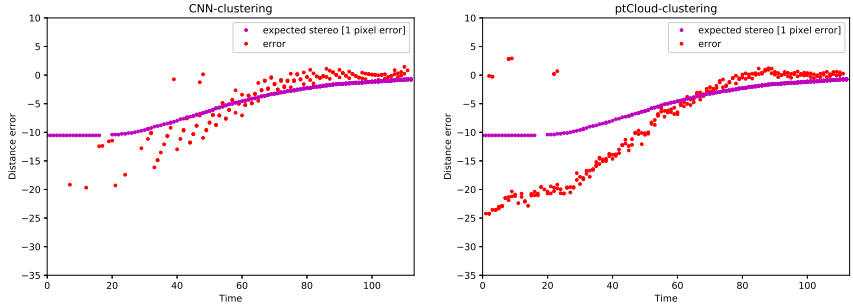
(a) Estimated distance **(b)** Estimated distance

Figure 8.18: Distance estimate of scenario 5



(a) Position in North-East **(b)** Position in North-East

Figure 8.19: Position estimate of scenario 5



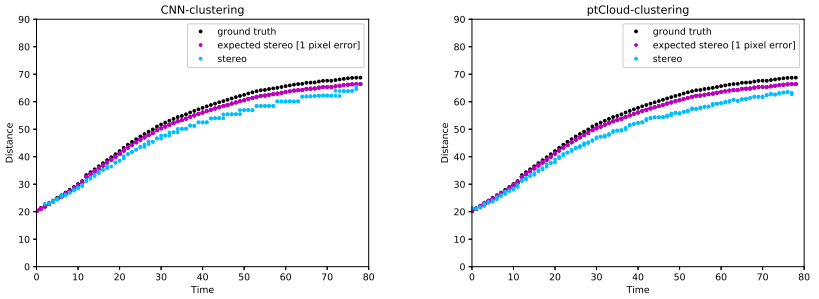
(a) Error of estimated distance **(b)** Error of estimated distance

Figure 8.20: Error of distance estimate of scenario 5

Figure 8.21: Scenario 6

8.2.5 Scenario 6

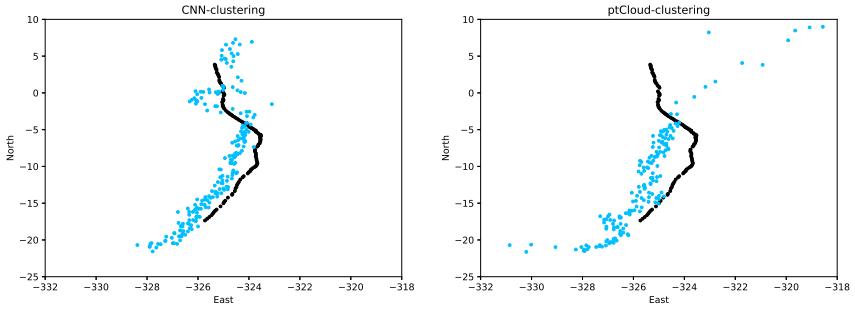
In scenario 6, milliAmpere is moving away from the target. The estimated distances are given in Figure 8.22, and the world coordinates in Figure 8.23. The scenario has similar distances as scenario 2. From Figure 8.24 it appears that the error increases with increasing distance, but with a larger magnitude than in scenario 2. At the end of the scene, the distance error actually decreases, this yields for distances above 60 meters. Overall, the point cloud algorithm serves a smooth and more dense error graph than the CNN-clustering.



(a) Estimated distance

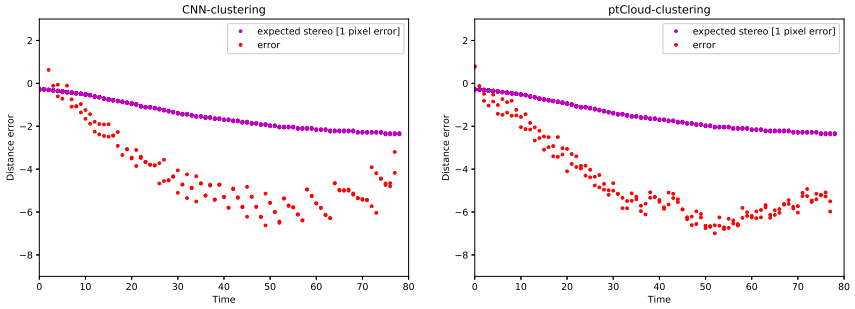
(b) Estimated distance

Figure 8.22: Distance estimate of scenario 6



(a) Position in North-East **(b)** Position in North-East

Figure 8.23: Position estimate of scenario 6



(a) Error of estimated distance **(b)** Error of estimated distance

Figure 8.24: Error of distance estimate of scenario 6

8.2.6 Application inside harbour

Scenario 1 and 7 are recorded while driving out of and into the harbour, respectively. The ground truth is thereby not the only true detection to be clustered. The estimated position of the detected objects is shown in Figure 8.26, in NED coordinate. The positions are plotted on a map of the test area to show where the detection originates from. There is a significant difference between the detections of the two clustering approaches. Due to YOLO being trained to detect only boats, CNN based clustering detects far less objects than point cloud clustering.

When navigating in an confined environment, it is important that all objects are detected to avoid collisions and getting stranded. Training the network on objects present in a harbour would make CNN based clustering more suitable for an environment like this. However, a segmentation-based method might be preferred in an environment like this. Neither the clustering or the disparity map is not tuned for a confined environment. The intended use of the algorithms are at open sea, but the harbour scenarios are included to show how they perform in a different environment. This shows that the system is sensitive to the chosen operational environment.

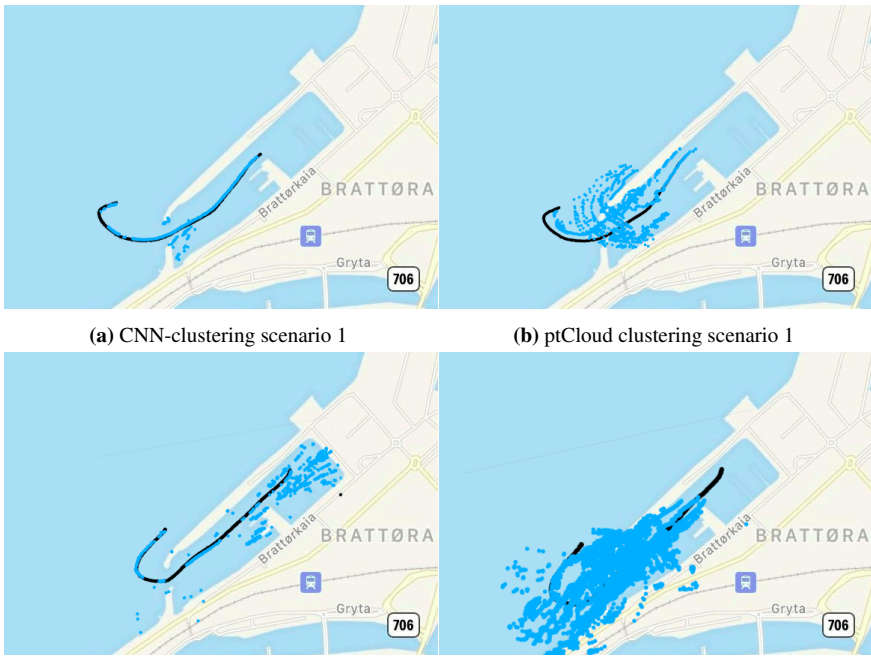
(a) Start of scenario 1

(b) End of scenario 1

(c) Start of scenario 7

(d) End of scenario 7

Figure 8.25: Recorded data from harbour



(a) CNN-clustering scenario 1

(b) ptCloud clustering scenario 1

(c) CNN -clustering scenario 7

(d) ptCloud-clustering scenario 7

Figure 8.26: Estimated position of detected objects in harbour

8.2.7 Comparing detection techniques

The five scenarios are plotted together, with the result of point cloud clustering and CNN-clustering in the same plot in Figure 8.27. The most noticeable is the sparse result of CNN-

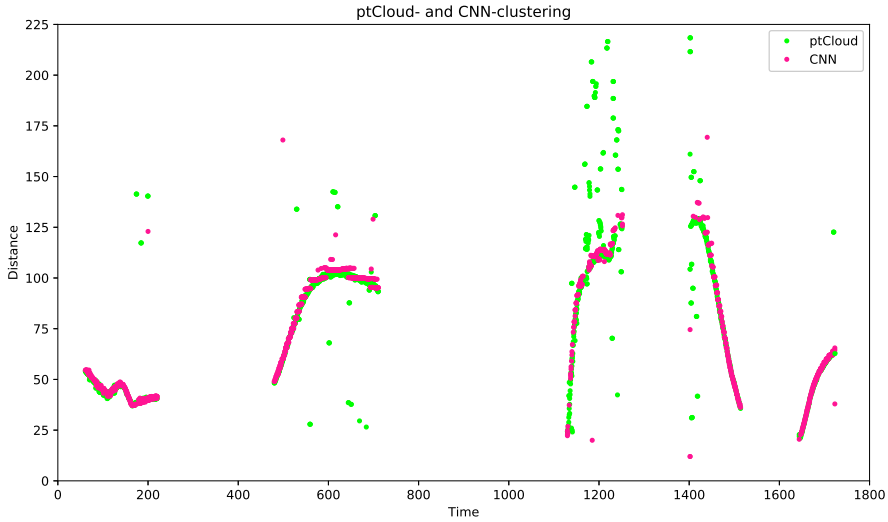


Figure 8.27: Depths of ptCloud and CNN clustering

clustering compared to point cloud clustering. This is especially seen in scene three, four and five as the distances increases in a shorter period of time. The estimated depth of the two approaches are approximately equal. This is expected as they are both implemented using the same disparity map algorithm and stereo parameters. Overall, the point cloud clustering has a denser distance estimate with more noise present. At great distances the stereo cameras estimations has a higher error. Significant deviations are present at distances greater than 75 meters. This is expected as the reprojection error of the stereo camera has a larger influence at greater distances.

8.3 Discussion

This section discusses the results found in the previous section, and assesses the overall performance of the stereo system. The experimental error in the distance estimates turned out to be higher than expected. It can be caused by several reasons; the parameters of the clustering, the disparity map or the calibration.

8.3.1 Clustering techniques

The detection of the target and its estimated depth is highly dependent on the algorithms and parameters used in the stereo system. For CNN based clustering the reliability of the network is of high importance. The threshold was set to avoid miss-classifications of

the same object in consecutive frames. The resulting plots shows few false positives, but increased the number of false negatives. This caused a sparser output compared to point cloud clustering. As a result of this, it seems like an even lower YOLO-threshold may be favourable.

Decreasing the precision further will results in more detections. However, with the given weights, the network tends to have FPs even for an YOLO-threshold approximately equal to 0. From the confusion matrix (7.8), a decrease in precision for the given weights gives no noticeable increase in TP. This can also be seen from Figure 7.10 in Section 7.3.2. The technique is limited to the quality of the network, and based on the results the networks performance can benefit from further training.

For the localization based on the YOLO-bounding boxes, the centroid and median value gave a satisfying result. The distance estimates even seems to be more correct than the distances directly extracted from the point cloud. The inequalities in the depth estimates between the two algorithms is most likely due to the network's tendency to only detect parts of the boat. For the given test scenarios, the wrong positioning of the bounding boxes does not have any noticeable impact on the results. However, this should be noted as it may cause errors in different scenarios.

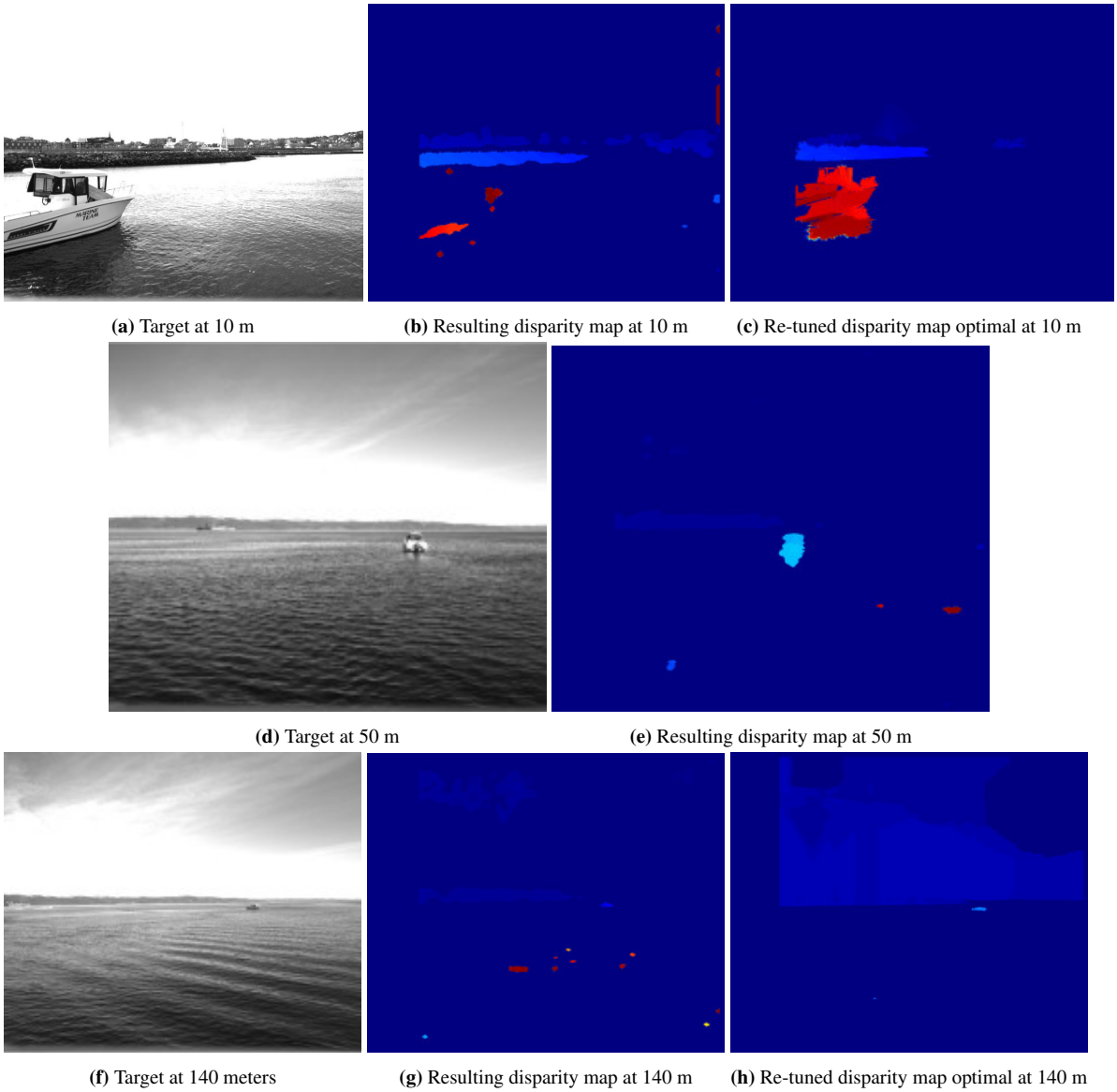
Point cloud clustering produces a dense output, but more false detections are present. This is a direct consequence of the chosen parameters. The wide range of distances in the scenarios impose challenges for the selection of parameters. Setting a higher value for the minimum number of points in a cluster would remove a lot of the noise. On the other hand, for distances above 110 meters the target will be discarded as noise. Alternatively, an increase in the number of accumulations or decreasing the value of the standard deviation multiplier in the filtering, will discard some more noise. However, the noise is randomly distributed and can thus be accounted for by standard Markovian models in a tracking algorithm.

8.3.2 Disparity map

The results are directly dependent on the parameter values of the disparity map. The chosen parameter of the disparity map are optimal at approximately 50 meters, while the distance to the target varies between 10 and 160 meters. Figure 8.28 shows examples of the resulting disparity maps at short, medium and far range distances. Figure 8.28b, 8.28e and 8.28g illustrated the original parameters set for point cloud clustering. At 140 meters noise from the sea produces bigger clusters than the target. This causes the noise and lack of detection towards the end of scenario 4 in Figure 8.14. In the re-tuned disparity map in Figure 8.28h noise is removed by reducing the maximum disparity. This will segment out points at short distances. The minimum required points in a cluster can thus be decreased, making object detection optimal for far distances.

At 10 meters the resulting disparity map divides the boat into different segments. This is shown in Figure 8.26d, where both the target and the pier is segmented into several parts, creating multiple detections of the same object. This also causes problems for CNN clustering, as it takes the median value of the disparities within a bounding box. When the object appears sparse, the depth is estimated to infinity, e.g. no target is localized.

Considering the range of operating distances, it was hard to find optimal values. From the results, the chosen disparity parameters performed better than expected considering



(a) Target at 10 m **(b)** Resulting disparity map at 10 m **(c)** Re-tuned disparity map optimal at 10 m

(d) Target at 50 m **(e)** Resulting disparity map at 50 m

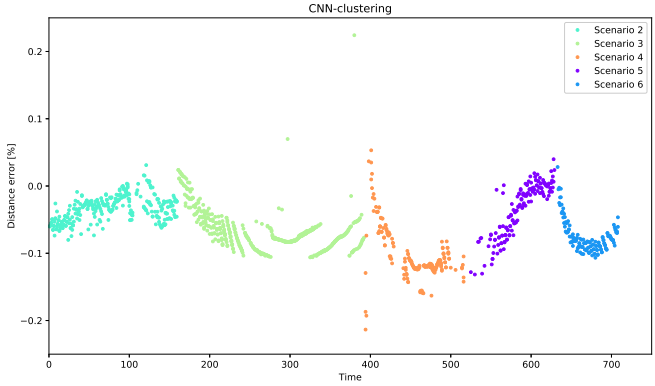
(f) Target at 140 meters **(g)** Resulting disparity map at 140 m **(h)** Re-tuned disparity map optimal at 140 m

Figure 8.28: Disparity map for targets at different distances

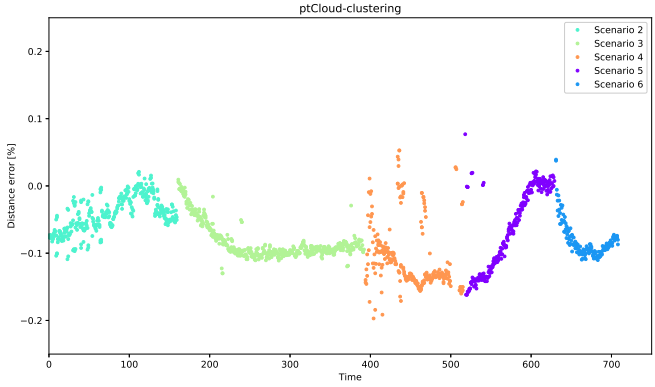
the variations in the scenes. However, as shown throughout this section more favourable parameters exist for the different operating ranges. Having a disparity map that is optimal for the distance it is used on improves the result of both the detection techniques. A solution could be to dynamically set the disparity map parameters depending on the speed of the vessel. Another option, is running parallel disparity maps optimal for short, medium and far range distances.

8.3.3 Error of the estimated distances

The mean percentage error of the two clustering techniques is plotted in Figure 8.29. The error is calculated from the ground truth. The CNN-clustering yields an average absolute error percentage of 0.069, and the point cloud clustering 0.076. These calculations are based on a adjusted ground truth, combining the GPS measurements with the LiDAR. In addition, 2 meters are removed from scenario 4 and the beginning of scenario 2 to account for the GPS's position in the boat. From the Figure 8.29, the magnitude of the error varies



(a) Percentage error of estimated distance from CNN-clustering



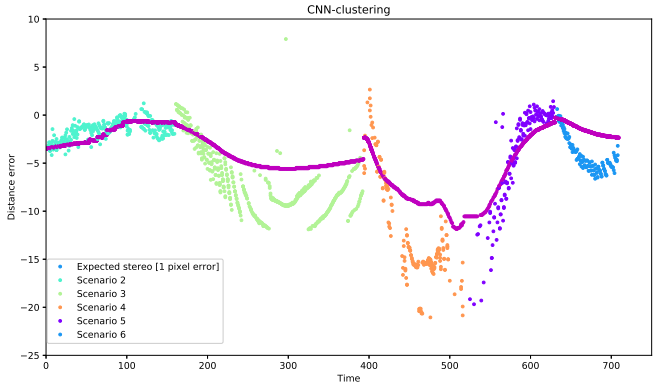
(b) Percentage error of estimated distance from ptCloud-clustering

Figure 8.29: Percentage error of estimated distance from the ground truth for all scenarios

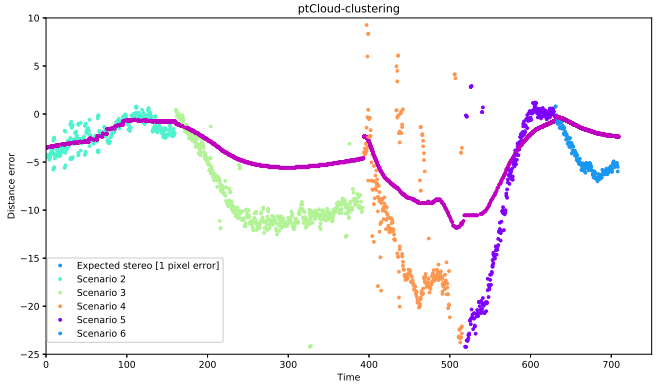
between the scenarios. The percentage error is three times as high as the error from the test scenes in Chapter 5.

It is worth noticing that there seems to be a constant offset, a bias. This is probably due to errors in the estimation of the focal length or the baseline. Both directly influences the depth estimates (2.10). A constant error can also have occurred due to the transformation between the camera and the GPS on milliAmpere being measured by hand.

8.3.4 Reprojection error



(a) CNN-clustering



(b) ptCloud-clustering

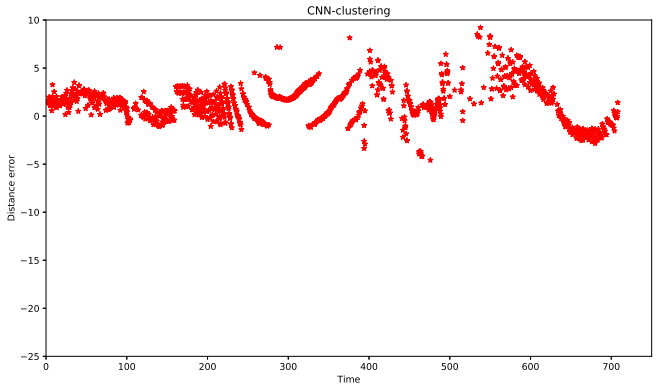
Figure 8.30: Error of all scenario and expected value when there is a pixel error of 1

In Figure 8.30, the error of all scenarios are given with the expected value when one pixel error is present. This confirms that the theoretical reprojection error is too optimistic. To find the true reprojection error for the system, equation (8.1) is minimized.

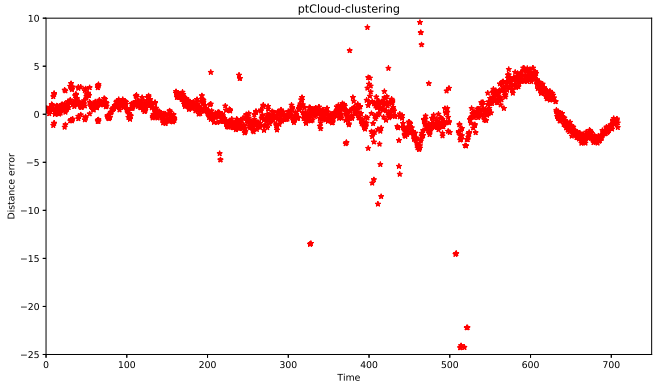
$$\boldsymbol{\mu}(e(dist)) = \frac{1}{N}\sum(dist_{stereo} - (dist_{GT} + e(dist))) \tag{8.1}$$

$dist_{stereo}$ is the estimated distance from the stereo camera, $dist_{GT}$ is the ground truth distance, and $e(dist)$ is the reprojection error given in pixels. This gave the CNN-clustering a reprojection error of 1.588 pixels, and the point cloud clustering 1.988 pixels. One may argue that the expected value should be zero, however this confirms that there is a constant offset in the stereo system.

Figure 8.31a and 8.31b shows how the estimated distance from the stereo camera deviates from the given reprojection error. Point cloud clustering yields a higher reprojection



(a) Difference between estimated distance from CNN-clustering and expected distance when reprojection error is of 1.583 pixels



(b) Difference between estimated distance from ptCloud-clustering and expected distance when reprojection error is of 1.988 pixels

Figure 8.31: Difference between estimated distance and expected distance for all scenarios

error. This is due to detections of noise present in the disparity map. This is not caused by wrong detections of the target, and could thus be removed. However, it is problematic to separate noise from wrong estimations.

It should also be noted how much Scenario 5 and 6 differs from the other. Considering the distance to the target, a better performance in scenario 6 was expected. Especially, as the error is low for scenario two, which includes the same range of distances. In Scenario 5 the error in Figure 8.31 is positive, which deviates from all other results. As the GPS most likely has a weaker signal at these scenarios, the adjusted ground truth using the LiDAR might not be as accurate as desired.

8.3.5 Uncertainty in the stereo system

The standard deviation of the Figures 8.31a and 8.31b, measures the expected uncertainties of the system. With a mean of 1.583 pixels, the standard deviation of CNN-clustering is calculated to 3.882 meters. For the ptCloud-clustering the standard deviation is 6.921 meters calculated from a mean value of 1.988 pixels.

The standard deviation of ptCloud-clustering is of significant size. By removing scenario 5 and 6, the standard deviation is reduced to 2.668 meters. The distribution of points may indicate a Gaussian distribution. Assuming this, ptCloud-clustering has a variance of 1.633 meters. Removing scenario 5 and 6 impose no significant changes to the measured standard deviation of CNN-clustering. The depth error, shown in Table 8.1, indicates the

Real world-depth	Disparity	Depth error CNN-clustering	Depth error ptCloud-clustering
10 m	229.3	±0.070m	±0.087 m
20 m	114.7	±0.277m	-±.348 m
30 m	76.5	±0.621m	±0.779 m
40 m	57.3	±1.107m	±1.390 m
50 m	45.9	±1.727m	±2.169 m
60 m	38.2	±2.488m	±3.125 m
70 m	32.8	±3.384m	±4.250 m
80 m	28.7	±4.423m	±5.554 m
90 m	25.5	±5.605m	±7.040 m
100 m	22.9	±6.922m	±8.694 m
110 m	20.9	±8.350m	±10.487 m
120 m	19.1	±9.952m	±12.499 m
130 m	17.7	±11.654m	±14.636 m
140 m	16.4	±13.498m	±16.952 m
150 m	15.3	±15.509m	±19.476 m

Table 8.1: Expected depth error for CNN-clustering (1.583 pixels) and ptCloud-clustering (1.988)

expected reprojection error under the assumption that the disparity error is 1.583 and 1.988 pixel.

8.3.6 Limitations of the stereo system

The experimental measured error and standard deviation turned out to be higher than in the theoretical consideration where more optimistic estimations about the possible setup and measurement deviations was made. It can be caused by several reasons which finally accumulate to a higher error. This includes higher errors in the estimation of the baseline and the relative orientation of the cameras but also higher errors in the estimation of the matching position of corresponding points.

There is a constant offset in the distance estimates. The constant offset indicates errors in the estimates of the stereo parameters. This will influence both the reconstruction of world points and the rectification which directly affects the stereo matching.

The results indicates range limitations. In general, the disparity map of objects at far distances produces a sparse cluster of points. Figure 8.28h shows that even for an optimal disparity map at 140 meters, the target appears in relatively few pixels. At 140 meters an object will have a displacement of 16.4 pixels. Due to uncertainties in the stereo matching, the stereo system is likely to have range limit of approximately 200 meters. This corresponds to a displacement of only 11 pixels. To have an operational system at such great distances, a tuning of the disparity map and the point cloud clustering algorithm is imperative. With the stereo systems calculated reprojection error an object at 200 meters yields a depth error of almost 80 meters. Therefore, the baseline must be widened to reduce the slope of the depth error, based on the reprojection error and baseline (Figure 7.1b).

8.3.7 Overall performance

Some error in the estimated distance to the target is tolerable when the uncertainty of the system is known. From the ground truth, the reprojection error was calculated to 1.583 and 1.988 pixels for CNN-clustering and ptCloud-clustering, respectively. With additional testing, the uncertainty of the system can be determined more accurately.

Two operational clustering algorithms has been implemented, as well as a stereo matching procedure. The range of distances imposes challenges for setting optimal parameters for all scenarios. The result of point cloud clustering is influenced by noise present in the disparity map. While, for CNN-clustering the result is dependent on the quality of the network. Even though miss-classifications and lack of detections occurred, the target was detected and localized for all scenarios. The CNN based clustering procedure is robust against noise in the disparity map. Point cloud clustering is more prone to noise in the disparity map, but gave a more dense and consistent result.

Scenario 1 and 7 show how sensitive the behavior of the system is to changes in the environment. For the system to produce beneficial results in these scenarios, an adjustment of the parameters is necessary. The performance of the system is dependent on set parameters and the operational environment. Using different parameters of the algorithms at different distances, could also help improve the overall result. However, the utilized parameters provided a reasonable result considering the wide range of distances in the scenarios. Overall, the parameters extracts the target and estimates the distance with a offset. With further testing of the system using a more reliable ground truth a lower uncertainty is expected.

Chapter 9

Conclusion and future work

9.1 Conclusion

A stereo setup for an autonomous ferry was introduced, calibrated, implemented and tested. A far range extrinsic calibration procedure was proposed, and tested on scenes of different depths. The LiDAR was used as the ground truth to confirm that extrinsic calibration highly influence the accuracy of the reconstructed world points. The findings showed that a careful selection of the scene is of higher importance than calibrating at the operating distance. A scene where salient points are available in the cameras entire field of view will improve the calibration accuracy.

A ROS-based software architecture for object detection was implemented and tested in a marine environment. The disparity map was implemented using a correlation based correspondence algorithm, and Fast Global Image Smoothing Based on Weighted Least Squares. This provided a robust foundation for clustering. Two different techniques were utilized for object detection. A convolution neural network was applied for classification, and used in combination with the disparity map to extract 3D positions of objects. The method is robust against noise in the disparity map, but appeared to be partially inconsistent in the estimates. However, the technique is limited to the quality of the network. The alternative detection method based on hierarchical clustering using Euclidean distance made consistent detections, but was more prone to noise in the disparity map. The noise can be accounted for by standard Markovian models in a tracking algorithm.

The system was tested on the autonomous ferry milliAmpere, and the targets was detected and its localization given in the common world frame. With a stereo configuration optimal for 50 meters, the stereo system was found to have a reprojection error of 1.583 and 1.988 pixels, for CNN- and ptCloud-clustering respectively. The average absolute error for the systems distance estimates is 7%. With additional testing, the uncertainty of the system can be determined more accurately. Based on the constant bias, the error is expected to be adjusted to a more acceptable range. While the implemented system shows potential, it is

still not clear that the detection performance is good enough to rely on these methods in an autonomous collision avoidance system.

9.2 Future Work

The thesis has experimented with a stereo configuration for use on an autonomous ferry. The work has laid the foundation for using stereo camera for visual vessel detection on an unmanned surface vehicles in general. The system was tested in the operation environment, in and close to Trondheim Harbor, but there are still several issues to be solved before the proposed stereo vision solution can be used as a part of a sensor fusion system. Therefore, the following tasks are suggested to continue the work of this thesis.

- Due to COVID-19 and bad weather conditions opportunities for testing was limited. More tests of the system needs to be performed to find a more exact estimate of the uncertainty. In the test scenarios presented, the ground truth originates from a GPS corrected by LiDAR data. A more reliable ground truth would be provided by e.g, a GPS using RTK.

- The stereo camera system should be properly mounted and integrated on milliAmpere. This allows for a calibration for finding the rigid-body transformation between the stereo camera and the BODY frame of the vessel. This will improve the accuracy of the system. When a proper calibration of the stereo camera and milliAmpere's BODY frame is performed, the resulting 3D coordinate frames should be implemented using the ROS-library tf. This will make the information about the coordinate frames available to all ROS components.

- The physical constraints of the cameras needs to be taken into consideration. The operating temperature is above 0 degrees and they are not waterproof. The stereo setup is also sensitive to changes in temperature, vibrations and wind, causing the accuracy of the extrinsic stereo calibration to fluctuate over time.

- As already mentioned, more favourable disparity maps parameters exists for the different operating distances. Looking into dynamically set disparity maps depending on the speed, or running multiple maps in parallel could improve the result of both the detection techniques.

- The real-time capabilities of the system must be assessed. The running time of the system is imperative for the reliability and usefulness of the output data. YOLO is running on the GPU, using CUDA to reduce the running time. The CUDA toolkit can be useful throughout the system, especially for the point cloud clustering and stereo matching. For the overall system all the image processing will benefit of being in the same package. Implementing the nodes as nodelets will minimize memory usage, running multiple algorithms in the same process with zero-copy transport between algorithms.

- Comparing different extrinsic stereo parameters in a marine environment would of interest. The stereo camera parameters are set in the Spinnaker driver. As the sce-

narios were recorded using bagfiles, a dynamic change of the parameters on the captured data was not performed.

- Because of the constant bias, both the internal and external parameters of the stereo camera should be re-calibrated. When performing the stereo calibration the data should be normalized when performing linear triangulation. This will improve the numerical estimates in DLT. When evaluating the parameters the main focus was on depth. A further evaluation of the influencing factors of, and the accuracy of reconstructed x- and y-coordinates should be performed.

- The accuracy of the stereo camera is dependent on the estimations of the extrinsic parameters, thus changes in the configurations can lead wrongly reconstructed 3D points. Marita et al. (2006) proposed an on-line calibration method for monitoring the stability of the relative extrinsic parameters. The method is suitable for outdoor far range stereo 3D reconstruction applications. Accounting for external factors influencing the physical system would also be beneficial for the stereo camera mounted on milliAmpere.

- Widening the baseline and imposing a global method for stereo matching would increase the accuracy of the system. If a global method is used, then running time must be taken into consideration. Generally, global methods require more computational power than local methods.

- The possibility of fusing the stereo camera with other sensors should be examined. If data from different sensors are combined it will be possible to achieve less uncertainty than for a pure stereo camera system.

Bibliography

Abdullah, Sukarnur Che, e.a., 2019. Stereo vision for visual object tracking and distance measurement assessment. Journal of Mechanical Engineering 16, 121–134.

Benjdira, B., Khursheed, T., Koubaa, A., Ammar, A., Ouni, K., 2019. Car detection using unmanned aerial vehicles: Comparison between faster R-CNN and YOLOv3, in: 2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS), pp. 1–6.

Biber, P., Straßer, W., 2003. The normal distributions transform: A new approach to laser scan matching, in: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), pp. 2743–2748.

Christensen, H.I., Bowyer, K., Bunke, H., 1993. Active Robot Vision: camera heads, model based navigation and reactive control. volume 6. World Scientific.

Curtin, D.P., 2011. Understanding the Baseline. URL: `http://www.shortcourses.com/stereo/stereo3-14.html`.

Fooladgar, F., Samavi, S., Soroushmehr, S.M.R., Shirani, S., 2013. Geometrical analysis of localization error in stereo vision systems. IEEE Sensors Journal 13, 4236 – 4246.

Fossen, T.I., 2011. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons.

G., C., Stephens, M., 1988. A combined corner and edge detector. Alvey vision conference 15, 10–5244.

Golub, G.H., Van Loan, C.F., 2013. Matrix computations. volume 4th. JHU press.

Grini, S.V., 2019. Object Detection in Maritime Environments. Master's thesis. NTNU.

Hartley, R., Zisserman, A., 2004. Multiple View Geometry in computer vision, 2nd edition. Cambridge University Press.

Hirschmüller, H., 2005. 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), pp. 807–814 vol. 2.

Hirschmüller, H., 2011. Semi-global matching-motivation, developments and applications. Photogrammetric Week 11 , 173–184.

Hirschmüller, H., Gehrig, S., 2009. Stereo matching in the presence of sub-pixel calibration errors, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE. pp. 437–444.

Huber, P.J., 1985. Projection pursuit. The annals of Statistics , 435–475.

Kidd, J.R., 2017. Performance evaluation of the Velodyne VLP-16 system for surface feature surveying .

Konolige, K., 1998. Small vision systems: Hardware and implementation, in: Robotics research. Springer, pp. 203–212.

Krenker, A., Bešter, J., Kos, A., 2011. Introduction to the artificial neural networks, in: Suzuki, K. (Ed.), Artificial Neural Networks. IntechOpen, Rijeka. chapter 1.

Kukelova, Z., Martin, B., Pajdla, T., 2008. Polynomial eigenvalue solutions to the 5-PT and 6-PT relative pose problems. BMVC 2, 138–156.

Li, L., Zhang, S., Wu, J., 2019. Efficient object detection framework and hardware architecture for remote sensing images. Remote Sensing 11, 2376.

Li, Y., Min, D., Do, M.N., Lu, J., 2016. Fast guided global interpolation for depth and motion. European Conference on Computer Vision , 717–733.

Liu, Y., Aggarwal, J., 2005. Local and Global Stereo Methods. chapter 3.12. pp. 297–308.

Luoto, M., Marmion, M., Hjort, J., 2010. Assessing spatial uncertainty in predictive geomorphological mapping: A multi-modelling approach. Computers & Geosciences 36, 355–361.

Magnusson, M., 2009. The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection. Ph.D. thesis. Örebro University, School of Science and Technology.

Marita, T., Oniga, F., Nedevschi, S., Graf, T., Schmidt, R., 2006. Camera calibration method for far range stereovision sensors used in vehicles. IEEE Intelligent Vehicles Symposium , 356 – 363.

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

Min, D., Choi, S., Lu, J., Ham, B., Sohn, K., Do, M.N., 2014. Fast global image smoothing based on weighted least squares. IEEE Transactions on Image Processing 23, 5638–5653.

Nguyen, A., Le, B., 2013. 3d point cloud segmentation: A survey, in: 2013 6th IEEE conference on robotics, automation and mechatronics (RAM), IEEE. pp. 225–230.

Nguyen, P.H., Ahn, C.W., 2019. Stereo matching methods for imperfectly rectified stereo images. Symmetry 11, 570.

Ødegård Olsen, T., 2020. Stereo vision using local methods for autonomous ferry. 5th year specialization project at NTNU.

O'Shea, K., Nash, R., 2015. An introduction to convolutional neural networks. CoRR abs/1511.08458. URL: http://arxiv.org/abs/1511.08458, arXiv:1511.08458.

Park, K., Kim, S., Sohn, K., 2018. High-precision depth estimation with the 3d lidar and stereo fusion, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2156–2163.

Pendleton, S., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y., Rus, D., Ang, M., 2017. Perception, Planning, Control, and Coordination for Autonomous Vehicles. Machines 5, 6. URL: http://www.mdpi.com/2075-1702/5/1/6, doi:10.3390/machines5010006.

Praveen, S., 2019. Efficient depth estimation using sparse stereo-vision with other perception techniques, in: Advanced Image and Video Coding. IntechOpen.

Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R.C., Ng, A.Y., 2009. ROS: an open-source robot operating system, in: IEEE International Conference on Robotics and Automation.

Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A., 2015. You only look once: Unified, real-time object detection. CoRR abs/1506.02640. URL: http://arxiv.org/abs/1506.02640, arXiv:1506.02640.

Redmon, J., Farhadi, A., 2018. YOLOv3: An incremental improvement. CoRR abs/1804.02767. URL: http://arxiv.org/abs/1804.02767, arXiv:1804.02767.

robotics, O., . *Spinnaker SDK driver - ROS wiki* . http://wiki.ros.org/spinnaker_sdk_camera_driver. Open Source Robotics Foundation.

Rusu, R.B., Cousins, S., 2011. 3D is here: Point Cloud Library (PCL), in: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China. pp. 1–4.

Sasiadek, J.Z., Walker, M.J., 2019. Achievable stereo vision depth accuracy with changing camera baseline. 24th International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland , 152–157.

Schiller, I., Beder, C., Koch, R., 2012. Calibration of a PMD-camera using a planar calibration pattern together with a multi-camera setup. ISPRS 37.

Shi, J., Tomasi, C., 1994. Good features to track. IEEE conference on computer vision and pattern recognition 2, 593–600.

Shin, B., Mou, X., Mou, W., 2018. Vision-based navigation of an unmanned surface vehicle with object detection and tracking abilities. Machine Vision and Applications 29, 95–112.

Strecha, C., Von Hansen, W., Van Gool, L., Fua, P., Thoennessen, U., 2008. On benchmarking camera calibration and multi-view stereo for high resolution imagery. IEEE Conference on Computer Vision and Pattern Recognition , 1 – 8.

Templeton, B., 2013. Cameras or Lasers? URL: https://www.templetons.com/brad/robocars/cameras-lasers.html.

Thaher, R.H., Hussein, Z.K., 2014. Stereo vision distance estimation employing SAD with Canny edge detector. International Journal of Computer Applications 107.

Theimann, L., 2020. Comparison of depth information from stereo camera and lidar. 5th year specialization project at NTNU.

Tomasi, C., Kanade, T., 1991. Detection and tracking of point features. International Journal of Computer Vision .

Torr, P.H.S., Zisserman, A., 2000. MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding 78, 138–156.

Velodyne, 2019. Velodyne LiDAR Puck. Technical report.

Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q., 2019. Pseudo-lidar from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition URL: http://arxiv.org/abs/1812.07179.

Wang, Y., Lai, Z., Huang, G., Wang, B.H., van der Maaten, L., Campbell, M., Weinberger, K.Q., 2019. Anytime stereo image depth estimation on mobile devices, in: 2019 International Conference on Robotics and Automation (ICRA), pp. 5893–5900.

Wang, Y., Wang, X., Yin, L., 2019. Estimation of extrinsic parameters for dynamic binocular stereo vision using unknown-sized rectangle images. Review of Scientific Instruments 90, 065108.

Warren, M., McKinnon, D., , Upcroft, B., 2013. Online calibration of stereo rigs for long-term autonomy. IEEE International Conference on Robotics and Automation 37, 3692–3698.

Yu, S., Zhu, R., Yu, L., Ai, W., 2018. Effect of checkerboard on the accuracy of camera calibration. Pacific Rim Conference on Multimedia , 619–629.

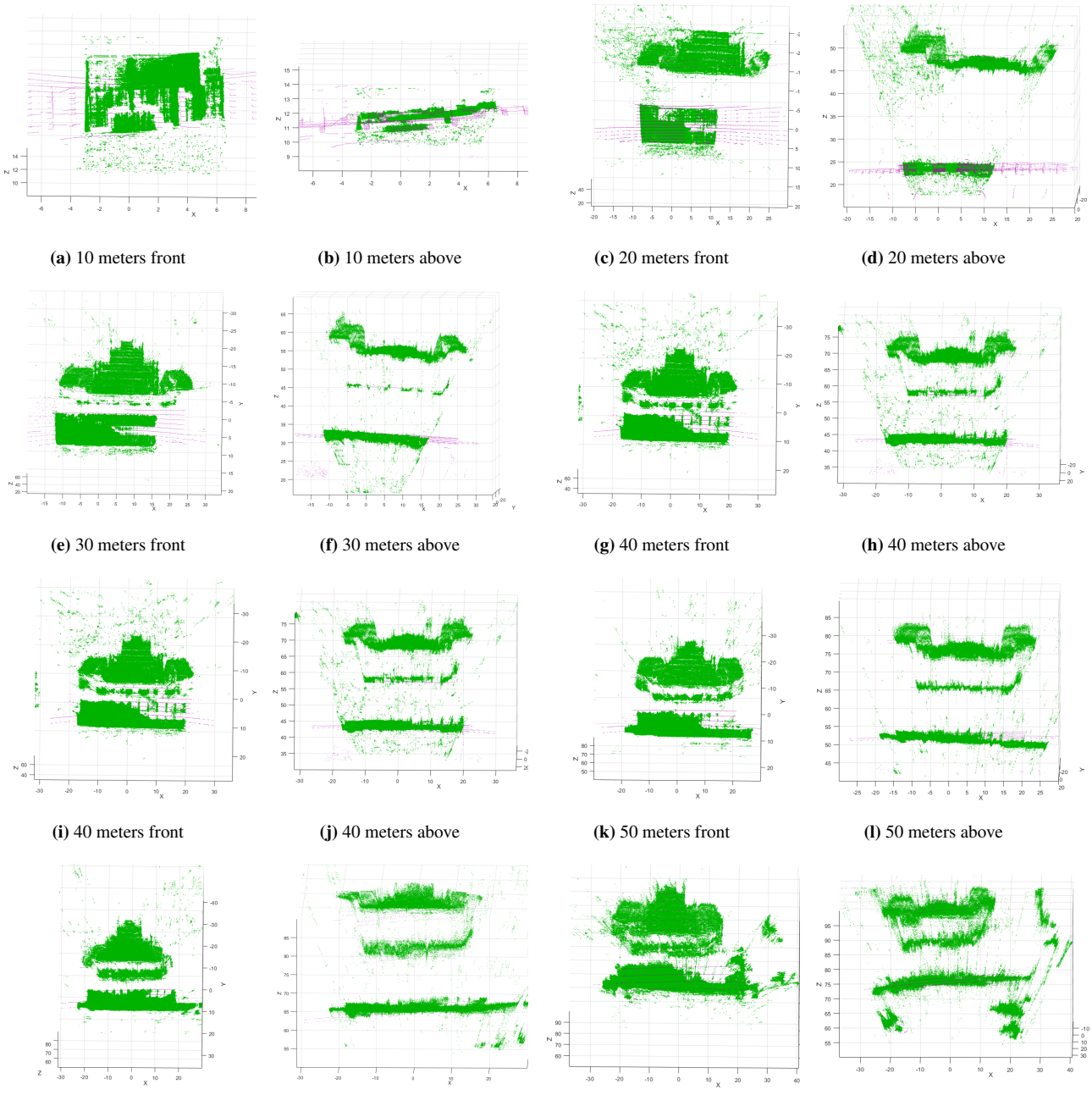
Zhang, Z., 2000. A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence 22, 1330 – 1334.

Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X., 2019. Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems 30, 3212–3232.

Appendix A

Reconstructed point clouds of calibration scenes

This appendix contains plots of the reconstructed point clouds of each calibration scene. The obtained stereo parameters are used in a semi-global matching algorithm to create a disparity map and to reconstruct a point cloud of the calibration scene. The point clouds are plotted together with the ground truth of the scene, the LiDAR point cloud. The stereo point cloud is given in green and the LiDAR in purple.



(a) 10 meters front **(b)** 10 meters above **(c)** 20 meters front **(d)** 20 meters above

(e) 30 meters front **(f)** 30 meters above **(g)** 40 meters front **(h)** 40 meters above

(i) 40 meters front **(j)** 40 meters above **(k)** 50 meters front **(l)** 50 meters above

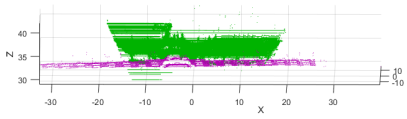
(m) 60 meters front **(n)** 60 meters above **(o)** 70 meters front **(p)** 70 meters above

Figure A.1: Stereo point cloud of calibration scenes plotted together with LiDAR point cloud

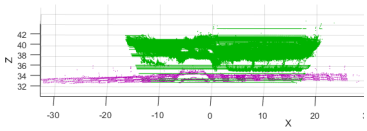
Appendix B

Reconstructed point clouds of test scenes

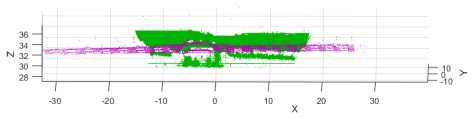
This appendix contains plots of the stereo point clouds of each test scene. The point clouds are plotted together with the ground truth of the scene, the LiDAR point cloud. The stereo point cloud is given in green and the LiDAR in purple.



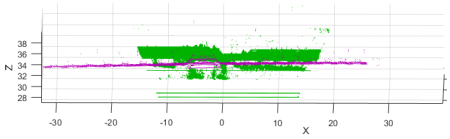
(a) 7.6 meter MATLAB calibration



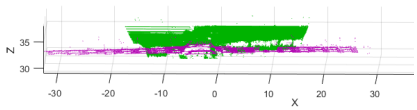
(b) 10 meter calibration



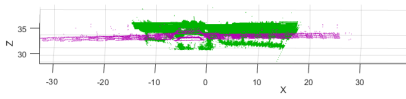
(c) 20 meter calibration



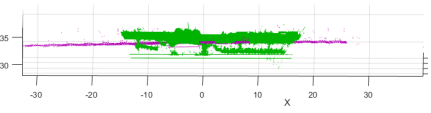
(d) 30 meter calibration



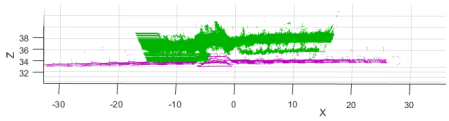
(e) 40 meter calibration



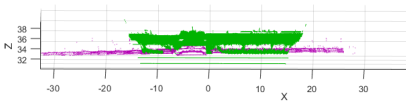
(f) 50 meter calibration



(g) 60 meter calibration

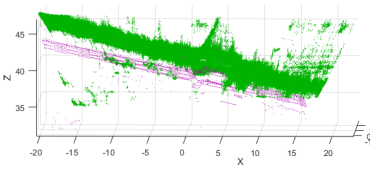


(h) 70 meter calibration

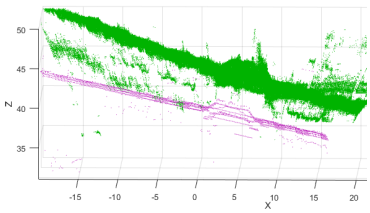


(i) Average calibration

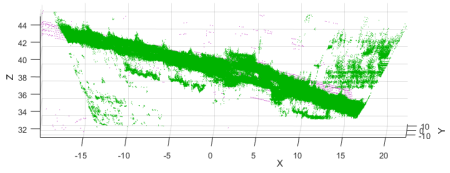
Figure B.1: Test scene at 30 meters



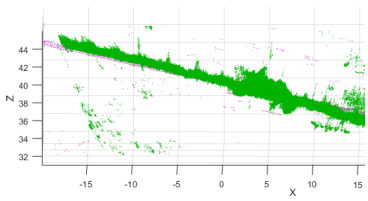
(a) 7.6 meter MATLAB calibration



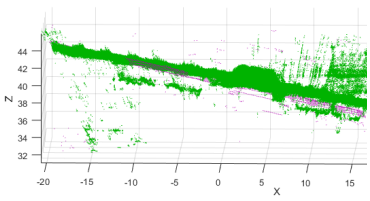
(b) 10 meter calibration



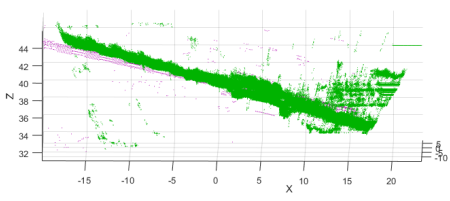
(c) 20 meter calibration



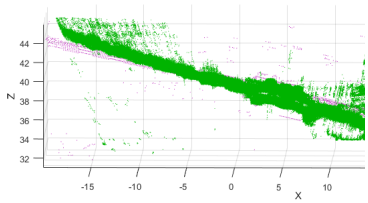
(d) 30 meter calibration



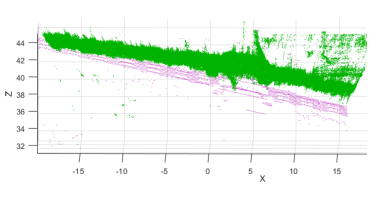
(e) 40 meter calibration



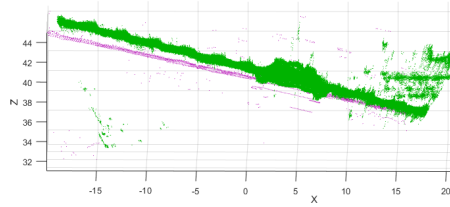
(f) 50 meter calibration



(g) 60 meter calibration

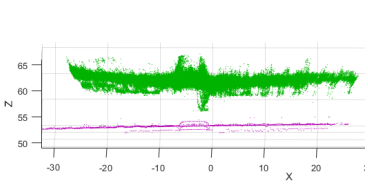


(h) 70 meter calibration

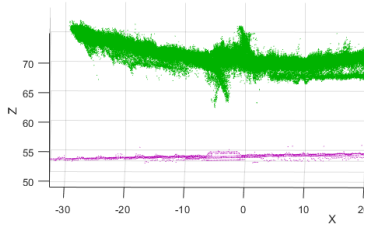


(i) Average calibration

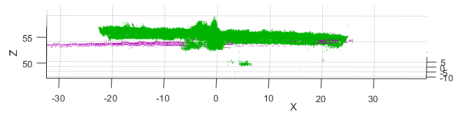
Figure B.2: Test scene at 40 meters



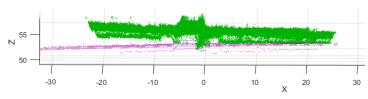
(a) 7.6 meter MATLAB calibration



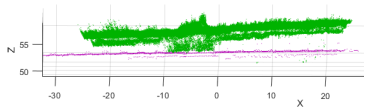
(b) 10 meter calibration



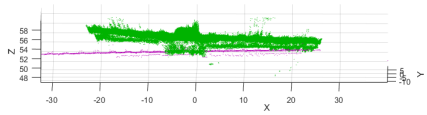
(c) 20 meter calibration



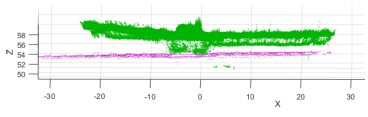
(d) 30 meter calibration



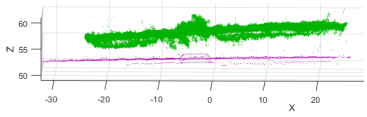
(e) 40 meter calibration



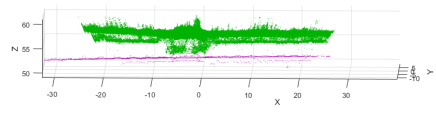
(f) 50 meter calibration



(g) 60 meter calibration

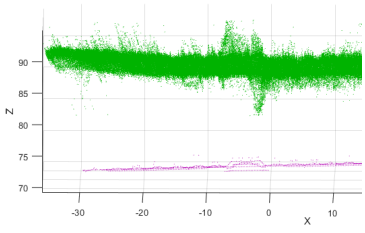


(h) 70 meter calibration

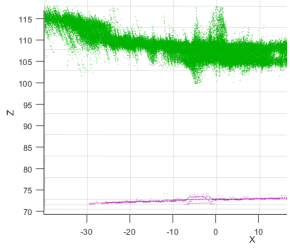


(i) Average calibration

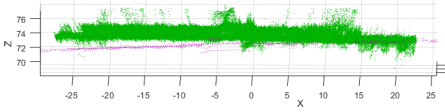
Figure B.3: Test scene at 50 meters



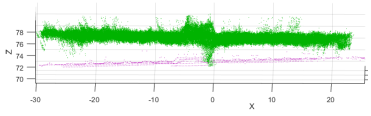
(a) 7.6 meter MATLAB calibration



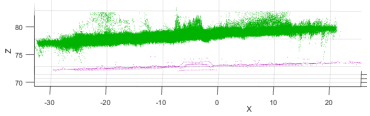
(b) 10 meter calibration



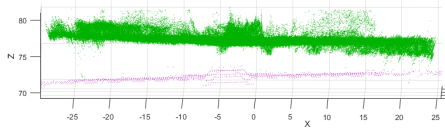
(c) 20 meter calibration



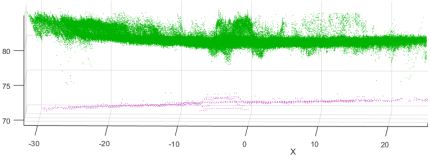
(d) 30 meter calibration



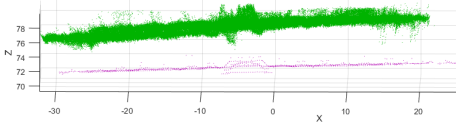
(e) 40 meter calibration



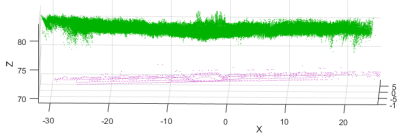
(f) 50 meter calibration



(g) 60 meter calibration

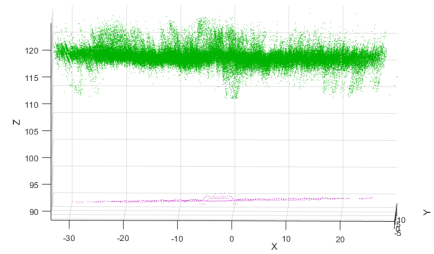


(h) 70 meter calibration

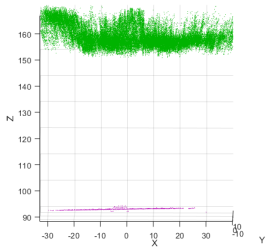


(i) Average calibration

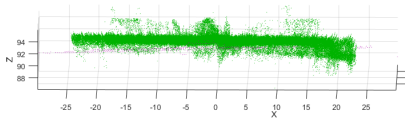
Figure B.4: Test scene at 70 meters



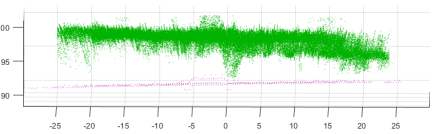
(a) 7.6 meter MATLAB calibration



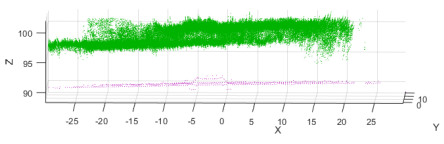
(b) 10 meter calibration



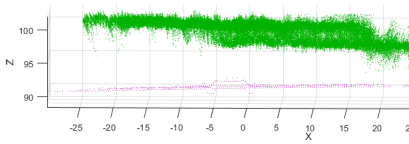
(c) 20 meter calibration



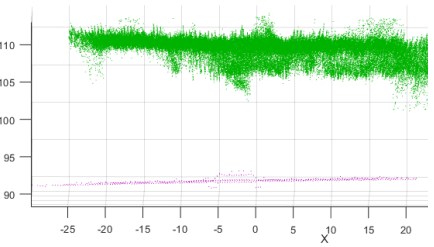
(d) 30 meter calibration



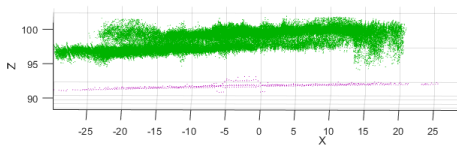
(e) 40 meter calibration



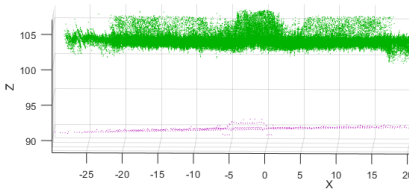
(f) 50 meter calibration



(g) 60 meter calibration



(h) 70 meter calibration



(i) Average calibration

Figure B.5: Test scene at 90 meters

Appendix C

Github repository, Readme

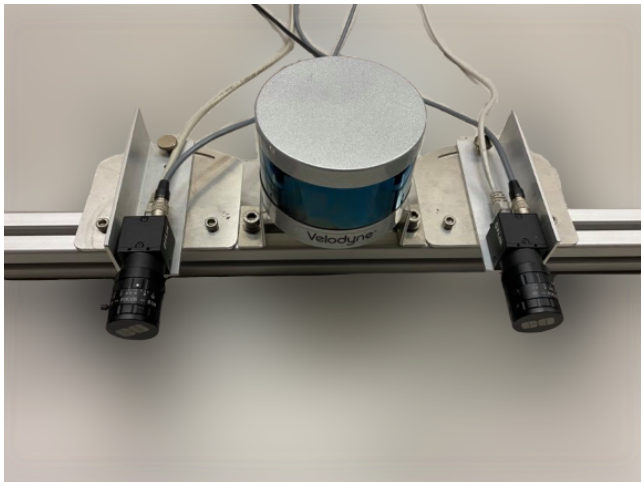
README.md

Stereo Vision for Autonomous ferry - code

The project is a part of TTK4900 - Engineering Cybernetics, Master's Thesis at Norwegian University of Science and Technology. "Stereo Vision for Autonomous ferry" goes into detail of the calibration, the chosen stereo setup, and the implementation for testing of dynamic scenes. The ownship has adequate data to sense, process, and understand its surroundings.

Contents

Getting Started



Example of the stereo system tested. The baseline in use measures 1.8meters, and the cameras are angled 1 degree inwards. The stereo system was tested using Blackfly S GigE with PoE, master software triggering, and GPIO pins for external triggering of the slave. The Velodyne LiDAR Puck-16 serves as ground truth for the system.

Prerequisites

- Set up a computer with GPU and Ubuntu 16.04. The repo is tested on a Dell Precision 7530 with NVIDIA Quadro P3200.
- Set up a stereo system, and perform a stereo calibration of the cameras.
- Install Spinnaker and verify that you can run your cameras with SpinViw (Windows operating system is preferred). Set up a stereo system with either hardware or software triggering. Modify the yaml files in spinnaker-sdk-driver by replacing the cam-ids and master cam serial number to match your camera's serial number. In the same yaml file, include the calibration parameters. (If using Matlab for calibration, make sure to convert the numbers to OpenCV calibration definitions)
- Optionally: Set up the LiDAR driver. An extrinsic calibration of the LiDAR and the stereo camera must be performed for the LiDAR to serve as a basis for comparison.

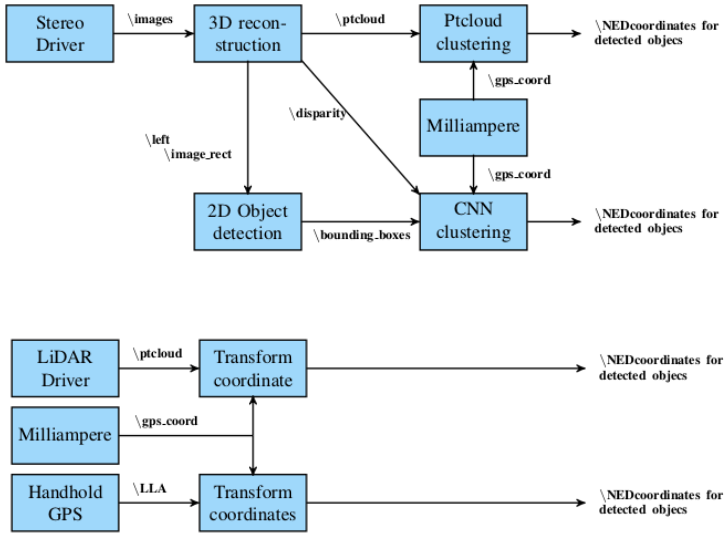
Installing

- ROS Kinetic
- CUDA 10.2 for runnning darknet ros (YOLOv3) on GPU.
- Python. The scripts is tested with Python 2.7.12
    i. Matplotlib and OpenCV

```
sudo apt install python-pip #Installs Python pip
python -mpip install -U pip
python -mpip install -U matplotlib #Plot the results by installing Matplotlib
python -mpip install -U opencv-python #Show animation by opencv
```

- For the stereoTuner the gtk3 development files is needed for cmake: `sudo apt-get install build-essential libgtk-3-dev`

System overview



This shows an illustration of the overview of the stereo system and the ground truth. For a full overview run:

```
rosrun rqt_graph rqt_graph
```

Launch

```
cd Master
catkin build
source devel/setup.bash
roslaunch launch clustering_cnn file:= "your bag file"  #launch bagfile, stereo_image_proc, yolo and clustering_cnn
roslaunch launch clustering_ptcloud file:= "your bag file"  #launch bagfile, stereo_image_proc, clustering_ptcloud
```

Run individual packages

Camera driver

- `roslaunch spinnaker_sdk_camera_driver acquisition.launch`
- Or Rosbag: `rosbag play "filename" --clock`

Stereo image proc

- `ROS_NAMESPACE=camera_array rosrun stereo_image_proc stereo_image_proc`

Display images

- Rectified images and disparity map`
    i. `rosrun image_view stereo_view stereo:=/camera_array image:=image_rect_color`
- Raw stereo images 2. `rosrun image_view stereo_view stereo:=/camera_array image:=image_raw`
- Left rectified image 3. `rosrun image_view image_view image:=/camera_arr/left/image_rect_color`

Clustering Point Cloud

- `roslaunch clustering pcl_obstacle_detector.launch #Need the bagfile to be run with the "--clock"`
- Tuning the parameters: `rosrun rqt_reconfigure rqt_reconfigure`
- Visualize in rviz: `rosrun rviz rviz -f velodyne` The package filters and clusters the point cloud, publishing the point clouds, and transforms the clusters to NED coordinates. The class makes use of PassThrough filter, Statistical Outlier Removal, Voxel Grid filter, accumulates subsequent point clouds, and Euclidean clustering.

Darknet_ros (YOLOv3)

- `roslaunch darknet_ros yolo_v3.launch #subscribes on camera_array/left/image_rect_color`

Clustering Convolutional Neural Network

- `rosrun clustering_cnn clustering_cnn`

Navigation data

- `rosrun navigation_data *filename*`

Connect to Network on MilliAmpere

If the code is to be run with the master core on the ferry Milliampere a local network needs to be setup. The best solution is to set up a separate static network on the computer through usb3.

```
ifconfig #find your network
sudo nano /etc/network/interfaces #add/make the network static
nano /etc/hosts #add milliAmpere as host on machine, further do the same at milliAmpere
sudo ifdown "your network" && sudo ifup "your network" #restarting interface
ping milliAmpere #check if connection established
```

Now that you have created a static network, the roscore has to be exported. This tells your machine that the roscore is running on another computer. Be sure to do this in every terminal window (can be an advantage to add in nano .bashrc)

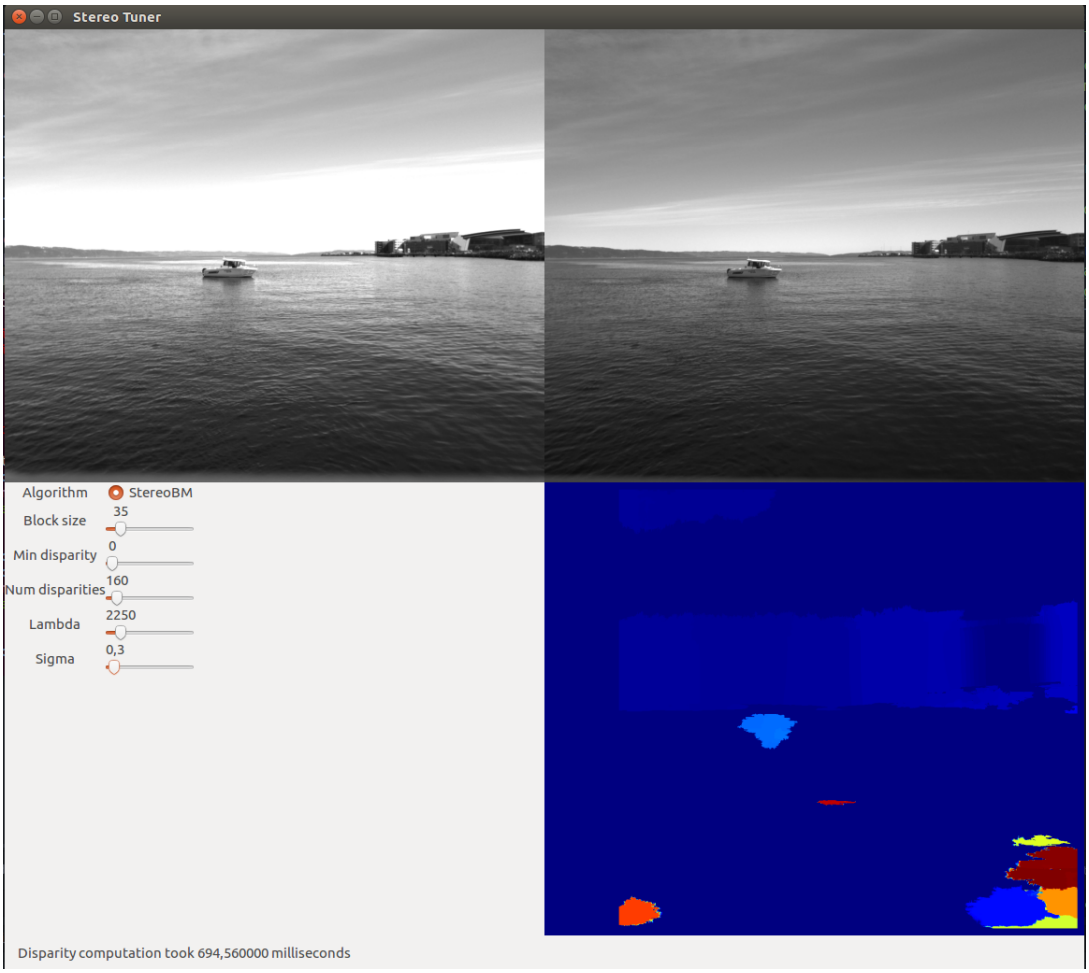
```
export ROS_MASTER_URI=http://milliAmpere:11311
echo $ROS_MASTER_URI
rostopic echo /topic #test that connection is established
```

Application and Scripts

stereoTuner

Application for tuning the disparity map, implemented using the stereoBM object and WLS filter from openCV. The GUI is a modified version of the repository stereo-tuner. Remember to rectify the images beforehand; you're welcome. It is a simple little GTK application that can be used to tune parameters for the OpenCV Stereo Vision algorithms.

```
mkdir build
cd build
cmake ..
make
./main
```



Label YOLO images

GUI for marking bounded boxes of objects in images for training Yolo v3.

- Put your `.jpg` -images in the directory `x64/Release/data/img`.
- Change the numer of classes (objects for detection) in file `x64/Release/data/obj.data
- Put names of the objects, one for each line in the file `x64/Release/data/obj.names If the images are used for training a custom dataset visit AlexBA for a more detailed explanation.

```
cmake .
make
./linux_mark.sh
```

Precision-recall curve for YOLOv3

Plots and calculates the precision-recall curve from ground truth images. It iterates through two for-loops, one with IoU-threshold and the second with the YOLO-threshold. Make sure to input detection images from the network with threshold less than the ones in the for-loop in main.py. The mAP script is a modified version of the code in the github repository mAp. It outputs a precision-recall curve for each threshold and IoU-threshold in main.py

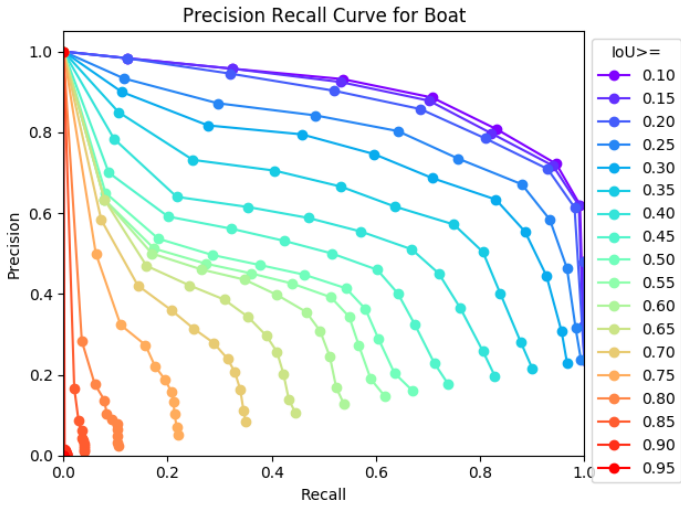
- Create the ground truth files using Label YOLO images
    i. Insert images into the folder input/ground-truth/images

   ii. Insert ground-truth files into ground-truth/
   iii. Add class list to the file scripts/extra/class_list.txt
   iv. Run the python script: python convert_gt_yolo.py to convert the txt-files to the correct format
- Create the detection-results files by running darknet_ros (It is created code (commented out) for saving box-files and rectified images in /src/darknet_ros/darknet_ros/src/YoloObjectDetector.cpp)
   i. Copy the detection-results files into the folder input/detection-results/
- Run the script: `python main.py`
   i. Note: To run the script once, for a given IoU-threshold and YOLO-threshold,run the following command line. This wil process mAP, detection result information etc in the folder /output. `python precision-recall.py "IoU-threshold" "YOLO-threshold"`

Note: be consistent with using rectified/not-rectified images.

```
python main.py
```

Feel free to edit the main file with your preferred values in the for-loops.



Plot and calculate ground truth depth and stereo depth

Matches handhold-GPS csv file with the GPS from MA by satellite time. Match the stereo ros-time and plots estimated depth and NED coordinates in a beautiful graph.

Authors and License

Good-luck from two soon-to-be well educated grown-ups.

- **Trine Ødegård Olsen** - trineoo
- **Lina Theimann** - linact

This project is licensed under the MIT License - see the LICENSE.md file for details

Appendix D

Labeling ground truth dataset for YOLOv3 evaluation

To evaluate the accuracy of the weights in the network, a ground truth dataset was labeled. The dataset is captured outside the port Brattøra and consists of 483 images. The images are rectified with the given calibration parameters, and manually labeled with the graphical user interface developed by AlexeyAB[1]. The GUI created for marking bounded boxes of objects in images for neural network is shown in Figure D.2. The GUI output a textfile for each image, containing the class of labeled objects with the corresponding bounding box. The dataset contains images of one boat at different heading from the camera's point of view. All the images are taken 40- to 150-meters from the boat. An example of the pictures in the dataset is displayed in Figure D.1. The pictures are easy to label and are considered a relatively fair scene considering the training set. The dataset is expected to yield satisfactory results, given the operating environment.

[1] https://github.com/AlexeyAB/Yolo_mark

Figure D.1: Images of the dataset

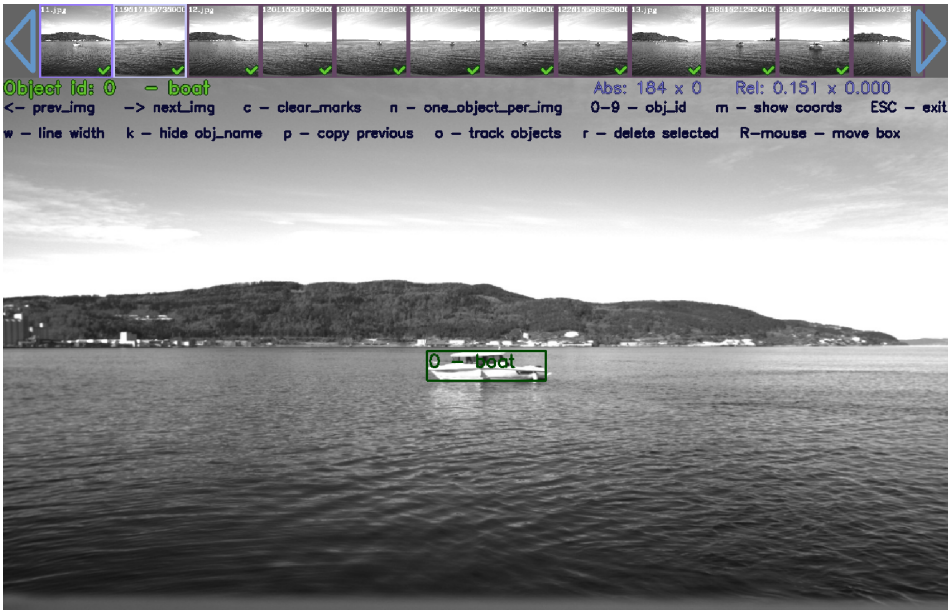


Figure D.2: GUI for labeling images

Appendix E

Result of different YOLO-thresholds

With IoU-threshold of 0.2, different YOLO-threshold yields the following result. Used to find the systems optimal threshold.

YOLO-threshold	$IoU >= 0.20$		
	Area	**Precision**	**Recall**
0.03	0.2379	0.2379	1.0000
0.04	0.2379	0.2379	1.0000
0.05	0.2379	0.2379	1.0000
0.06	0.2564	0.2564	1.0000
0.07	0.2712	0.2712	1.0000
0.08	0.2880	0.2880	1.0000
0.09	0.3028	0.3028	1.0000
0.10	0.3205	0.3205	1.0000
0.11	0.3371	0.3371	1.0000
0.12	0.3526	0.3526	1.0000
0.13	0.3698	0.3698	1.0000
0.14	0.3861	0.3861	1.0000
0.15	0.4022	0.4022	1.0000
0.16	0.4178	0.4178	1.0000
0.17	0.4347	0.4347	1.0000
0.18	0.4476	0.4476	1.0000
0.19	0.4649	0.4649	1.0000
0.20	0.4792	0.4792	1.0000
0.21	0.4898	0.4908	0.9979
⋮	⋮	⋮	⋮

YOLO-threshold	$IoU >= 0.20$		
	Area	**Precision**	**Recall**
0.22	0.5037	0.5047	0.9979
0.23	0.5172	0.5183	0.9979
0.24	0.5368	0.5379	0.9979
0.25	0.5503	0.5515	0.9979
0.26	0.5659	0.5671	0.9979
0.27	0.5747	0.5783	0.9938
0.28	0.5847	0.5909	0.9896
0.29	0.5940	0.6015	0.9876
0.30	0.6025	0.6140	0.9814
0.31	0.6087	0.6216	0.9793
0.32	0.6232	0.6391	0.9752
0.33	0.6318	0.6479	0.9752
0.34	0.6378	0.6582	0.9689
0.35	0.6450	0.6686	0.9648
0.36	0.6508	0.6789	0.9586
0.37	0.6519	0.6830	0.9545
0.38	0.6579	0.6923	0.9503
0.39	0.6603	0.6994	0.9441
0.40	0.6594	0.7093	0.9296
0.41	0.6664	0.7217	0.9234
0.42	0.6633	0.7265	0.9130
0.43	0.6610	0.7356	0.8986
0.44	0.6665	0.7470	0.8923
0.45	0.6631	0.7590	0.8737
0.46	0.6540	0.7648	0.8551
0.47	0.6478	0.7669	0.8447
0.48	0.6390	0.7697	0.8302
0.49	0.6379	0.7780	0.8199
0.50	0.6356	0.7851	0.8095
0.51	0.6394	0.7939	0.8054
0.52	0.6434	0.8071	0.7971
0.53	0.6402	0.8137	0.7867
0.54	0.6277	0.8172	0.7681
0.55	0.6266	0.8247	0.7598
0.56	0.6171	0.8326	0.7412
0.57	0.6097	0.8413	0.7246
0.58	0.6079	0.8536	0.7122
0.59	0.6064	0.8589	0.7060
0.60	0.5877	0.8575	0.6853
0.61	0.5852	0.8670	0.6749
0.62	0.5639	0.8674	0.6501
⋮	⋮	⋮	⋮

YOLO-threshold	$IoU >= 0.20$		
	Area	**Precision**	**Recall**
0.63	0.5657	0.8785	0.6439
0.64	0.5489	0.8779	0.6253
0.65	0.5419	0.8813	0.6149
0.66	0.5223	0.8851	0.5901
0.67	0.5182	0.8971	0.5776
0.68	0.4973	0.8963	0.5549
0.69	0.4843	0.8997	0.5383
0.70	0.4692	0.9029	0.5197
0.71	0.4504	0.9026	0.4990
0.72	0.4271	0.9048	0.4720
0.73	0.4165	0.9061	0.4596
0.74	0.4001	0.9030	0.4431
0.75	0.3937	0.9276	0.4244
0.76	0.3772	6 0.9249	0.4079
0.77	0.3564	0.9254	0.3851
0.78	0.3373	0.9309	0.3623
0.79	0.3159	0.9419	0.3354
0.80	0.3033	0.9451	0.3209
0.81	0.2804	0.9470	0.2961
0.82	0.2533	0.9485	0.2671
0.83	0.2304	0.9512	0.2422
0.84	0.2137	0.9558	0.2236
0.85	0.1951	0.9519	0.2050
0.86	0.1825	0.9583	0.1905
0.87	0.1639	0.9540	0.1718
0.88	0.1514	0.9620	0.1573
0.89	0.1307	0.9565	0.1366
0.90	0.1201	0.9833	0.1222
0.91	0.1056	1.0000	0.1056
0.92	0.0828	1.0000	0.0828
0.93	0.0745	1.0000	0.0745
0.94	0.0621	1.0000	0.0621
0.95	0.0518	1.0000	0.0518
0.96	0.0455	1.0000	0.0455
0.97	0.0414	1.0000	0.0414
0.98	0.0373	1.0000	0.0373
0.99	0.0269	1.0000	0.0269

Appendix F

The milliAmpere system

The system running on Milliampere can be comprised in the following graph.

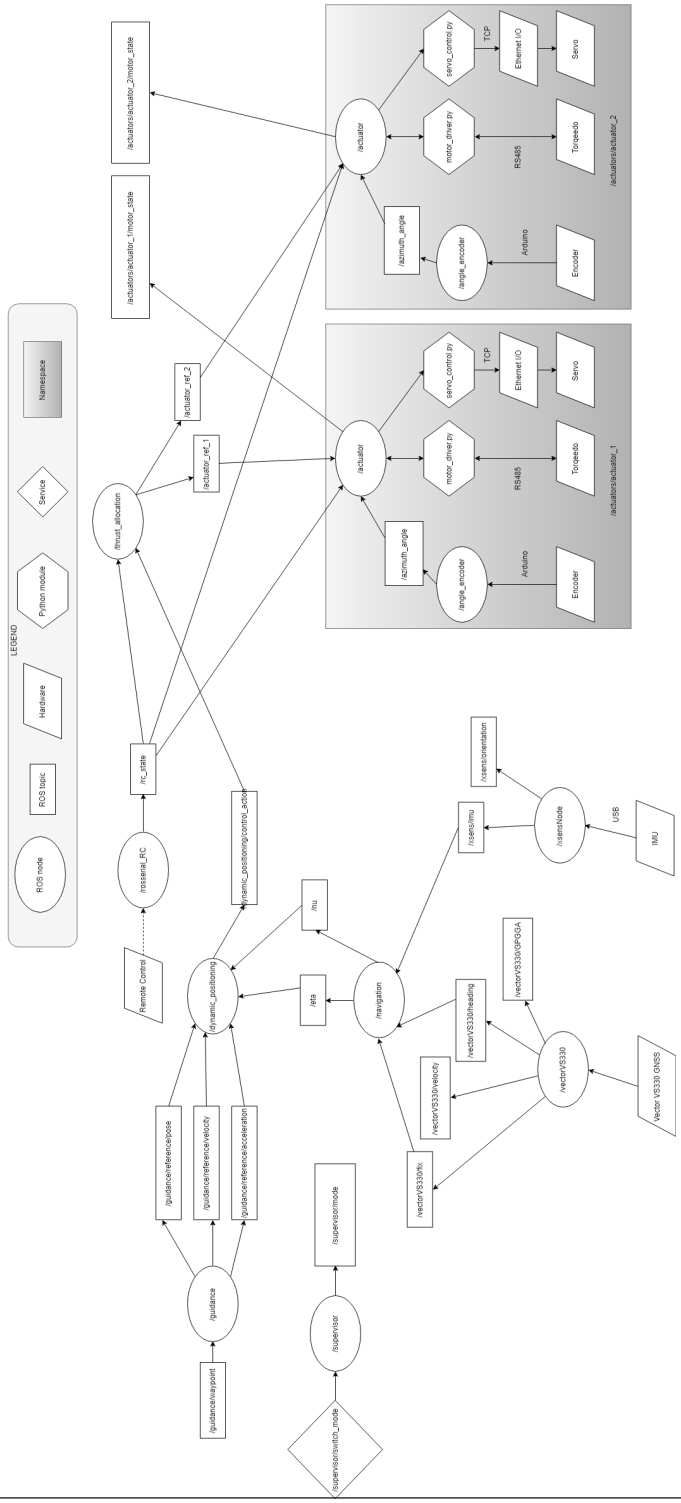
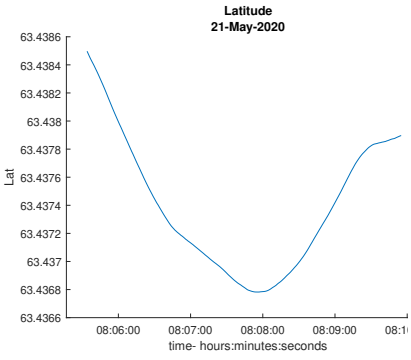


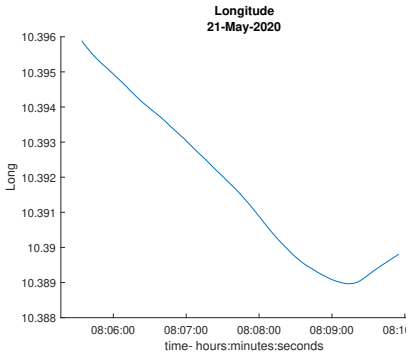
Figure F.1: Graph of the ROS system on the ferry Milliampere

Appendix G

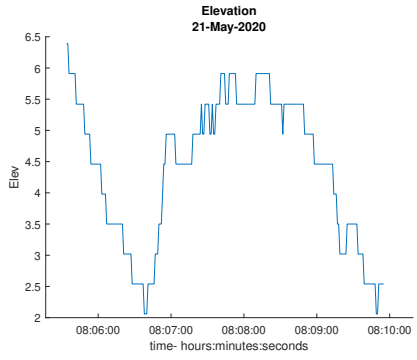
Ground truth accuracy plots



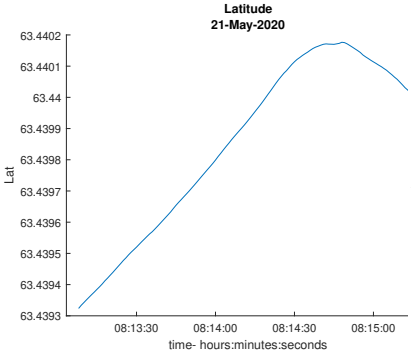
(a) Latitude scenario 1



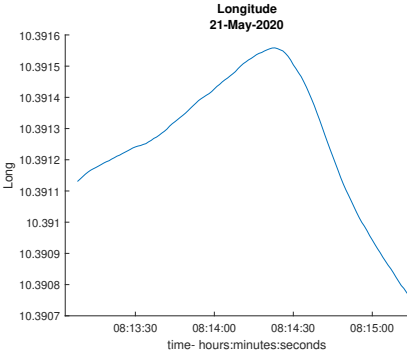
(b) Longitude scenario 1



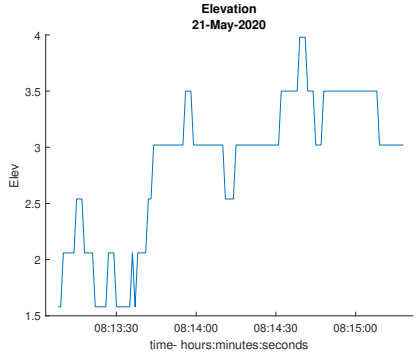
(c) Elevation scenario 1



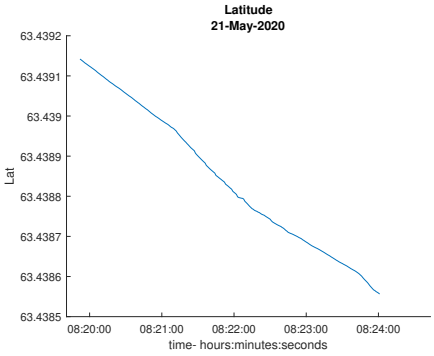
(d) Latitude scenario 2



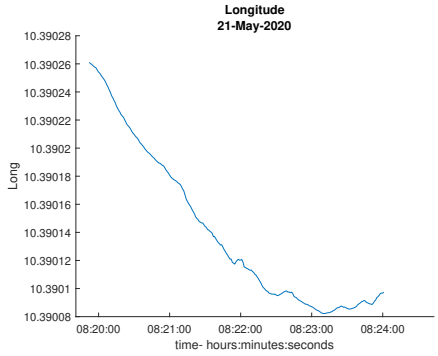
(e) Longitude scenario 2



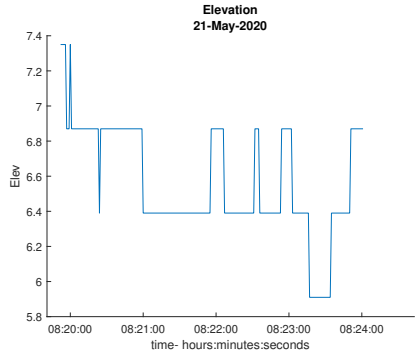
(f) Elevation scenario 2



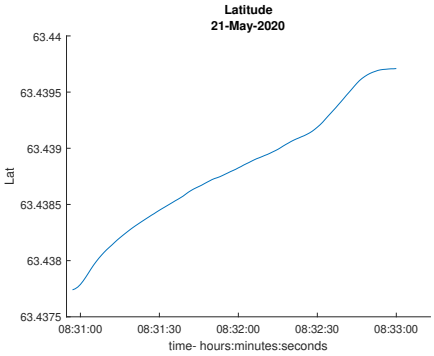
(a) Latitude scenario 3



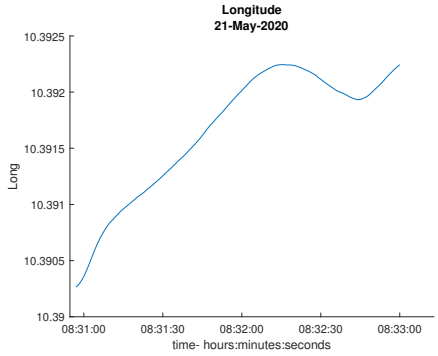
(b) Longitude scenario 3



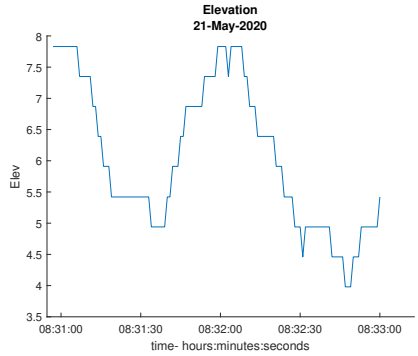
(c) Elevation scenario 3



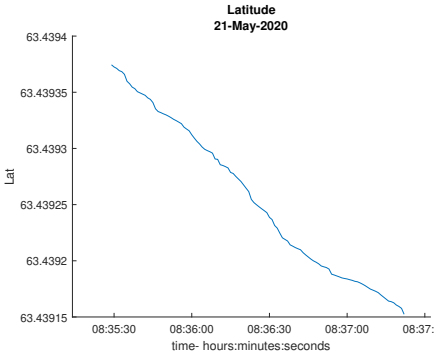
(d) Latitude scenario 4



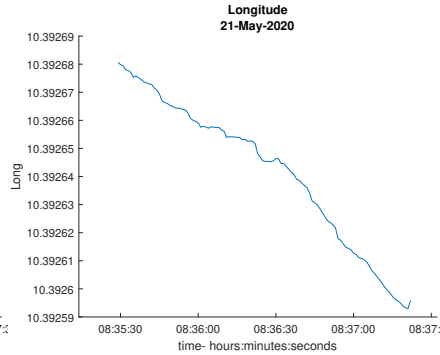
(e) Longitude scenario 4



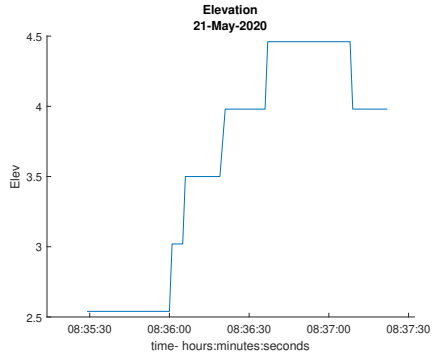
(f) Elevation scenario 4



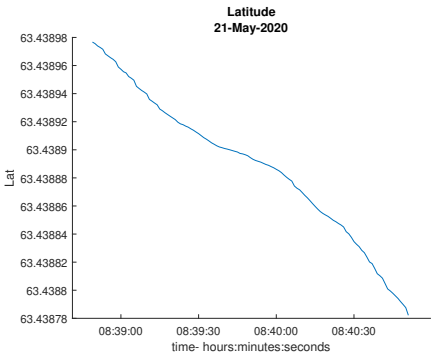
(g) Latitude scenario 5



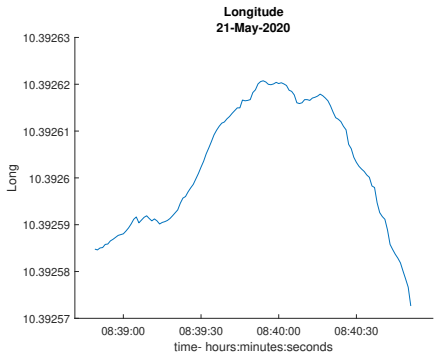
(h) Longitude scenario 5



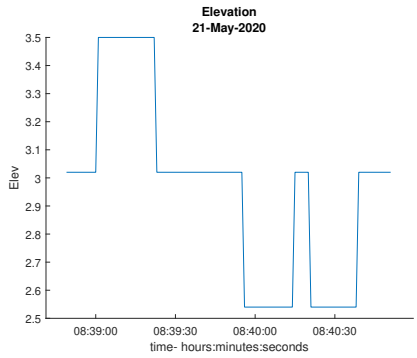
(i) Elevation scenario 5



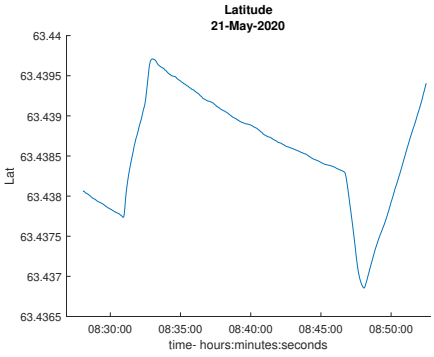
(j) Latitude scenario 6



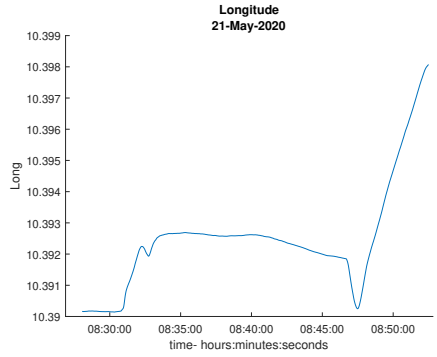
(k) Longitude scenario 6



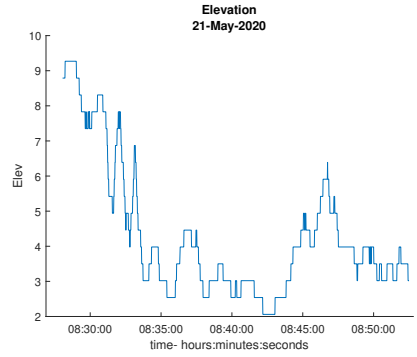
(l) Elevation scenario 6



(m) Longitude scenario 7

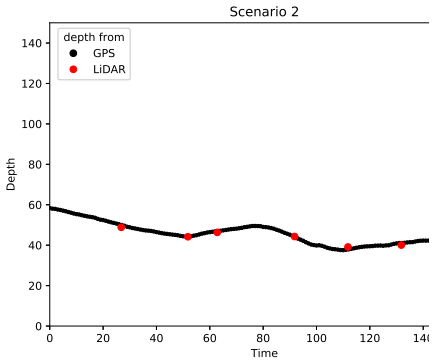


(n) Latitude scenario 7

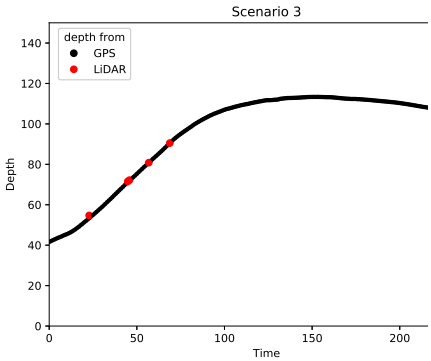


(o) Elevation scenario 7

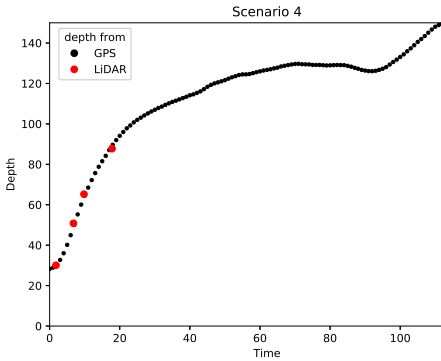
Figure G.2: GPS longitude, latitude and elevation plots



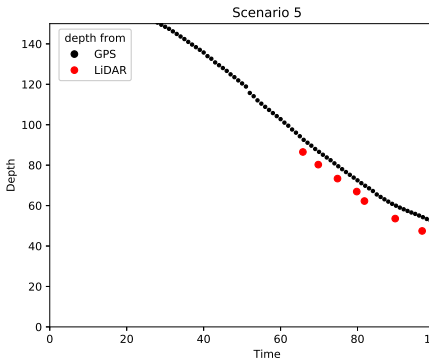
(a) Scene 2



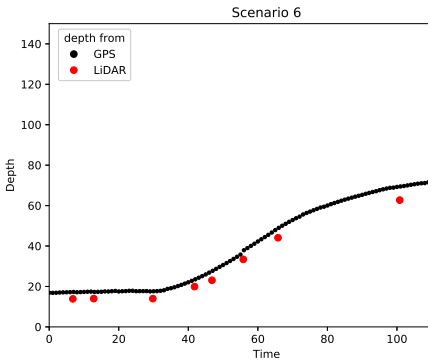
(b) Scene 3



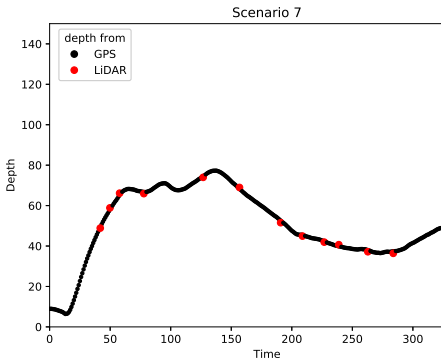
(c) Scene 4



(d) Scene 5



(e) Scene 6



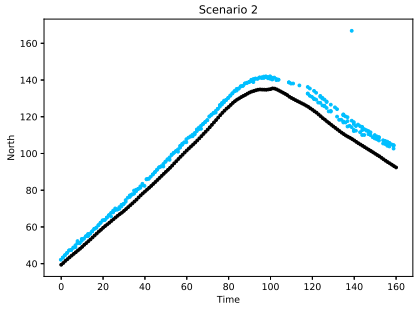
(f) Scene 7

Figure G.3: Depth from LiDAR and GPS

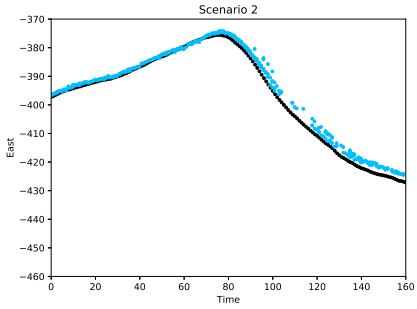
Appendix H

Result Plots

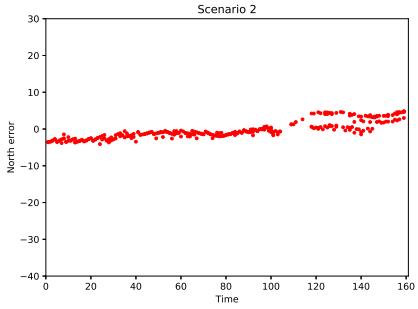
H.1 CNN-clustering



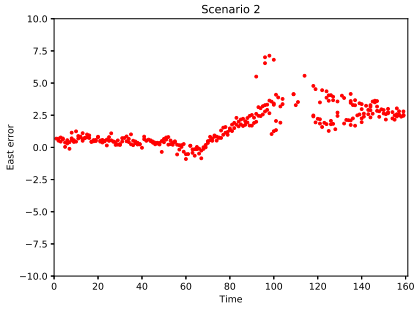
(a) North-time



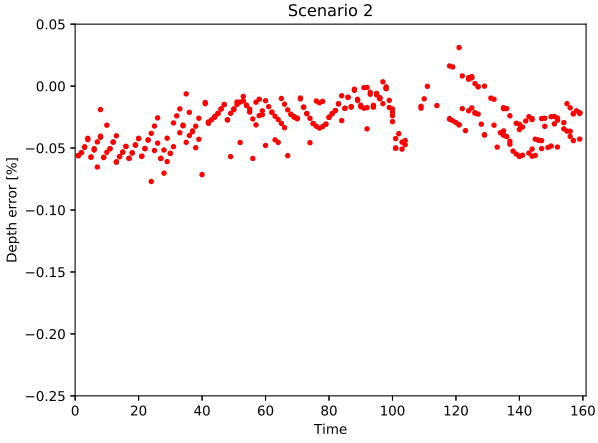
(b) East-time



(c) North-time error

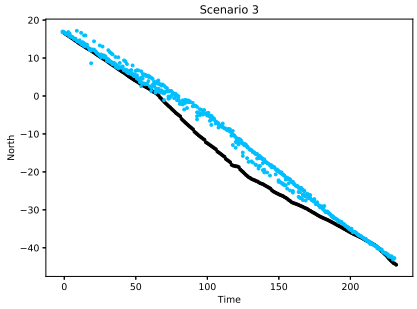


(d) East-time error

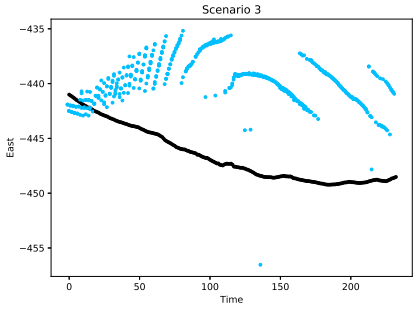


(e) Depth-time error percentage

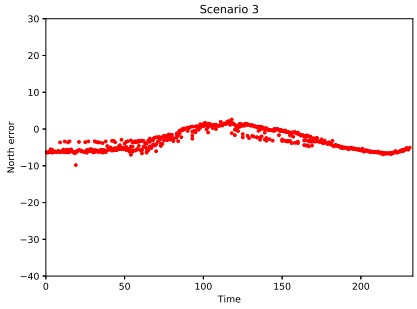
Figure H.1: Scenario 2



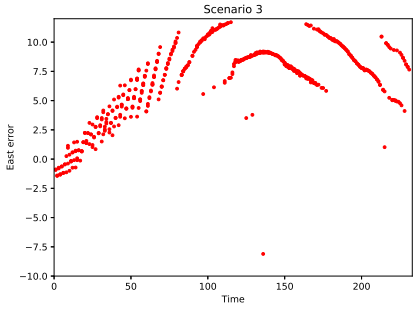
(a) North-time



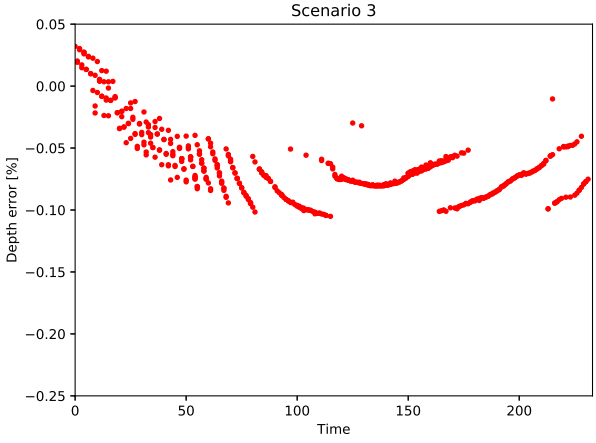
(b) East-time



(c) North-time error

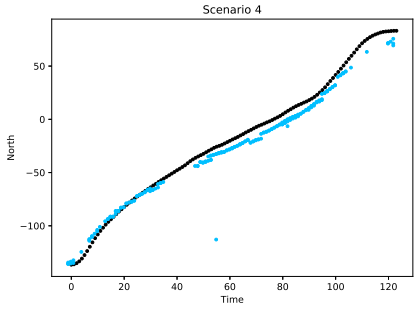


(d) East-time error

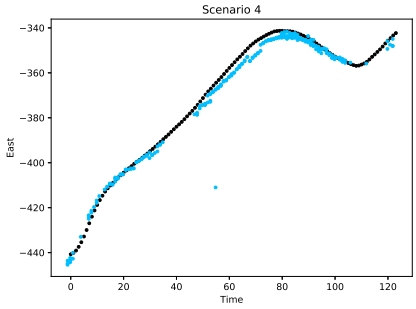


(e) Depth-time error percentage

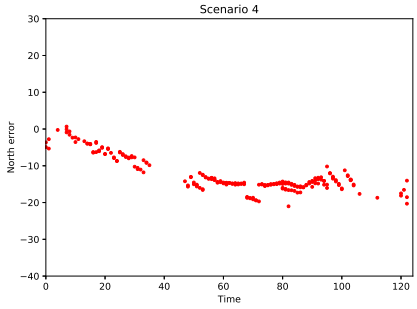
Figure H.2: Scenario 3



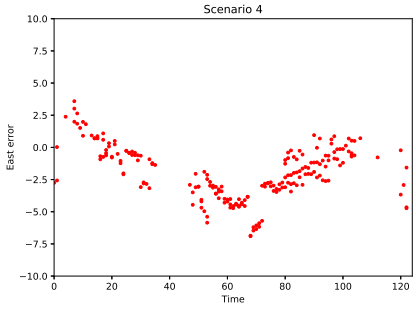
(a) North-time



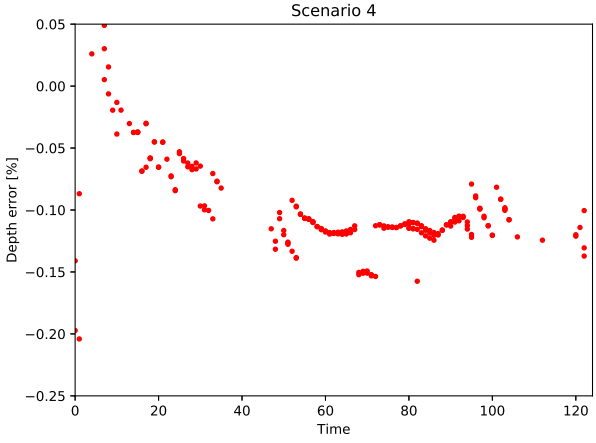
(b) East-time



(c) North-time error

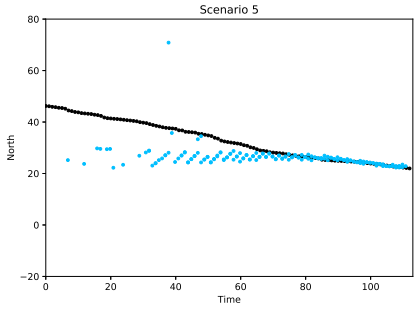


(d) East-time error

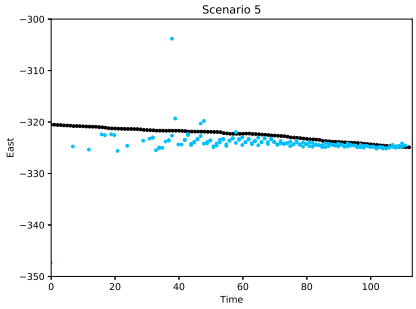


(e) Depth-time error percentage

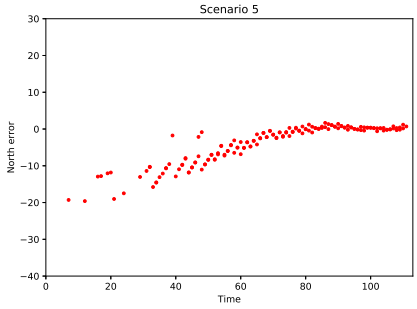
Figure H.3: Scenario 4



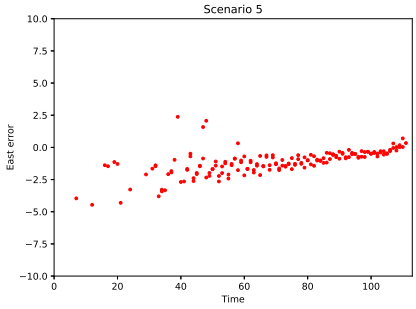
(a) North-time



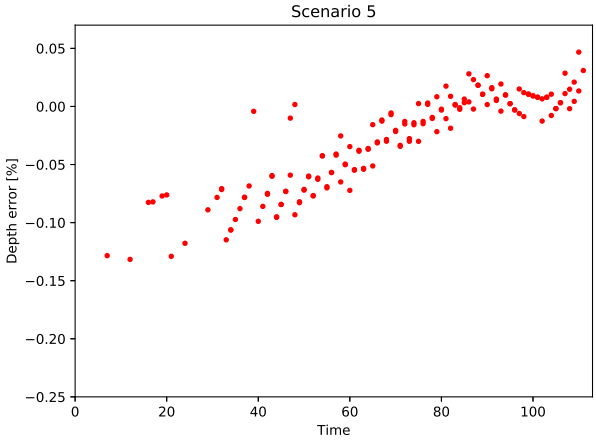
(b) East-time



(c) North-time error

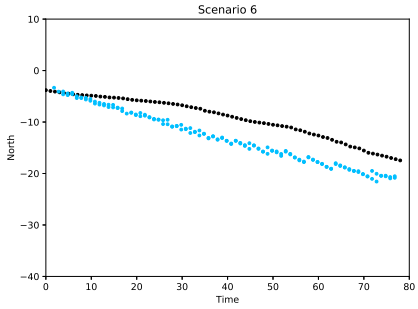


(d) East-time error

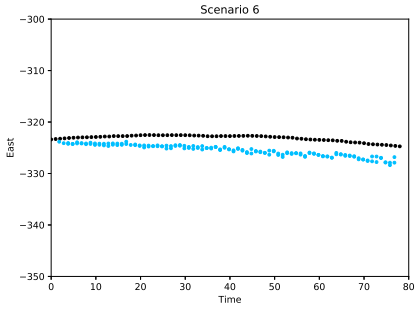


(e) Depth-time error percentage

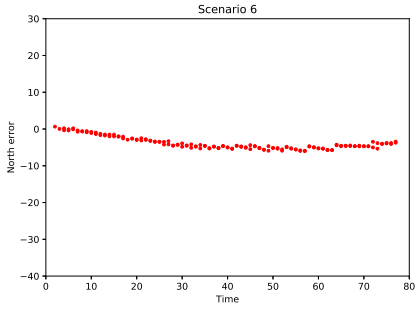
Figure H.4: Scenario 5



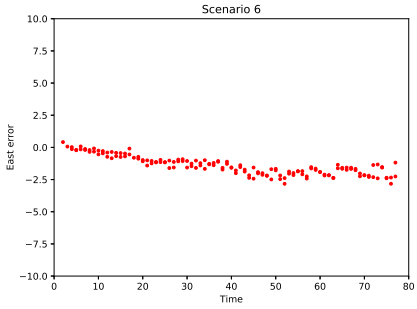
(a) North-time



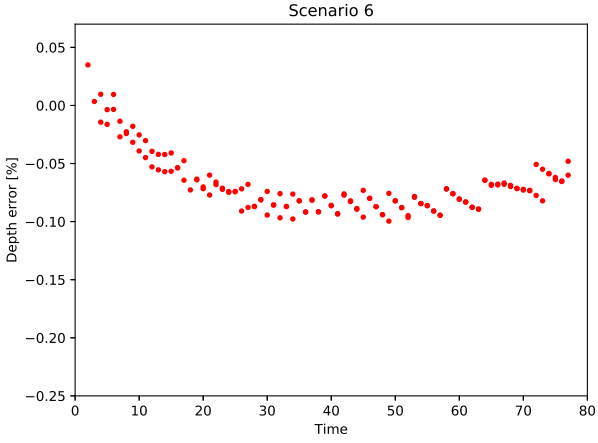
(b) East-time



(c) North-time error



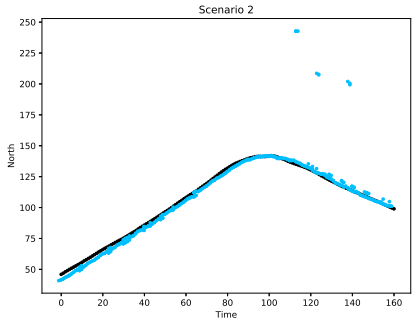
(d) East-time error



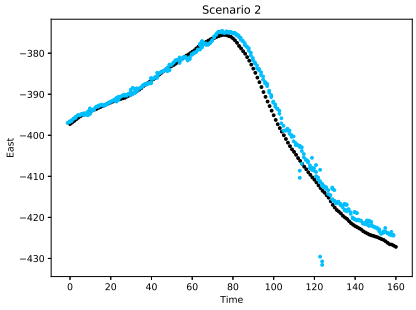
(e) Depth-time error percentage

Figure H.5: Scenario 6

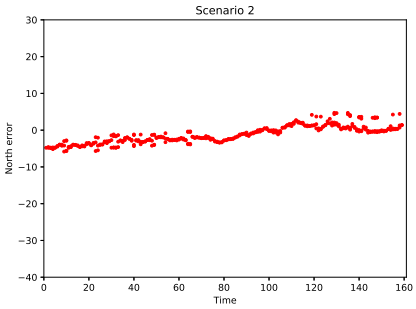
H.2 Ptcloud-clustering



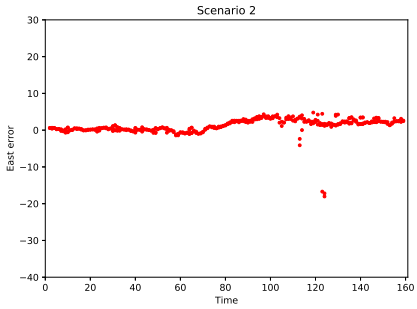
(a) North-time



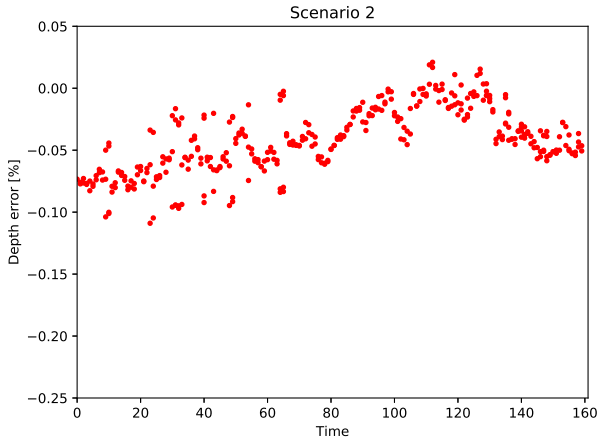
(b) East-time



(c) North-time error

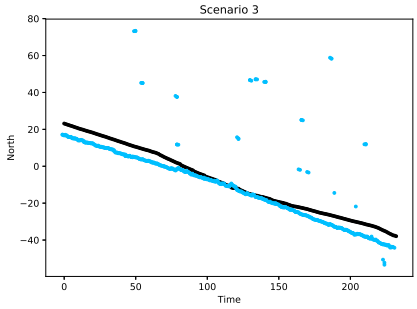


(d) East-time error

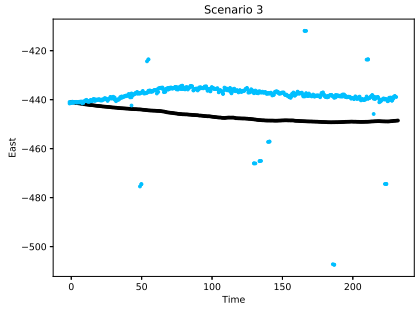


(e) Depth-time error percentage

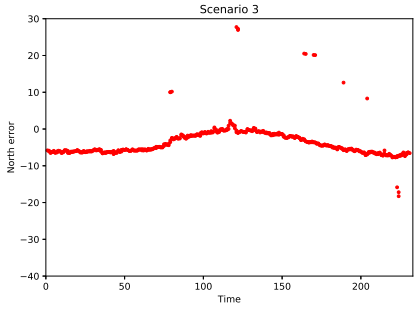
Figure H.6: Scenario 2



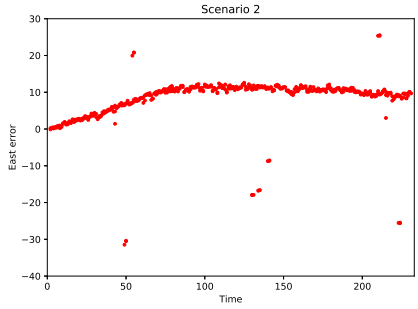
(a) North-time



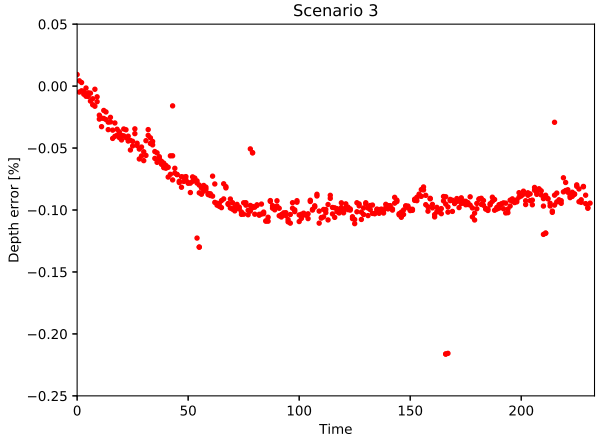
(b) East-time



(c) North-time error

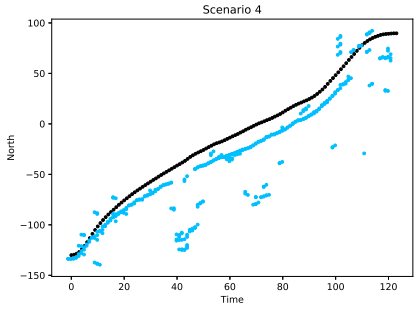


(d) East-time error

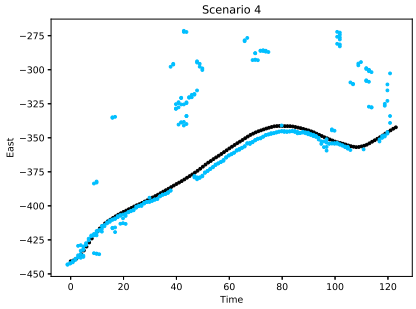


(e) Depth-time error percentage

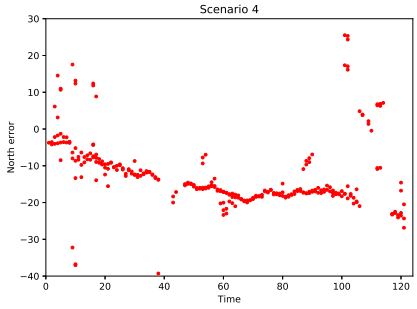
Figure H.7: Scenario 3



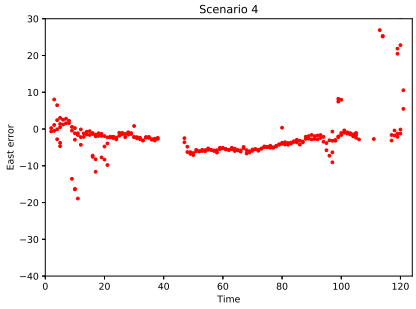
(a) North-time



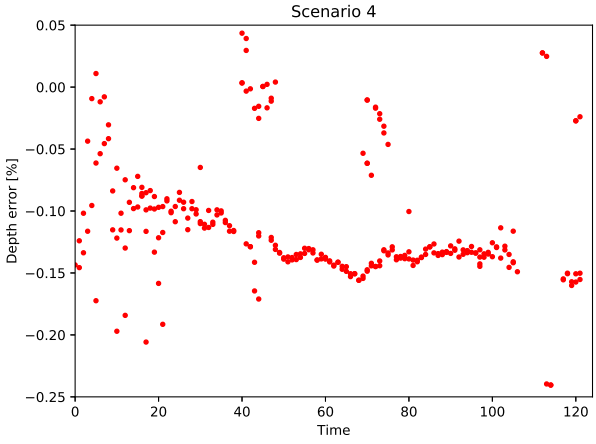
(b) East-time



(c) North-time error

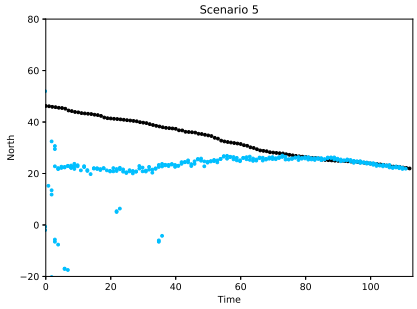


(d) East-time error

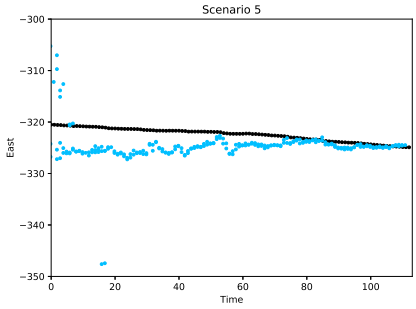


(e) Depth-time error percentage

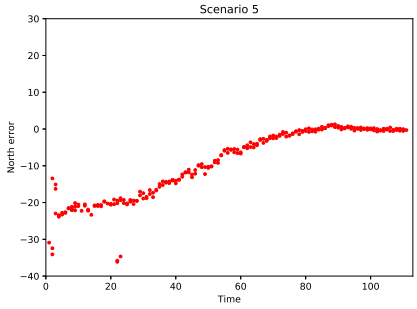
Figure H.8: Scenario 4



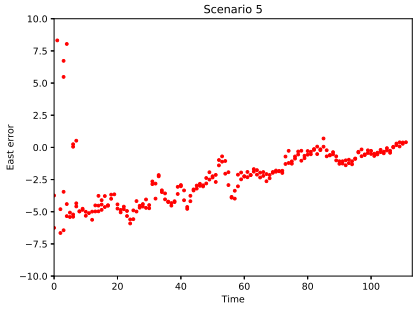
(a) North-time



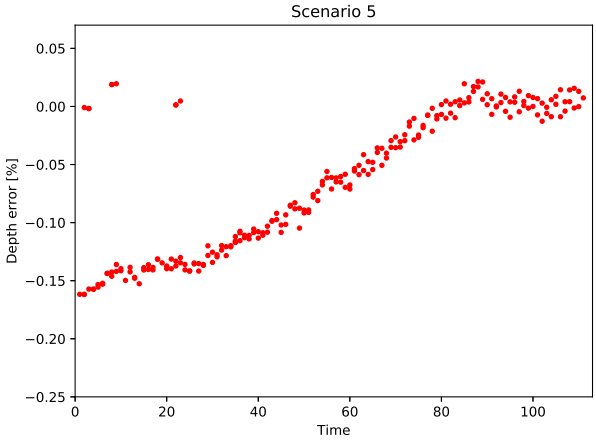
(b) East-time



(c) North-time error

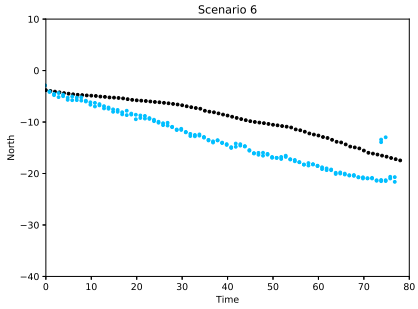


(d) East-time error

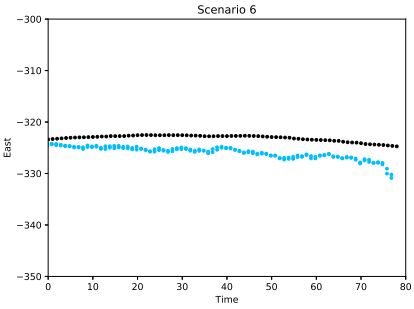


(e) Depth-time error percentage

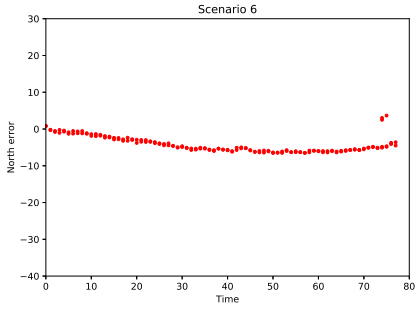
Figure H.9: Scenario 5



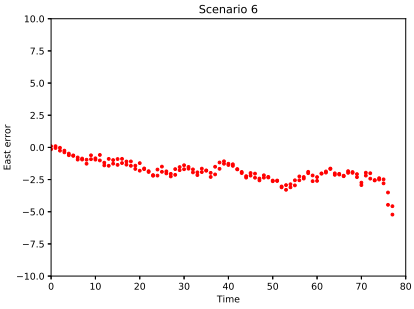
(a) North-time



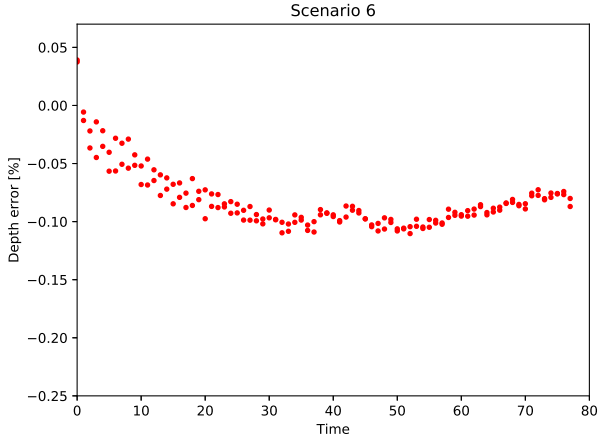
(b) East-time



(c) North-time error



(d) East-time error



(e) Depth-time error percentage

Figure H.10: Scenario 6

Lina C. K. Theimann, Trine Ø. Olsen

NTNU
Norwegian University of
Science and Technology