

# Camera-Lidar sensor fusion in real time for autonomous surface vehicles

**Håkon Gjertsen Norbye**

haakongjertsen@gmail.com

January 2019

Specialization project submitted to

*Norwegian University of Science and Technology*

in partial fulfillment for the award of the degree of

**MASTER OF SCIENCE**

in Cybernetics and Robotics



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

---

---

---

## Problem description

The Revolt project by DNV-GL in collaboration with NTNU aims to build knowledge around autonomous surface vehicles, their application and challenges. This includes several areas of interest such as collision avoidance, multi-target tracking and sensor fusion on embedded systems.

The purpose of this specialization project is to prepare a real-time sensor fusion pipeline on DNV GL's ship ReVolt. The project diverges from previous contributions by focusing on synchronized data acquisition. The specialized project addresses the following tasks:

1. Installation of Velodyne Lidar and Ladybug5+ camera on ReVolt
  2. Synchronization of all exteroceptive sensors using GPS-time
  3. Extrinsic calibration of exteroceptive sensors, including transformation from Revolt's BODY-frame to NED-frame.
  4. Implementation of a detector based on a convolution neural network (CNN) for the detection of boats in camera data.
  5. Recording of time-synchronized data from sensors
  6. Analysis of the extent to which Lidar detections correspond to camera detections and vice versa
-

---

# Abstract

Sensor fusion is a key component for providing autonomous surface vehicles and ships with better situational awareness of its environment. Exteroceptive sensors such as lidar and camera provide limited information by themselves, but when combined creates a system resembling an eye, distinguishing colors, features and distances. Fusing sensory data requires it to be captured and processed within the same short time-frame to accommodate the dynamics of the system and its environment. Synchronization of sensory data is therefor crucial for sensor fusion to work properly.

In this specialization project, a system for synchronization of a Velodyne VLP-16 lidar and a Flir Ladybug5+ camera is implemented using NMEA-183 strings and PPS provide by a Hemisphere VS330 GNSS. Accuracy of timestamping using GPS-time and the system clock is compared. Sensor models for both lidar and camera are presented. A possible method for extrinsic calibration of lidar and camera is presented and the transformation-matrices from sensor to NED-frame are formulated. Two convolutional neural networks for object detection are integrated. Extensive experiments in maritime environments onboard the ReVolt model-scale vessel are conducted providing a significant amount of sensory data for further analysis and research.

The experiments demonstrated that synchronization using GNSS worked as intended, providing each exteroceptive sensor with accurate timestamping using GPS-time. Using the system clock provided acceptable results as well when synchronized to an external time-source, however it might not be an optimal solution given its requirement for continuous wireless connection. The camera used in this project was discovered to have an unacceptable transmitting-delay. The cause of this delay is yet to be investigated.

---

# Preface

This project report presents the work done during the course TTK4550. I would like to thank my supervisors, Edmund Førland Brekke at NTNU and Geir Hamre at DNV GL for their input, knowledge and support throughout the entire project. In addition, I would like to thank Tor Arne Pedersen for lending his time and support during testing of the developed system. Lastly I would like to thank Stefano Bertelli, Terje Haugen and everyone at the mechanical workshop at the Department of Engineering Cybernetics(ITK) for providing me with tools and advice when I needed it the most.

# Table of Contents

<b>Abstract</b>	<b>i</b>
<b>Preface</b>	<b>ii</b>
<b>Table of Contents</b>	<b>iv</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>viii</b>
<b>Abbreviations</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Review of Previous Work . . . . .	1
1.3 Contributions . . . . .	2
1.4 Report Outline . . . . .	3
<b>2 Background, Theory and Sensor Models</b>	<b>5</b>
2.1 Sensor fusion . . . . .	5
2.1.1 Introduction . . . . .	5
2.1.2 Method . . . . .	5
2.1.3 Timing constraints . . . . .	6
2.2 Camera . . . . .	6
2.2.1 Pinhole Model . . . . .	6
2.3 Spatial Data Acquisition using 3D lidar . . . . .	9
2.3.1 Lidar sensor model . . . . .	10
2.4 Computer Vision . . . . .	11
2.4.1 Convolutional neural nets . . . . .	12
2.5 Synchronization . . . . .	14
2.5.1 Time sources . . . . .	14

---

2.5.2	Time-disparities . . . . .	14
2.5.3	Time delays . . . . .	15
2.6	Robotic Operating System . . . . .	15
2.6.1	Roscore . . . . .	16
2.6.2	Rosbag . . . . .	16
<b>3</b>	<b>Implementation</b>	<b>17</b>
3.1	Hardware and Pipeline . . . . .	17
3.1.1	Sensor Integration . . . . .	17
3.1.2	ROS implementation, drivers and packages . . . . .	20
3.2	Calibration . . . . .	21
3.2.1	Camera calibration . . . . .	21
3.2.2	Lidar-Camera Calibration . . . . .	21
3.3	Transformation to a common world frame . . . . .	22
3.3.1	Body frame . . . . .	22
3.3.2	Local NED frame . . . . .	22
3.3.3	Estimation of position . . . . .	24
3.4	Synchronization . . . . .	24
3.4.1	Setup . . . . .	24
3.4.2	GPS-time . . . . .	25
3.4.3	Ros-time . . . . .	26
3.5	Visual Detection . . . . .	26
3.5.1	YOLOv3 . . . . .	26
3.5.2	SSD . . . . .	26
<b>4</b>	<b>Experiments and Data Collection</b>	<b>29</b>
4.1	Setup . . . . .	29
4.2	Performed Experiments . . . . .	29
4.2.1	Day 1 - Object detection and range measurements . . . . .	29
4.2.2	Day 2 - GPS-Synchronization and point estimation . . . . .	30
4.2.3	Interference - Noise from USB3 . . . . .	32
<b>5</b>	<b>Results and Discussion</b>	<b>35</b>
5.1	Visual detection . . . . .	35
5.1.1	Quality . . . . .	35
5.1.2	Distance . . . . .	35
5.2	Synchronization . . . . .	36
5.3	Experiments to be followed up . . . . .	37
5.4	Quality of recorded sensory data . . . . .	38
<b>6</b>	<b>Conclusion and Future Work</b>	<b>39</b>
6.1	Conclusion . . . . .	39
6.2	Future work . . . . .	39
	<b>Bibliography</b>	<b>40</b>

---

# List of Tables

2.1	Real-time constrains . . . . .	6
2.2	Velodyne VLP-16 Specifications . . . . .	10
3.1	Ladybug5+ specifications . . . . .	18
3.2	On-board laptop specifications . . . . .	20
3.3	WGS-84 parameters . . . . .	23
3.4	NMEA-messages . . . . .	25
5.1	Synchronization delays . . . . .	37



---

# List of Figures

2.1	Pinhole model . . . . .	7
2.2	Central-projection model . . . . .	8
2.3	Radial distortion . . . . .	8
2.4	Velodyne VLP-16 Lidar . . . . .	9
2.5	Spherical data from lidar . . . . .	11
2.6	Top and sideview of the spherical coordinate system used by the lidar[40]	11
2.7	Kernel in neural networks . . . . .	12
2.8	Structure of convolutional neural networks . . . . .	13
2.9	YOLOv3 Pipeline . . . . .	13
2.10	Faster-RCNN pipeline [32] . . . . .	14
2.11	Illustration of stratum sources . . . . .	15
2.12	Ros communication . . . . .	16
3.1	The ReVolt model ship . . . . .	17
3.2	General Purpose I/O Connector for the ladybug5+ . . . . .	18
3.3	Interface box for VLP-16 lidar . . . . .	19
3.4	Positioning of exteroceptive sensors implemented during this project . .	20
3.5	Pitch, roll, yaw rotations of the BODY frame . . . . .	23
3.6	Overview of system clocks . . . . .	24
3.7	Signal overview for synchronization . . . . .	25
3.8	Labeling of dataset from Nidevla . . . . .	27
4.1	Experimental setup . . . . .	30
4.2	Gunnerus Workboat . . . . .	31
4.5	Test area 2 . . . . .	31
4.3	Test area 1 . . . . .	32
4.4	Detection test . . . . .	32
4.6	Synchronization and point estimation testing . . . . .	33
4.7	On land testing . . . . .	33
5.1	Detections using SSD . . . . .	36

---

5.2	Lidar scan of Nidevla . . . . .	36
5.3	Comparison between ladybug and lidar . . . . .	37
5.4	Panoramic image by ladybug5+ . . . . .	38

---

# Abbreviations

ASV	=	Autonomous Surface Vehicle
USV	=	Unmanned Surface Vehicle
AAWA	=	Advanced Autonomous Waterborne Applications
MP	=	Mega Pixels
FOV	=	Field Of View
PPS	=	Pulse Per Second
TOF	=	Time Of Flight
GNSS	=	Global Navigation Satellite System
AHRS	=	Attitude and Heading Reference System
IMU	=	Inertial Measurement Unit
RTK	=	Real-Time Kinematic
FPS	=	Frames Per Second
TCP	=	Transmission Control Protocol
UDP	=	User Datagram Protocol
ROS	=	Robot Operating System
SDK	=	Software Development Kit
RTC	=	Real Time Clock
NMEA	=	National Marine Electronics Association
NED	=	North-East-Down
HOG	=	Histogram of Gradients
SSD	=	Single Shot Detector
YOLO	=	You Only Look Once
DP	=	Dynamic Positioning
LIDAR	=	Light Detection and Ranging

---

# Chapter 1

## Introduction

Autonomous ships are expected to be used in maritime industry within 10 years[11]. Leading this development is organizations with projects within Autonomous surface vehicle(ASV) / Unmanned Surface Vehicle(USV). The last decade has seen vast improvements within the ASV/USV industry, and it does not seem to be slowing down.

### 1.1 Motivation

As a leading organization within ship classification it is essential that DNVGL understands new technology introduced to the maritime sector. Their participation in Advanced Autonomous Waterborne Applications (AAWA) partnership supports their continuous commitment to this. “Autonomous shipping is the future of the maritime industry. As disruptive as the smart phone, the smart ship will revolutionise the landscape of ship design and operations”[23].

The Revolt project is a result of this commitment. In collaboration with NTNU, Kongsberg and Maritime Robotics, the Autosea project[16] was founded with the main goal of developing methods for guidance and navigation for autonomous ships. Sensor fusion is part of the foundation for these tasks, providing improved situational awareness to be used in navigation and control. This requires synchronized sensory data in real time.

### 1.2 Review of Previous Work

This specialized project is a continuation of previous work done on Revolt. Much of the research done on Revolt have had an element which involved synchronization or equipment used for this purpose.

- Kamsvåg [24] developed a Ros-based architecture for camera-lidar sensor fusion and tracking on the Revolt model ship, where lidar was the primary sensor. DBSCAN was used to cluster lidar-data and performed within expectations. Object detection

was done using Faster R-CNN with a previously trained model provided by Tangstad [36]. Synchronization of sensory data was done in two steps. The camera was triggered by the arrival of TCP-packets from scans by the lidar. Secondly, timestamping of data was done on arrival using ROS time. Lidar managed to track targets with satisfactory results within 10-50 meters. Faster-RCNN had limited range, with few detections passed 20 meters.

- Alfheim et al. [3] proposed a dynamic positioning (DP) system for the Revolt model ship. It used high quality navigational sensors already integrated on Revolt using ROS architecture. One of the navigation sensors provided high quality GNSS-data on NMEA-183 format and options for synchronization using GPS-time.

A significant amount of research on synchronization and sensor fusion has been done the last two decades. However, finding relevant research in sensor fusion which provides detailed description of synchronization of sensory data has proved to be difficult. Especially research that uses the same type of sensors used in this project. Two articles provide some information:

- Schneider et al. [35] developed a system for accurate synchronization of camera and lidar in a dynamic system. The system also provides a solution for occlusion given different viewpoints of the fused sensors. By examining the bearing information provided by lidar-packets, it is possible to estimate the duration of a Lidar revolution and by that providing a trigger signal for synchronizing the camera.
- Albrektsen et al. [1] developed a timing board that accurately records Time of Validity from sensors. A wide variety of sensors have been successfully integrated with the board. It is also capable of triggering cameras to synchronize with GNSS, providing accurate time and position data to images.

Previous work shows that there is a possibility for precise timestamping using GNSS on Revolt. Possible sources for time-delays are also well documented.

## 1.3 Contributions

The main contributions in this specialized project are:

1. Installation of the Lidar and Camera on the Revolt model ship.
2. Synchronization of all exteroceptive sensors using GPS-time and PPS from the Hemisphere VS330 GNSS.
3. Integrating YOLOv3 provided by Grini[42] and SSD for object detection of maritime vessels.
4. Comprehensive experiments of lidar and ladybug5+ camera on Revolt.

## 1.4 Report Outline

Chapter two gives a brief introduction to relevant theory and principles used throughout the project. This includes an introduction to sensor fusion, sensor models, kernels and feature extraction in convolutional neural networks, synchronization and the Robotic Operating System.

Chapter three describes the integration of sensors on the existing system on Revolt, both hardware and software. A possible calibration procedure for extrinsic parameters. The different transformation-frames derived during this project. The process for synchronizing sensory data and last but not least the visual detection pipeline.

Chapter four provides the experimental setup used during testing. It also goes into detail of each experiment performed during this project.

Chapter five presents and discusses the results of performed experiments, this includes the detection pipeline, synchronization-test and interference by USB3. Experiments to be followed up is also mentioned in this chapter.

Chapter six concludes this specialization project and suggests new problems to be investigated





# Background, Theory and Sensor Models

The aim of this chapter is to provide the theoretical principles and theory used throughout the report. This also includes an introduction to the sensors used in this project and their respective sensor models.

## 2.1 Sensor fusion

### 2.1.1 Introduction

Physical systems is bound to have uncertainty in regards to behavior and dynamics. In some cases, this uncertainty can lead to dangerous situations. Sensors, both external and internal are used to combat this uncertainty, by providing information of environmental or internal dynamics.

Unfortunately there does not exist a sensor that does it all, providing a wide span of information needed to create a complete picture that gives the machine full situational awareness. There do however exists methods to decrease uncertainty, by combining sensory data from several sources/sensors such that the resulting data creates a better sense of awareness than if the sensor data was used individually [13]. This process is called sensor fusion.

### 2.1.2 Method

A good foundation is necessary before sensor fusion is possible / delivers reliable results. The project goal is to provide proper timestamping to sensory data, minimizing time-disparities, as well as creating an object detection pipeline with the goal to provide reliable

detections in a real-time setting for a high dynamic system. Given the practical aspect of this projects, it is crucial to use correct methods and have a cohesive project plan. It can be summarized by these points:

1. Evaluate the time-constraints of the system, choose hardware and algorithms accordingly. The dynamics of the system affects its time-constraints.
2. Use a common time-reference for the relevant sensors used during sensor fusion. This can be the system clock synchronized with an external source or internal clocks on sensors synchronized to GPS-time from a GNSS. A possible different approach is synchronization through triggering as mentioned in [35] and [1], or passive synchronization discussed in [26].
3. Proper calibration of both intrinsic and extrinsic parameters. Extrinsic parameters / poses between sensor frames are used to project sensory data between different frames.

### 2.1.3 Timing constraints

Systems with real-time constraints can categorize their time-constraints into three categories based on the consequences for failing to deliver on time. This categorization can be seen in table 2.1.

**Table 2.1:** The terminology for timing constraints in real-time systems

Constraints	Consequence for missing deadline
Soft real-time constraints	Quality of data degrades, reduces quality of service
Firm real-time constraints	Data is no longer considered useful, reduces quality of service
Hard real-time constraints	System failure

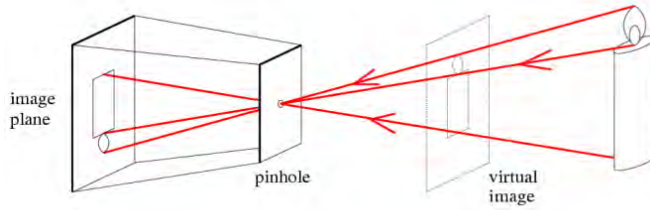
Firm real-time constraints is most relevant for this project. A sensor missing its deadline for providing data to be fused reduces the overall quality for that iteration of sensor fusion, but does not necessary lead to system failure.

## 2.2 Camera

### 2.2.1 Pinhole Model

A camera is essentially projecting 3D-points onto a 2D plane. The pinhole model describes this relationship mathematically, and proves to be a good approximation to the behavior of a camera. This section is large based on [20].

As illustrated in figure 2.1 the pinhole model inverts the image. By picturing a image plane in front of the pinhole one avoids working with an inverted image.



**Figure 2.1:** The pinhole model

### Intrinsic and Extrinsic parameters

As seen in figure 2.2 the mapping from Euculean 3-space  $\mathbb{R}^3$  to Euculean 2-space  $\mathbb{R}^2$  in the image plane can be described as

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T \quad (2.1)$$

Where  $f$  is the focal length, which is the distance from camera origin  $C$  too the principal point  $Z = f$ , where the  $Z$ -axis/principal axis intersects the image plane. This gives the following linear mapping / matrix in homogeneous coordinates

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.2)$$

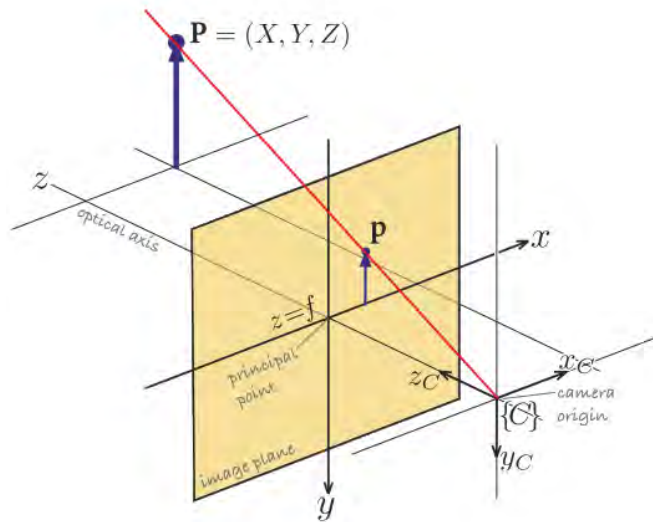
This mapping does not take into account that the origin of coordinates in the image plane might not be at the principal point. This general mapping is added

$$(X, Y, Z)^T \mapsto (fX/Z + p_x, fY/Z + p_y)^T \quad (2.3)$$

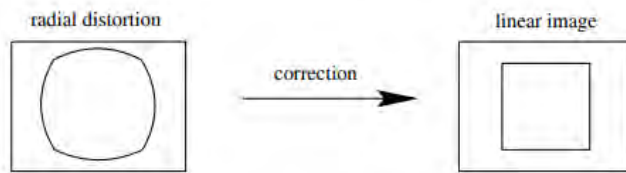
In homogeneous coordinates this becomes equation 2.4 where equation 2.5 is the camera matrix that describes the intrinsic/internal parameters. The matrix is then customized to CCD cameras, show in equation 2.6, where  $s_x$  and  $s_y$  is the pixel size.  $c_x$  and  $c_y$  is the center of the image.

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.4)$$

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$



**Figure 2.2:** "The image plane is at a distance  $f$  in front of the camera's origin, on which a noninverted image is formed. The camera's coordinate frame is right-handed with the  $Z$ -axis defining the center of the field of view" as described in [5]



**Figure 2.3:** Example of radial distortion, figures are taken from [20]

$$\mathbf{K} = \begin{bmatrix} f/s_x & 0 & c_x \\ 0 & f/s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.6)$$

The extrinsic parameters describes the pose/coordinate system of the camera in relation to a known/fixed coordinate system. These parameters can be estimated using a process proposed in chapter 3.2.

### Radial distortion

The model for radial distortion is given in equation 2.7 where  $(\tilde{x}, \tilde{y})$  is the image position that obeys linear projection.  $(x_d, y_d)$  is the image position after radial distortion.  $\tilde{r}$  is the

radial distance  $\sqrt{\tilde{x}^2 + \tilde{y}^2}$  from the centre for radial distortion.  $L(\tilde{r})$  is the distortion factor, which is a function of radius  $\tilde{r}$ [20].

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\tilde{r}) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} \quad (2.7)$$

Equation 2.8 corrects this distortion, where  $(x, y)$  are the measured coordinates,  $(\tilde{x}, \tilde{y})$  are the corrected coordinates, and  $(x_c, y_c)$  is the center of radial distortion, with  $r^2 = (x - x_c)^2 + (y - y_c)^2$ [20].

$$\hat{x} = x_c + L(r)(x - x_c) \quad \hat{y} = y_c + L(r)(y - y_c) \quad (2.8)$$

### Translation and rotation

When combining data from exteroceptive sensors such as the lidar and camera, the extrinsic parameters between their respective frames/coordinate systems must be derived for the data to be combined and used correctly. Here the position and orientation of frame  $b$  relative to frame  $a$  is given by the homogeneous transformation matrix [12].

$$\mathbf{T}_b^a = \begin{pmatrix} \mathbf{R}_b^a & \mathbf{r}_{ab}^a \\ \mathbf{0}^T & 1 \end{pmatrix} \in SE(3) \quad (2.9)$$

Where  $R_b^a$  is the rotational matrix,  $r_{ab}^a$  is the translation, and  $0^T$  is the zero-matrix.



**Figure 2.4:** Illustration of the Velodyne VLP-16 Lidar. Picture is taken from [38]

## 2.3 Spatial Data Acquisition using 3D lidar

The lidar used in this project is the Velodyne VLP-16 puck as shown in figure 2.4. It provides a full 360°3D scan of its environment in real time, returning not only distance measurements but also the intensity of reflections. The full list of specifications can be seen in table 2.2. The lidar uses an array of 16 infra-red lasers paired with detectors[39]. Mounted vertically at 2°interval they create a field of view (FOV) equal to  $\pm 15.0^\circ$  from the center as shown in figure 2.6b. It supports three different return modes: **Strongest**, **Last** and **Dual** which can be configured via the sensor's web interface.

**Table 2.2:** Velodyne VLP-16 Specifications

Model	VLP-16
Sensor	16 Channels
Measurement Range	100 m
Range Accuracy	Up to $\pm 3$ cm
Field of View (Vertical)	$30^\circ, \pm 15.0^\circ$
Angular Resolution (Vertical)	$2.0^\circ$
Field of View ( Horizontal)	$360^\circ$
Angular Resolution(Horizontal / Azimuth)	$0.1 - 0.4^\circ$
Rotation Rate	5-20 Hz
Precision Timestamps	&GPRMC or &GPGGA strings and PPS
Weight	830 g

### 2.3.1 Lidar sensor model

Each reflected point is represented in a spherical coordinate frame with radius  $R$ , elevation  $\omega$  and azimuth  $\alpha$ , see figure 2.5 and 2.6. This can be converted to a Cartesian coordinate frame using trigonometry[40] as shown in equation 2.10.

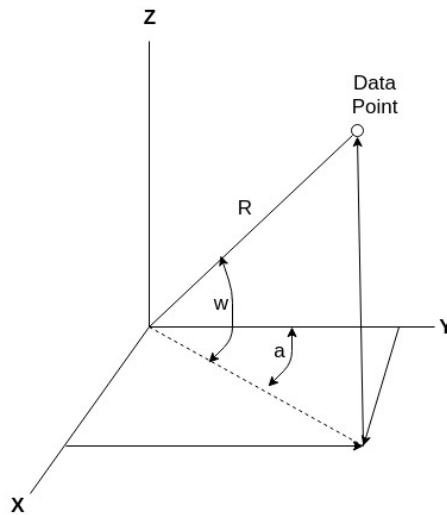
$$\begin{aligned} X &= R \cos \omega \sin \alpha \\ Y &= R \cos \omega \cos \alpha \\ Z &= R \sin \beta \end{aligned} \tag{2.10}$$

The lidar uses time-of-flight(TOF) methodology. The range  $R$  is given by equation 2.11 where  $c$  is the speed of light,  $t_t$  and  $t_r$  is the time of transmitted and received pulse.

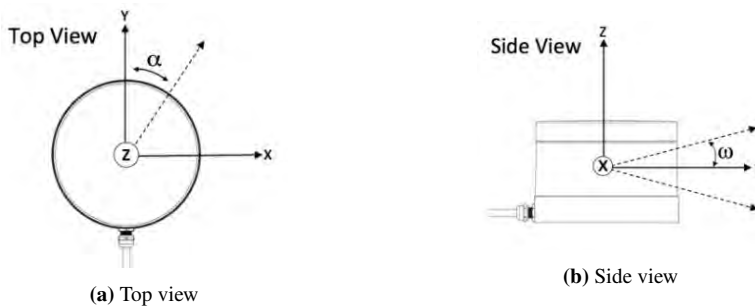
$$R = \frac{c}{2} (t_r - t_t) \tag{2.11}$$

$$\mathbf{P}_{ij}^l = \begin{bmatrix} R_{ij} \cos \omega_i \sin \alpha_j \\ R_{ij} \cos \omega_i \cos \alpha_j \\ R_{ij} \sin \omega_i \end{bmatrix} \tag{2.12}$$

Each point in a pointcloud  $\mathbf{P}^l$  can be represented with equation 2.12 where  $\omega_i$  and  $\alpha_j$  are the vertical and horizontal angel of the fired laser pulse.



**Figure 2.5:** Each datapoint is given in spherical coordinates (radius  $R$ , elevation  $w$ , azimuth  $\alpha$ ) and needs to be converted to Cartesian coordinates. Note: When a laser pulse doesn't result in a measurement, such as when a laser is shot skyward, both distance and reflectivity value will be 0 as described in [40]



**Figure 2.6:** Top and sideview of the spherical coordinate system used by the lidar[40]

## 2.4 Computer Vision

Computer vision has seen rapid improvements the last decade. Deep learning has introduced new methods/algorithms for object detection and recognition with vast improvements in real-time performance and accuracy. This section will focus on convolutional neural networks and assumes that the reader has basic understanding of deep learning in computer vision.

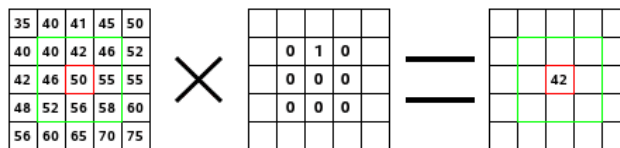


## 2.4.1 Convolutional neural nets

How does one extract information/features from an image? For humans this process seems intuitional and we manage to distinguish objects with similar characteristics easily in real time. However for computers this task is quite the challenge. In traditional computer vision, methods such as histogram of oriented gradients (HOG)[7] would be used to create feature descriptors. By comparing feature descriptors with known descriptors HOG is able to detect objects. This method is quite computational heavy, not suited for real time purposes and not end-to-end trainable. End-to-end trainable means that the entire system can be trained in one operation. This is where convolutional neural nets is a big leap in the right direction. Given the right properties, it can be considered as a feature extractor and classifier, with both real-time applications and considered end-to-end trainable.

### Feature extraction

How does convolutional neural networks extract features? The solution goes by many names, such as filters, kernels, sliders. Each pixel in an RGB-image is represented by three numbers between 0 and 255. One colour of the entire picture can be illustrated as an array with these pixel-values.



**Figure 2.7:** The convolution-operation, referred to as kernel or filters in neural networks. Illustration is given by [18]

The kernel operation is done by sliding a window of size  $N \times N$  depending on the filter-size  $N$  over the entire picture as illustrated in figure 2.7. Each pixel-value is multiplied by the corresponding filter-value and summed up. The filters can be seen as feature-detectors. Usually each convolution layer consists off several filters, where each one detects a different feature in an image.

### Real time performance

Real time performance is determined by several factors, but there is especially two that should be mentioned. The first one is the structure of the network. Secondly, the detection pipeline.

The size obviously plays a big role in real-time performance. The more layers the network consist off, the more computational heavy the operation is. Regarding the detection pipeline there is two examples that illustrates this well, which is the differences between You Only Look Once(YOLO)v3[31] and Faster-RCNN[32].

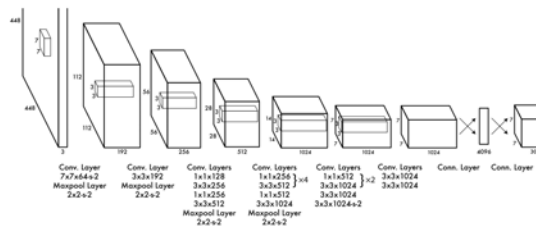


Figure 2.8: Structure of YOLOv3 [30]

YOLOv3 uses one network for the entire detection pipeline. This is illustrated in figure 2.9 where the bounding boxes and confidence score of said bounding boxes are computed in parallel with class probability. These are then combined and presented as final detections which has reached a set threshold.

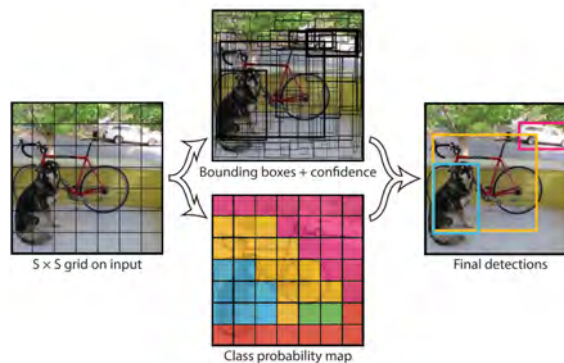
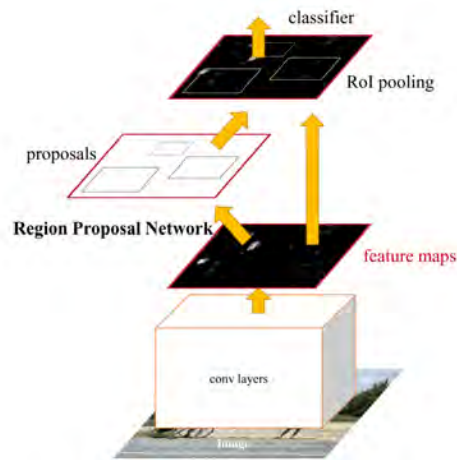


Figure 2.9: YoloV3 Pipeline[30]

Faster-RCNN diverges from this approach by using a region proposals network, essentially using two networks instead of one in the detection pipeline. The pipeline illustrated in figure 2.10 and specifically the region proposal network uses the feature-map given by earlier convolution layers to predict possible locations for bounding boxes. These proposal are sent to the classifier and evaluated.



**Figure 2.10:** Faster-RCNN pipeline [32]

Having only one network for the entire detection pipeline gives YOLOv3 an advantage regarding real-time performance. However, Faster-RCNN has proven to give more reliable detections. For more in-depth information the reader is advised to read [30].

## 2.5 Synchronization

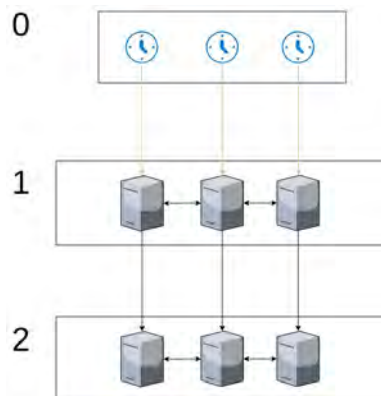
In this report synchronization refers to precise time-stamping of sensory data.

### 2.5.1 Time sources

A time-reference is needed when time-stamping sensory data. The quality of this reference can be described by the terminology "Stratum source" which is a hierarchic system from 0 to 16 where 0 is the most accurate time-source representing GPS-time. Time-servers connected to a stratum 0 source are referred to as stratum 1 sources and so on. From this it can be concluded that using time-references with a high stratum source increases the probability of introducing a greater disparity between the reference and true GPS-time.

### 2.5.2 Time-disparities

"When Kalman filter measurements compare the outputs of two different navigation systems, it is important to ensure that those outputs correspond to the same time of validity. Otherwise, differences in the navigation system outputs due to the time lag between them will be falsely attributed by the Kalman filter to the states, corrupting the estimates of those states. The greater the level of dynamics encountered, the larger the impact of a given time-synchronization error will be." [19]



**Figure 2.11:** Illustration of the hierarchic structure of stratum sources. The clocks represents gps-clocks. The computers are time-servers.

Time of validity has the same importance as in the given example when fusing sensory data. Revolt can be described as a system which experiences high levels of dynamics. Waves is one example where both roll and pitch can experience this. If there are high levels of time-disparities with what is assumed to be synchronized data, then translating sensory data between sensor-frames will introduce translation/transformation errors.

### 2.5.3 Time delays

Retrieving sensory data from sensors includes several steps that might introduce time-delays. Some examples of this are: The time-difference between time of capture and transmitting data from a sensor. Finite transmitting-speeds from sensor to computer through some data-transfer-interface. Continuously polling data rather than reacting when receiving. Time-stamping when receiving data rather than at the time of capture[1]. All these are possible contributors to a sum of delay that might affect the performance of the system, depending on its dynamics and requirements.

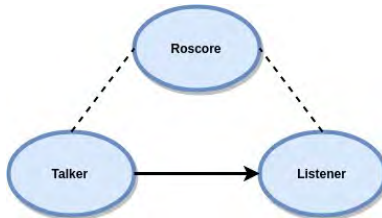
## 2.6 Robotic Operating System

This section is largely based on [29]. Robotic Operating System, commonly referred to as ROS is an open source framework that provides libraries and tools to help software developers create robot applications. This includes drivers, libraries, visualizers, package management and more[27]. The main philosophy of ROS is decoupling and re-usability. Ros systems usually consist of several small computer programs that are connected to each other, communicating through message-passing. These programs can be written in the language which provides the best efficiency and solution for the specific task, without problems. This multilingual approach makes it that much easier to accomplish high-productivity. Each program or software module is referred to as a node, emphasizing that one program is part of a larger system.

## 2.6.1 Roscore

Roscore provides connection information to nodes. This is used by nodes to form a direct peer-to-peer connection with other nodes publishing and subscribing to the same topics. See the following command for how to use it:

```
$ user@hostname rosrn
```



**Figure 2.12:** This illustrates the ephemeral connection between Roscore and nodes, initializing peer-to-peer communication as well as message-passing using topics between nodes.

## 2.6.2 Rosbag

Rosbag is a tool used for recording and replaying published messages on topics. Its a great tool, especially when debugging software. Experimental sensory data can be recorded and used during development to test new algorithms. See the following commands for recording of three or all topics respectively.

```
$ user@hostname rosbag record topic1 topic2 topic3
```

```
$ user@hostname rosbag record -a
```

# Implementation

This chapter provides detailed descriptions of implementations done throughout the specialization project.



Figure 3.1: The ReVolt model ship [11] in 2016

## 3.1 Hardware and Pipeline

### 3.1.1 Sensor Integration

#### Ladybug5+

The camera used in this project is the Flir Ladybug5+. It consists of 6 5MP imaging sensors, providing a FOV close to 90% of a full sphere. Specifications can be seen in table 3.1. With its FOV and variety of software configurations, it provides a good foundation for being used in sensor fusion. Its internal clock can be synchronized using PPS and NMEA-strings, providing timestamps with accuracy close to  $20 \mu\text{s}$ [21]. The camera is delivered calibrated, which removes the uncertainty/error of manual calibration.


The ladybug5+ have two different ways of delivering its captured images. By using the provided software in Windows (LadybugCapturePro), where image formatting and

**Table 3.1:** Flir Ladybug5+ specifications, given in datasheet[15]

Model	LD5P-U3-51S5C-B
Imaging sensor	6 x Sony IMX264 2/3", 3,45 $\mu$ m
Resolution	2448x2048 at 30 FPS 2448x1024 at 60 FPS
Global shutter	Yes
Field of View	90% of full sphere
Digital Interface	USB3
Weight	3 kg
Precision Timestamps	RS232 GPS NMEA strings and PPS over GPIO

stitching is done automatically. The other choice is using the provided library in Ubuntu 16.04. It provides "image capture only". Image formatting and stitching is not performed, and each picture is delivered individually. ReVolt is based on Ubuntu, so the last option was chosen for this project.

The sensor is powered by 12-24V from one of two voltage regulators provided and installed with assistance from DNV GL. This source is connected to the ladybug using the I/O connector shown in figure 3.2. The sensor is interfaced with the on-board laptop using USB3 for data transfer. Each of the six camera-sensors is set to capture 2 frames per second(FPS), resulting in a total of 12 FPS combined. Each image is originally 2464x2048 but is resized down to 30% of original size, giving an image size of 739x614. The image is captured sideways by default, so a rotation of the image is necessary to get it on the correct form. Each image is captured in raw8 format.

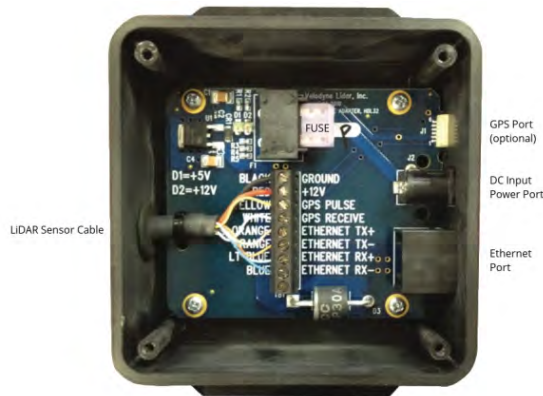


Green	1	OPTO_GND	Ground for opto-isolated IO pins
Blue	2	I0	Opto-isolated input (default Trigger in)
Brown	3	O1	Opto-isolated output
Orange	4	IO2	Input/Output / GPS data
White	5	+3.3 V	Power external circuitry up to 150 mA
Black	6	GND	Ground for bi-directional IO, V <sub>EXT</sub> , +3.3 V pins
Red	7	V <sub>EXT</sub>	Allows the camera to be powered externally
Red	8	V <sub>EXT</sub>	Allows the camera to be powered externally
Red	9	V <sub>EXT</sub>	Allows the camera to be powered externally
Green	10	OPTO_GND	Ground for opto-isolated IO pins
Yellow	11	IO3	Input/Output / PPS signal
Black	12	GND	Ground for bi-directional IO, V <sub>EXT</sub> , +3.3 V pins

**Figure 3.2:** Four of the 12 provided pins were used in this project. These were pin 4 for gps-data, pin 12 for pps-pulse and pin 6 & 7 for power and ground respectively.[15]

### Velodyne VLP-16

The velodyne VLP-16 lidar comes with its own interface box, providing convenient connections for power, Ethernet and GPS inputs. It also provides protection from power irregularities. The interface box and therefore the lidar is powered by 12V and up to 3.0A from the second voltage regulator provided and installed with assistance from DNV GL. The rotation rate is set to 10Hz, providing a full 3D scan every 0.1 second. The distance measurement is given by the strongest reflection. Data packets are transferred over TCP.



**Figure 3.3:** The interface box showing the different connections available[40]

### Navigational sensors

ReVolt is equipped with an Xsens MTI-G-710, which is an industrial grade miniature GNSS-aided, IMU-enhanced GNSS/INS and AHRS, providing high-quality inertial measurements[43]. Precise heading and real-time kinematic positioning is given by the Hemisphere VS330 GNSS compass with resulting heading and position accuracies of  $\pm 0.2$  and  $\pm 1$ cm[3]. Both sensors can provide accurate PPS-signals and NMEA-messages. These are connected to the on-board computer, a Tank-720 high performance embedded system. It handles data from both sensors and the existing control-system used on ReVolt.

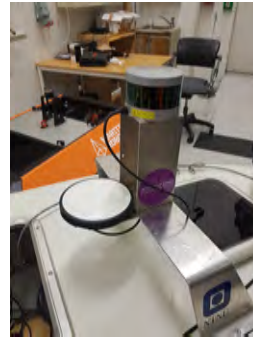
ReVolt has two targa hoops mounted at both ends of its deck. They were installed by the mechanical workshop at the Department of Engineering Cybernetics(ITK) at NTNU during previous projects on ReVolt [2]. The front hoop has a built in mounting mechanism for the velodyne VLP-16 lidar from previous implementations and was used for the same purpose during this project. Both hoops are also used for mounting the two antennas used by the Hemisphere VS330 GNSS. With limited space on the vessels deck, it was concluded to place the remaining sensor, the ladybug5+, at the forefront of ReVolts deck as shown in figure 3.4a. This position though not optimal given the obstruction by the two hoops and antennas, provided enough FOV for this project. In an optimal situation one would want the camera to have full 360°unobstructed view, however the lidar with its range measurements was prioritized and therefor given full 360°FOV. The mounting mechanism for the ladybug camera was also provided by the mechanical workshop at the



Department of Engineering Cybernetics(ITK) at NTNU. It should be mentioned that it was not possible to raise the sensor higher, since it would obstruct the forward view of the Velodyne VLP-16 lidar.



(a) Positioning of ladybug5+



(b) Positioning of Velodyne VLP-16 lidar

**Figure 3.4:** Positioning of exteroreceptive sensors implemented during this project

ReVolt was further equipped with an on-board laptop, specifications given in table 3.2. This was used to handle the two new sensors integrated with the existing system, given the high bandwidth from each sensor and limiting processing power on the on-board computer.

**Table 3.2:** On-board laptop specifications

Model	HP zbook 15
CPU	Intel Core i7-4800MQ 8x2.7 GHz
Memory	8 GM RAM
Graphics	Quadro K2100M/PCIe/SSE2
Storage	250 GP Solid State HDD
Operating System	Linux Ubuntu 16.04 LTS (Xenial)

### 3.1.2 ROS implementation, drivers and packages

The drivers for each sensor are based on ROS packages developed by the ROS community. The driver for the camera is the *pointgrey\_ladybug* driver from rpng[34], which uses the ladybug SDK to interface with the camera. This driver is an expansion of the Autoware driver[6], giving more options and support for the ladybug5+ camera. Flycapture2 SDK must be uninstalled for this driver to function properly. The driver allows the user to control several parameters through the accompanying ROS launch file, including framerate, shutter time and gain. Each captured set of images consist off 6 individual images, which are then published onto the ROS network as six individual topics, on the form: *ladybug/camera[0-5]/image\_raw*. All this is done in one single node. The driver for the Velodyne VLP-16

lidar is *velodyne\_driver*[33]. This driver receives raw data-packets over TCP and converts it to a point cloud given the transformation in 2.12, section 2.3.1. The point cloud also includes the intensity measurements from the reflected pulses.

## 3.2 Calibration

Intrinsic and extrinsic parameters were introduced in section 2.2.1. The intrinsic parameters must be known to correctly map 3D-points to a virtual image. The extrinsic parameters describes the rotation and translation between different poses/frames and is therefor required for relating sensory data from one sensor to another. See section 2.2.1.

Technical difficulties were prevalent during several steps in this project, which ultimately affected the calibration-process. Therefor there will only be given a short introduction to the calibration-process, the resulting transformation matrices and an explanation of the technical difficulties.

### 3.2.1 Camera calibration

The camera is said to be uncalibrated if the K-matrix, see equation 2.6, is not known. For this setup the K-matrices are known, since Point-Grey/Flir provide their cameras fully calibrated from factory. Using either provided software or library functions these matrices can be extracted from the cameras.

The images captured by the ladybug suffers from barrel distortion described in section 2.2.1. The provided software on Windows deals with radial distortion automatically, but the chosen camera-driver does not provide this functionality. Ladybug SDK is most suited for using .pgr stream files, which the current driver avoid using. It can be argued that drivers from Autoware and rpng does not use this filetype because of their use in high dynamic systems such as self-driving vehicles and that using stream files introduces unacceptable delays as described in section 2.5. It was attempted to extract the distortion-parameters from the calibration-file, an attempt which proved to be unsuccessful. Flir does not provide documentation for their calibration-files.

Through contact with Flir's technical department, it was established a possible solution for the problem. There exist two functions in Ladybug SDK, *ladybugUnrectifyPixel* and *ladybugRectifyPixel*, that might provide the necessary functionality for undistorting previously recorded data or data captured during run-time.

### 3.2.2 Lidar-Camera Calibration

When combining sensory data from several sources/sensors the rigid body transformation between the respective sensors must be found. This transformation can be found through a method proposed by An. Dahl, using 3D-3D correspondences as described in [10]. The git-repository for this implementation can be found at [9]. This was not completed during this project given technical difficulties relating to undistorting of images.

### 3.3 Transformation to a common world frame

Assume the transformation from lidar to camera is known. The next step is to find the transformation from camera-frame to the BODY-frame of ReVolt. Given the transformation from camera to BODY it is possible to find the rotation and translation between the lidar-frame and local NED-frame by first finding the transformation between BODY and local-NED. Multiplying these transformations gives the final transformation-matrix from lidar-frame to NED-frame.

#### Sensor frames

The resulting transformation-matrix from section 3.2.2 will take the form given in equation 3.1. This describes the rigid body transformation from the lidar frame to the camera frame.

$$\mathbf{T}_l^c = \begin{pmatrix} \mathbf{R}_l^c & \mathbf{r}_{cl}^c \\ \mathbf{0}^T & 1 \end{pmatrix} \in SE(3) \quad (3.1)$$

#### 3.3.1 Body frame

The center of ReVolts body-frame is at the Xsens MTI-G-710 IMU, which is located at the center of ReVolt. The z-axis of its coordinate-system points towards ground since the sensor is placed upside down. The x-axis aligns with the front of ReVolt. As illustrated in figure 2.2 and the positioning of the ladybug in figure 3.4a, the Z-axis of the cameras coordinate-system aligns with the front of ReVolt. Its Y-axis points towards ground. The rotation matrix for camera to body frame is therefor equal to equation 3.2. The translation is measured by hand and is equal to equation 3.3, where the units are measured in meters. The resulting rigid body transformation is given in equation 3.4

$$\mathbf{R}_c^b = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (3.2)$$

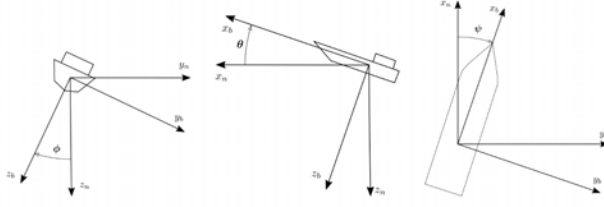
$$\mathbf{r}_{bc}^b = \begin{bmatrix} 1.5 \\ 0 \\ -0.17 \end{bmatrix} \quad (3.3)$$

$$\mathbf{T}_c^b = \begin{pmatrix} \mathbf{R}_c^b & \mathbf{r}_{bc}^b \\ \mathbf{0}^T & 1 \end{pmatrix} \in SE(3) \quad (3.4)$$

#### 3.3.2 Local NED frame

The rotation between Body-frame and North-East-Down-frame(NED) is given in [41] and [17]

$$\mathbf{R}_b^n(\Theta) = \begin{bmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + c(\psi)s(\theta)s(\phi) & s(\psi)s(\phi) + c(\psi)s(\theta)c(\phi) \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\psi)s(\theta)s(\phi) & -c(\psi)s(\phi) + s(\psi)s(\theta)c(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \quad (3.5)$$



**Figure 3.5:** Roll, pitch and yaw rotations of the BODY frame [24]. Figure is used with permission from the author.

where  $s = \sin$  and  $c = \cos$ . The NED-frame's origin  $O_n$  is defined relative to WGS-84, Earth's reference ellipsoid. GNSS delivers GPS-positions in the Earth-centered Earth-fixed (ECEF) frame. The NED-frame's origin relative to the ECEF-frame is given in longitude and latitude, hereby referred to as  $l$  and  $\mu$ . By transforming the position of the BODY-frame and local NED-frame from ECEF to x,y,z coordinates using equation 3.6, where  $N$  is equal to equation 3.7. The parameters  $r_e$  and  $r_p$  are given in table 3.3.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (N + h) \cos \mu \cos l \\ (N + h) \cos \mu \sin l \\ \left( \frac{r_p^2}{r_e^2} N + h \right) \sin \mu \end{bmatrix} \quad (3.6)$$

$$N = \frac{r_e^2}{\sqrt{r_e^2 \cos^2(\mu) + r_p^2 \sin^2(\mu)}} \quad (3.7)$$

**Table 3.3:** WGS-84 parameters[17]

Parameters	Comments
$r_e = 6378137m$	Equatorial radius
$r_p = 6356752m$	Polar axis radius

The rotation from ECEF to NED is given by [17]

$$\mathbf{R}_e^n(\Theta_{ne}) = \begin{bmatrix} -\cos(l) \sin(\mu) & -\sin(l) & -\cos(l) \cos(\mu) \\ -\sin(l) \sin(\mu) & \cos(l) & -\sin(l) \cos(\mu) \\ \cos(\mu) & 0 & -\sin(\mu) \end{bmatrix} \quad (3.8)$$

where  $\Theta_{ne} = [l, \mu]^\top$ . By subtracting the NED origin  $0_n$  in ECEF-coordinates from the resulting BODY-ECEF-coordinates and multiplying it with rotationmatrix 3.8 gives the correct position in a local NED frame. The resulting transformation matrix is shown in equation 3.9

$$\mathbf{T}_b^n = \begin{pmatrix} \mathbf{R}_b^n & \mathbf{r}_{bn}^n \\ \mathbf{0}^T & 1 \end{pmatrix} \quad (3.9)$$

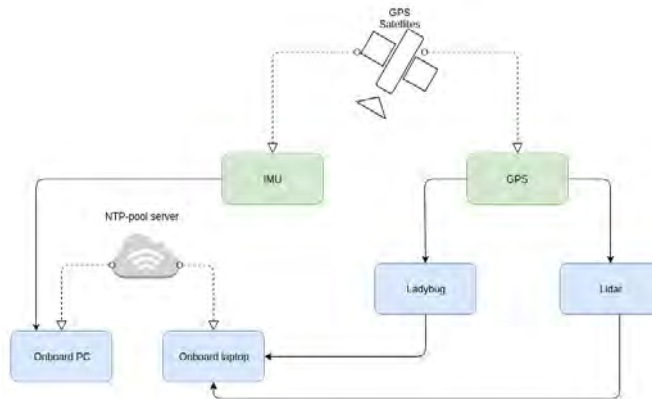
where  $r_{bn}^T$  is the translation between the origin of the local NED-frame and the BODY-frame in local NED-frame-coordinates.

### 3.3.3 Estimation of position

The local NED-frame is usually set after initialization of the navigation-system. After this the NED-coordinates of the BODY-frame is continuously given by the hemisphere. Inaccuracies in position of the BODY-frame relative to the local NED-frame will affect the projection of sensory data from sensor frames to the NED-frame negatively. It is therefor important to consider the necessity for estimating ReVolts position using navigation-data. No further estimation is done in this project based on the assumption that the navigation-data provided by the Hemisphere VS330 is of such quality that further estimation is of no use.

## 3.4 Synchronization

There are six clocks on ReVolt that need to be synchronized. The first two are the real-time clocks (RTC) on the onboard computer and laptop. The next two are the internal clocks on each of the exteroreceptive sensors introduced in section 3.1.1 and 3.1.1. The last two are the IMU and GPS-receiver. These two can provide GPS-time, which is considered a stratum 0 source. Since the last two provided GPS-time, they can be consider synchronized. That leaves four remaining clocks to be synchronized. Figure 3.6 illustrates this.



**Figure 3.6:** Blue boxes represents internal clocks that needs to be synchronized. Dashed lines marks wireless signals. Solid lines marks wired connections

### 3.4.1 Setup

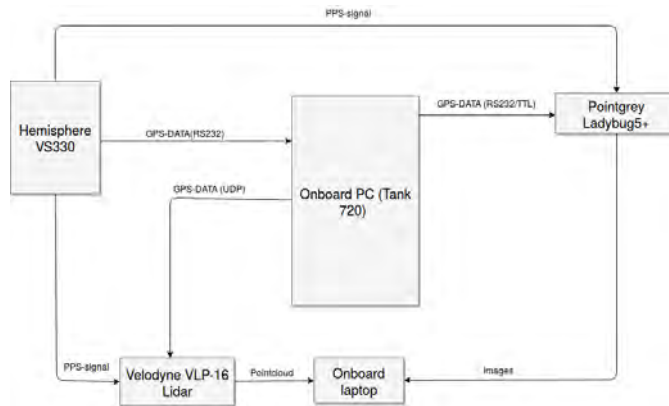
Two different time sources are used. The first is GPS-time provided by the Hemisphere VS330 GPS/GNSS-receiver. The Hemisphere is connected to the on-board computer

through a standard DB-9 female to DB-9 male cable providing NMEA-strings at rate of 20 messages per second for each type using the National Marine Electronics Association (NMEA) 0183 standard. See 3.4 for illustration and [4] for in-depth information of its format. The hemisphere also provides a pulse per second (PPS) signal, that indicates the exact moment a second has passed.

**Table 3.4:** NMEA-messages and their format[8]. Used in synchronization.

GPGGA	\$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
GPRMC	\$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A

The second time source is an NTP-pool server[28]. This is the default time server used by most linux distributions. This source corresponds to a stratum 2-4 source, depending on which NTP-pool being used. Both on-board computers connects to the server that provides the best accuracy. This requires a stable internet-connection, which is provided by 4G mobile broadband on-board.



**Figure 3.7:** Synchronization setup - Some abstractions have been made in this figure. The lidar provides a pointcloud by sending several packets, each stamped with GPS-time. Images from the ladybug5+ contains timestamps with GPS-time.

### 3.4.2 GPS-time

GPS-time is used to timestamp data in each exteroceptive sensor used in this project. The internal clock of each sensor is synchronized to a source that provides GPS-time. This is done by transmitting either a GPRMC or GPGGA-string with a corresponding PPS-signal, depending on the requirements for the specific sensor. The ladybug5+ receives both GPRMC and GPGGA from the on-board computer through a RS232-port. The ladybug only accepts signals with TTL voltage-levels(0-5V), so a converter is used to transform the RS232-signal to TTL. The velodyne VLP-16 lidar receives the data through UDP-messages broadcasted on 192.168.1.201, port 10110. The PPS-signal is connected straight

from the Hemisphere GNSS to the interface-box for the lidar and the I/O connector for the ladybug5+ as illustrated in figure 3.7. The PPS-signal is a CMOS, active high, 0-5V, which satisfies the voltage requirements of both sensors.

### 3.4.3 Ros-time

The reason for using both GPS and NTP-pools as time-sources is to see the difference in accuracy and analyze the need for synchronization using GPS-time. Rostime is used to timestamp the received data. Since this is done at the time of arrival on the on-board laptop and not the respective sensors then there will be some time-delays as described in section 2.5.3. Rostime uses the system-clock as reference when providing timestamps.

## 3.5 Visual Detection

Object detection is part of the sensor fusion pipeline. Its task is to detect and create bounding boxes around boats in real time with firm real-time constraints. Firm constraints is given by the dynamics of ReVolt, where data to be fused becomes both untrustworthy and useless given enough time. The center of each bounding box provides an angle through the transformations described in 3.3. This angle can be used with clustered detections from lidar-data to provide higher probability for correct detections. Two detection algorithms have been integrated and tested.

### 3.5.1 YOLOv3

YOLOv3, a detection-algorithm written by Joseph Chet Redmon, was chosen because of its real-time performance and accuracy. For in-depth information the reader is advised to read [31]. The weights trained for boat-detections were provided by Simen Grini[42]. The network proved hard to integrate.

### 3.5.2 SSD

An implementation of the Single Shot Detector(SSD)[25] was trained on the VOC-datasets[14] and images extracted from the detection-experiment explained in section 4.2.1. Images from the experiment had to be labeled by hand, for this LabelImg[37] was used as illustrated in figure 3.8. It streamlined the labeling-process avoiding loss of valuable time. The network was trained for two whole days on the given dataset. Due to limited time there were no opportunity to tweak both the network and dataset before testing it on data accumulated during the experiment-phase.

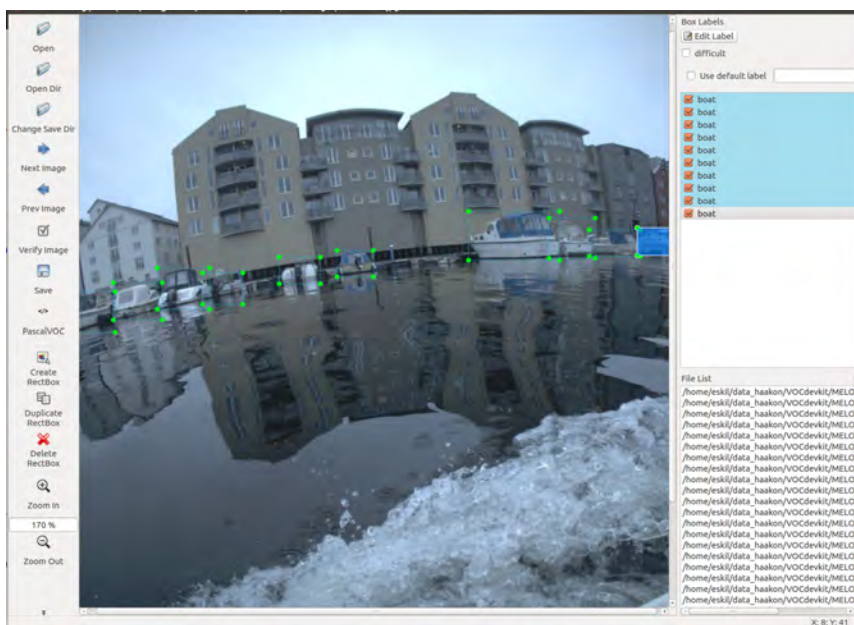


Figure 3.8: The dataset was labeled using LabelImg[37]





# Experiments and Data Collection

In this chapter the setup for the testing phase is presented, as well as the different experiments performed, their purpose and challenges.

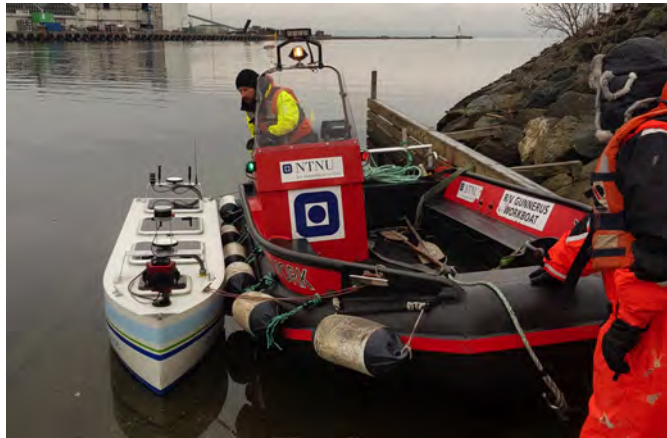
## 4.1 Setup

To analyze the suggested system and its configurations, several experiments were planned and performed to acquire sensory data. ReVolt was transferred from NTNU Gløshaugen to Trondheim Harbor two weeks in advance of the first experiment. This provided easy access to open areas with GPS-signals as well as Nidelva where the experiments were performed. The two weeks were used to finish up the technical implementations required for proper powering of sensors, synchronization and recording of sensory data. The experiments were performed late November. Gunnerus Workboat pictured at 4.1 was rented through NTNU and was used as a security-measurement and close proximity point for data recording on ReVolt. It was operated by Tor Arne Pedersen from DNV GL.

## 4.2 Performed Experiments

### 4.2.1 Day 1 - Object detection and range measurements

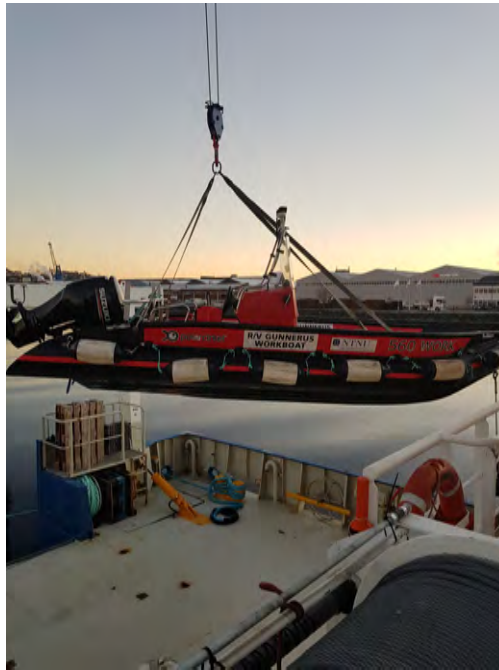
The first day of testing revolved around sensory data for object detection and distance measurements. It was done in proximity to the docks in Nidelva, close to Trondheim Sentralstasjon. Figure 4.3 shows an overview of the specific testing area. It's high concentration of boats provided a wide variety of sensory data, including different types of boats, sizes, obstructed views and distances. This experiment was done without gps-synchronized sensors as it was not implemented in time. Rostime and rosbag were used to timestamp and record sensory data respectively. ReVolt was controlled by Geir Hamre from DNV GL using a Spektrum DX6i handheld remote controller.



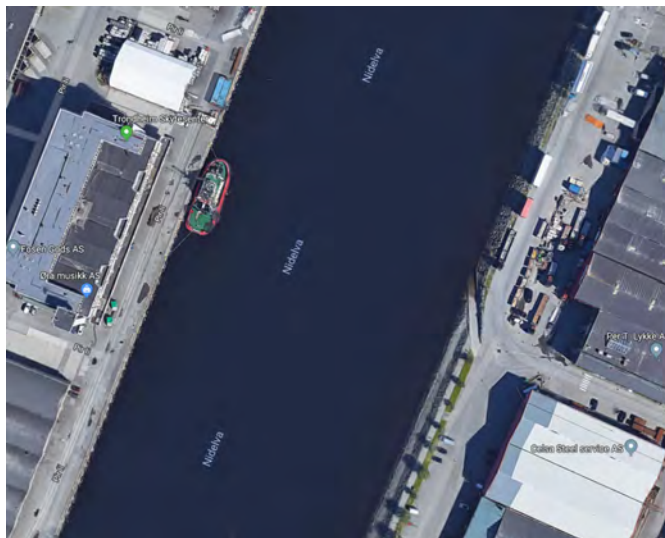
**Figure 4.1:** Tor Arne Pedersen and Geir Hamre from DNV GL contributed during the testing phase. Picture shows the launch of ReVolt on the first day of testing

## 4.2.2 Day 2 - GPS-Synchronization and point estimation

Final implementation of GPS-synchronized sensors were completed a week after the first experiment, see figure 4.7. The second experiment was performed after this and focused on visual and synchronization data from ReVolt, including timestamps using Rostime and GPS-time, IMU and camera images. The experiment had two purposes. Provide data for analyzing the need for synchronization using GPS-time versus system clock and Rostime. Secondly, provide imagery data for point estimation using the transformation found in section 3.3. Using this it can be shown how accurate true world-coordinates correlates to world-coordinates projected through the camera.



**Figure 4.2:** Gunnerus Workboat was rented from NTNU's maritime division



**Figure 4.5:** Area where synchronization-tests were done. The launch site for ReVolt can be seen to the right. Picture provide by Google Maps

The experiment was performed at the launch-site for ReVolt, illustrated in figure 4.5. The



**Figure 4.3:** Area where detection-experiments were done. Picture provide by Google Maps



**Figure 4.4:** ReVolt was controlled by Geir Hamre and driven up the entire length of the canal for recording visual sensory data

boat was connected to shore using a rope and further stabilized by one of the two people present. The boat was directed towards a known point on the other side of the canal and vigorously shaken to provide movement in all directions, shown in figure 4.6.

### 4.2.3 Interference - Noise from USB3

With the amount of sensors required for this practical project in the limited space on the ReVolt model ship, it is reasonable to believe that there will be some interference. Given the sensitivity of some sensors on-board (Hemisphere VS330, wireless), it was interesting to see what might affect it. It was not possible to get quantitative data on this experiment, since the hemisphere did not provide any software-tools for measuring signal strength. Therefore the experiment was done visually, using the signal-strength-bar on the frontpanel of the Hemisphere. The most prominent interference-source, USB3 between the laptop and ladybug, was chosen. By wrapping the cable around the antenna of the hemisphere and observing the signal bar, it was concluded that it did not interfere with the hemisphere. The wireless network was also not affected. The reasons for suspecting USB3 as an interference source can be seen in this article provided by intel [22].



**Figure 4.6:** Testing synchronization setup. ReVolt was shaken and stirred to provide kinematic-data from the IMU



(a) On land synchronization test



(b) Interference-testing using USB3 cable from Ladybug5+

**Figure 4.7:** The proposed system was tested on land before live tests in maritime environments



# Results and Discussion

## 5.1 Visual detection

### 5.1.1 Quality

YOLOv3 was tested using weights provided by Grini [42]. It did not provide sufficient results and were excluded from this report. One possible reason for insufficient results might be the dataset which the network was trained on. Grini's dataset consisted of a total of 1916 images recorded from various locations, primarily Hovik and Trondheim. These were not taken at an angle which replicates the experimental data recorded during this projects testing phase. Images captured during the testing-phase had a high degree of water-reflections as well, which the neural network was not trained for. The second reason might be human mistakes done during technical implementation of proposed network. It can be argued that the latter is the most probable reason, given the results presented in Grini's master degree which were of substantial greater quality.

Figure 5.1 illustrates the current results given by SSD. These were some of the best detections done by the network. These results can however be misleading. There is a possibility that the testing set became too similar to the training set, giving unrealistic good results when tested. This was not investigated further due to time constraints.

### 5.1.2 Distance

The lidar and camera has not been calibrated as mentioned in section 3.2.2. Their extrinsic parameters are still unknown. It is therefore not possible to accurately find the distance to a detected object using bounding boxes given by the detection-algorithms. It is however possible to get a distance measurement by examining the data by hand. Taking the bounding box which seems to be the greatest distance away and comparing it with lidar-data from the respective rosbag at the given timestamp as pictured in figure 5.3 should provide enough accuracy to be comparable with Kamsvågs results in [24].



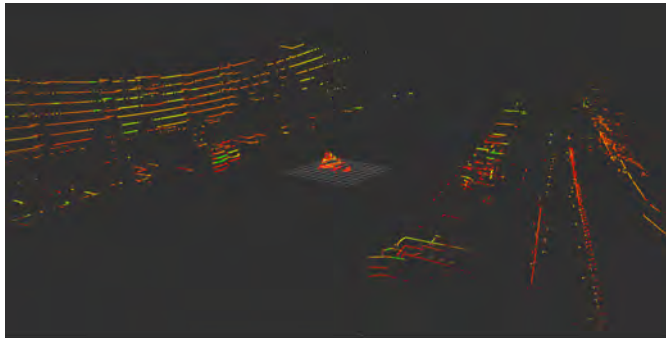


(a) Distance test



(b) Detection test

**Figure 5.1:** During distance-testing illustrated in 5.1a the lidar lost connection with the on-board laptop, resulting in no distance measurements in open sea environment.

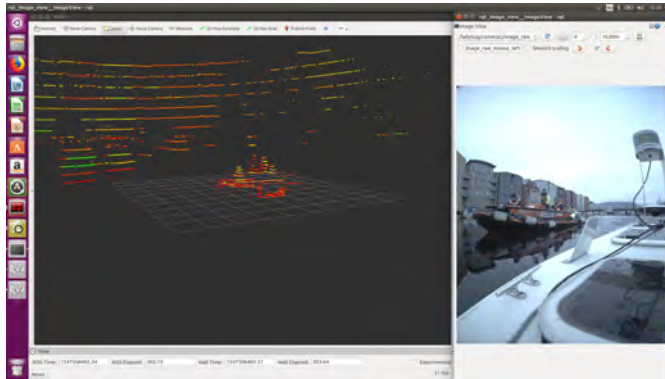


**Figure 5.2:** Lidar scan taken in Nidelva during the first day of testing. The lidar is located at the center of the coordinate-system seen in the middle. Rviz[27] was used to create this illustration.

However, the poor results presented in section 5.1.1 makes the comparison with Kamsvågs results unreliable and is therefore not included in this report.

## 5.2 Synchronization

The synchronization-test gave interesting results presented in table 5.1. By comparing time-reference-messages from relevant sensors, it can be argued that using ROS-time for timestamping provides negligible time-differences between computers on board as long as they are synced to NTP-servers through wireless. This can be seen because the Hemisphere and Lidar are connected to separate computers. Both are timestamped using GPS-time. Taking into account the transmitting-delay between sensor and computer gives a short timespan which represents the time-difference between the system clocks on both the on-



the  
**Figure 5.3**

board computers. This timespan is arguably acceptable given the dynamic of ReVolt. This does not mean that it is the best solution. System clocks on computers are not accurate and will drift if not continuously synchronized with an outside source. Using NTP-pools requires a constant wireless connection. Assuming this connection were to be lost would introduce unacceptable time-disparities between the on-board computers.

**Table 5.1:** The absolute value of timestamps made using rostime and GPS-time for each sensor is summed up and divided by the number of messages sent during one rosbag. This gives an indication of how long the transmitting delays are, as well as the performance of ROSTime. Values are given in ms.

Sensors	Result(ms)	Iterations / Number of messages
Hemisphere	35.445728	1716
Velodyne Lidar	16.597913	31230
Ladybug5+	909.882412	85

The ladybug5+ camera seems to be severely affected by transmitting/intermediate delays, reaching close to a second delay between capture of image to received image on the on-board laptop. This delay is unacceptable in a sensor fusion system with the dynamics of ReVolt. The cause of this delay is not known. It might be software-related. There might be delays introduced through the use of USB3. It might be the amount of data transferred over USB3. This has yet to be investigated.

### 5.3 Experiments to be followed up

Since undistortion of images and extrinsic calibration were uncompleted, the point-estimation test was not doable. This experiment was planned to test the transformations and extrinsic parameters found in section 3.3. The experiment was to be performed as follows, given

data recorded during day 2 of testing:

By projecting a virtual point through the camera to an existing known point, using the position and kinematics of ReVolt, it could be determined how accurate the transformations/extrinsic parameters are. The projecting would only give an angle since images do not have depth-information, but using this angle and trigonometry it is possible to determine the distance between virtual and real points. This test would also give an indication for whether synchronization of sensory data is necessary, compared to timestamping data at arrival.



**Figure 5.4:** Panoramic image captured using ladybug+5 software on Windows

## 5.4 Quality of recorded sensory data

The experiments provided sensory data from all relevant sensors. There were some mishaps, where cables fell out and sensors stopped working. However, this was not a substantial problem and only affected a small part of the experiments. This data is in good condition to be used for further research.

# Conclusion and Future Work

## 6.1 Conclusion

This project has several contributions. The first is installation of the ladybug5+ on ReVolt. This provides ReVolt with a much large FOV than previous implementations. Synchronization of every exteroceptive sensor to GPS-time is another contribution, laying a good foundation for proper synchronized sensor fusion. A considerable amount of time was used to make the latter work properly, investigating signal-disturbances, connecting signals, testing software, but proved to be time well spent. Another unforeseen contribution by synchronizing the sensors was discovering the transmitting-delay between the ladybug5+ and on-board computer. One final contribution was the experiments performed during this project. The detection-experiment done in Nidelva provides a dataset which can be used in training new neural networks.

Technical difficulties were prevalent throughout the project and caused unexpected delays and problems which contributed to tasks being left uncompleted. Undistorting of images were one of these, directly affecting the completion of the calibration process described in section 3.2.

## 6.2 Future work

- Point-estimation as mentioned in section 5.3.
- The ladybug-driver can be improved. Writing an entire new driver which relies more on the Ladybug SDK will provide more options and possibly better results.
- The current system for synchronization is based on comparing times and using the data that is closest to each other for fusion. By creating a trigger for the ladybug-camera and synchronizing the trigger with the lidar as done in [35] could provide lower time-disparities.

- 
- Investigate the functions mentioned in section 3.2 for undistorting images using Ladybug SDK.
  - Calibrate the extrinsic parameters between ladybug5+ and lidar using method referred to in section 3.2.2.
  - The ladybug5+ is currently not supported on embedded platforms. If the suggested system is to be tested on an embedded platform, another camera has to be used.
  - The ladybug5+ coordinate-system is located at the center of the ladybug5+. Each camera-sensor comes with its own transformation-matrix to reach this frame. Rather than calibrating one camera-sensor as proposed in this project, calibrate for the main-frame of the ladybug5+ camera. By doing this each camera can be used rather than just the front camera.

# Bibliography

- [1] Sigurd M. Albrektsen and Tor A. Johansen. SyncBoard-A high accuracy sensor timing board for UAV payloads. *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, pages 1706–1715, 2017.
- [2] Henrik Alfheim and Kjetil Mugerud. Dynamic Positioning of the ReVolt Model-Scale Ship. 2016.
- [3] Henrik Lemcke Alfheim and Kjetil Mugerud. Development of a Dynamic Positioning System for the ReVolt Model Ship. *IFAC-PapersOnLine*, 51(29):116–121, 2018.
- [4] K. Betke. The NMEA 0183 protocol. Technical Report January 2000, 2001.
- [5] Peter Corke. *Robotics, Vision, and Control*. Springer, second edition, 2011.
- [6] CPFL. CPFL Autoware.
- [7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.
- [8] Dale DePriest. NMEA data.
- [9] A Dhall, K Chelani, V Radhakrishnan, and K. M. Krishna. LiDAR-Camera Calibration using 3D-3D Point correspondences, ROS package. *ArXiv e-prints*, 2017.
- [10] Ankit Dhall, Kunal Chelani, Vishnu Radhakrishnan, and K M Krishna. LiDAR-Camera Calibration using 3D-3D Point correspondences. *CoRR*, abs/1705.0, 2017.
- [11] DNVGL. Autonomous and remotely-operated ships - DNV GL.
- [12] Olav Egeland and Jan Tommy Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, Trondheim, 2002.
- [13] Wilfried Elmenreich. An introduction to sensor fusion. *Austria: Vienna University Of Technology*, (February):1–28, 2002.

- 
- [14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.
- [15] FLIR. *LD5P Technical Reference*. Flir, 2018.
- [16] Christian Fossen. *Autosea*.
- [17] Thor I Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.
- [18] GIMP. Convolution Matrix - Kernels.
- [19] Paul Groves. *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems, Second Edition*. 2013.
- [20] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2004.
- [21] Hemisphere. *Vector VS330 GNSS Compass User's Guide*. Hemisphere, 2018.
- [22] Intel. USB 3.0\* Radio Frequency Interference Impact on 2.4 GHz Wireless Devices. Technical report, 2012.
- [23] Esa Jokioinen, Jonne Poikonen, Mika Hyvönen, Antti Kolu, Tero Jokela, Jari Tissari, Ari Paasio, Henrik Ringebom, Felix Collin, Mika Viljanen, Risto Jalonen, Risto Tuominen, Mikael Wahlström, Jouni Saarni, Sini Nordberg-Davies, and Hannu Makkonen. Remote and Autonomous Ships The next steps Remote and Autonomous Ship-The next steps. Technical report.
- [24] Vegard Kamsvåg. Fusion between camera and lidar for autonomous surface vehicles. (July), 2018.
- [25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.
- [26] Edwin Olson. A Passive Solution to the Sensor Synchronization Problem. Technical report.
- [27] Open Source Robotics Foundation. Documentation - ROS Wiki.
- [28] NTP Pool Project. NTP Pool, 2018.
- [29] Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS*. O'Reilly Media Inc, first edit edition, 2015.
- [30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. Technical report.
- [31] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. Technical report, 2018.

- 
- [32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Technical report.
- [33] ROS. Ros-drivers/Velodyne.
- [34] RPNG. RPNG Pointgrey Ladybug Driver.
- [35] Sebastian Schneider, Michael Himmelsbach, Thorsten Luettel, and Hans-Joachim Wuensche. Fusing vision and lidar-synchronization, correction and occlusion reasoning. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 388–393. IEEE, 2010.
- [36] Tangstad. Visual Detection of Maritime Vessels. Technical report.
- [37] Tzutalin. LabelImg.
- [38] Velodyne. VLP-16 Puck Website.
- [39] Velodyne. VLP-16 Data sheet. 2018.
- [40] Velodyne. VLP-16 User Manual 63-9243 Rev. D. 2018.
- [41] Bjørnar Vik. Integrated satellite and inertial navigation systems. *Department of Engineering Cybernetics, NTNU*, 2009.
- [42] Simen Viken Grini. Object Detection in Maritime Environments. 2019.
- [43] Xsens. MTi-G-710 - Products - Xsens 3D motion tracking, 2018.



---