Master's thesis

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

Marcus Steffensen Vormdal

USV trash detection and tracking using multimodal data fusion

Master's thesis in Cybernetics and Robotics
Supervisor: Håkon Hagen Helgesen
Co-supervisor: Gulleik Lundtorp Olsen
June 2023

NTNU
Norwegian University of
Science and Technology

Marcus Steffensen Vormdal

USV trash detection and tracking using multimodal data fusion

Master's thesis in Cybernetics and Robotics
Supervisor: Håkon Hagen Helgesen
Co-supervisor: Gulleik Lundtorp Olsen
June 2023

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

Human-made pollution is a large and growing problem on a global scale. This manifests all around the world, with one specific case being the accumulation of plastics and other non-biodegradable trash in oceans, rivers and lakes. The consequences of this ranges from death of marine life to the irreversible introduction of micro-plastics in our environment. As the sources for the trash are many, the solutions needed to counteract the problem also have to be varied.

This thesis aims to contribute to such a solution, where the final product is intended to be an Unmanned Surface Vehicle (USV) capable of collecting trash in sheltered waters such as harbours. The focus of the thesis is how an USV can detect and track floating trash while operating autonomously. The thesis first provides a review of current USV solutions that have been proposed or developed, as well as relevant technology used in maritime settings that could be leveraged to better performance. A complete system for the detection and tracking of trash is developed, using a sensor-suite consisting of LiDAR and a monocular camera mounted on an USV. The system utilizes a pre-trained Yolov7 model for object detection, an established georeferencing method and a tailored LiDAR filtering process. Measurements from the sensors (modalities) are fused together using a Joint Probabilistic Data Association filter (JPDA) to produce tracks in a geodetic reference frame. Efforts were done to synchronize and unify data from different sources. However, due to weaknesses in the capture setup some manual synchronization had to be performed.

The performance of the overarching system and the individual sensors were validated using three datasets that were acquired as part of the thesis. It was shown that the methods implemented for both the camera and the LiDAR had merit as detectors, however each with its own weaknesses. The georeferencing method was shown to be sensitive to offsets in pose and orientation, and performed worse further away from the sensor. The LiDAR used, VLP-16, lost sight of objects due to low vertical resolution, also more prevalent further from the sensor. The JPDA filter showed great promise both in fusing measurements into consistent tracks and keeping the established tracks even in less than ideal-scenarios. A test in an uncontrolled environment at Brattørkaia showed that the system was able to track objects on the water surface while the USV was undergoing movement.

Sammendrag

Menneskeskapt forurensning er et stort og økende problem på global skala. Dette manifesterer seg over hele verden, hvor en spesifikk situasjon er akkumuleringen av plast og annet ikke-nedbrytbart avfall i hav, elver og innsjøer. Konsekvensene av dette strekker seg fra død av maritimt liv til den irreversible introduksjonen av mikroplast i miljøet vårt. Ettersom kildene til søppelet er mange, må også løsningene for å motvirke problemet være varierte.

Denne avhandlingen har som mål å bidra til en slik løsning, der det ferdige produktet skal være et Ubemannet Overflatefartøy (UOF) i stand til å samle opp søppel i skjermede farvann som havner. Fokuset i avhandlingen er hvordan et UOF kan oppdage og spore flytende søppel under autonom drift. Avhandlingen går først gjennom nåværende UOF-løsninger som er foreslått eller utviklet, samt relevant teknologi som brukes i maritime omgivelser og kan utnyttes for å forbedre ytelsen. Et komplett system for deteksjon og sporing av flytende søppel blir utviklet, basert på en sensorplatform bestående av LiDAR og et monokulært kamera montert på en UOF. Systemet benytter en forhåndstrent Yolov7-modell for objektdeteksjon, en etablert metode for georeferering og en spesialisert LiDAR-filteringsprosess. Målinger fra sensorene (modalitetene) fusjoneres ved hjelp av et Joint Probabilistic Data Association-filter (JPDA) for å produsere spor i en geodetisk referanseramme. En innsats ble gjort for å synkronisere og forene data fra forskjellige kilder. På grunn av svakheter i data-oppsamlingen måtte det i tillegg synkroniseres manuelt.

Ytelsen til det overordnede systemet og de individuelle sensorene ble validert ved hjelp av tre datasett som ble samlet som en del av avhandlingen. Det ble vist at metodene som ble implementert for både kameraet og LiDAR-en hadde potensiale som detektorer, men hver med sine egne svakheter. Metoden for georeferering viste seg å være følsom for forskyvninger i posisjon og vinkel, og skalerte dårlig ved store avstander. LiDAR-en som ble brukt, VLP-16, mistet oversikt over objekter på grunn av lav vertikal oppløsning, noe som også var mer utbredt lengre vekk fra sensoren. JPDA-filteret ble vist å være lovende både når det gjaldt å fusjonere målinger til konsistente spor og for å opprettholde de etablerte sporene selv i mindre ideelle scenarioer. En test i et ukontrollert miljø på Brattørkaia viste at systemet var i stand til å spore objekter på vannoverflaten når UOF-en var i bevegelse.

Preface

This thesis concludes a M.Sc. degree for the Cybernetics and Robotics study program at the Norwegian University of Science of Technology. It is written in collaboration with the company Clean Sea Solutions. Some of the key methods used in the specialization project fall 2022 (Vormdal (2022)) is carried over and applied in a new framework. I would like to extend my gratitude to my supervisor Håkon Hagen Helgesen for providing crucial guidance and feedback throughout the project. I also want to thank my co-supervisor Gulleik Lundtorp Olsen at Clean Sea Solutions for his valuable insights and help in acquiring the datasets. This project is part of a larger undertaking to develop the Aquadrone v2, intended to be a fully autonomous USV for trash collection. It is developed around the USV platform Otter supplied by Martime robotics. The final experimental test was intended to include ground-truth measurements for validation. However, due to large synchronization issues originating from the USV platform, weaknesses in the testing setup and limited availability of the equipment, no viable ground-truth data could be acquired.

Table of Contents

1

Introduction

Industrialization and globalization have undeniably improved the standard of living for most countries on Earth. However, with this improvement new environmental challenges have appeared. Many of these challenges will require the combined effort of the global community to develop solutions and counteract changes before irreversible damage is done to the planet. One such challenge is trash following waterways out into the ocean. Here it can harm marine life and decomposes over time into smaller particulates, *microplastics*. In an effort to counteract this process several solutions have been proposed, ranging from collection at sea, passive river waste collectors to more autonomous solutions. The main intent of this thesis is to expand on the latter by researching, developing and integrating a system for autonomous detection and tracking of floating trash using a tailored set of sensors. The thesis is written in collaboration with Clean Sea Solutions AS, which aims to develop a fully autonomous Unmanned Surface Vehicle (USV) based on the Otter platform from Maritime Robotics AS.

1.1 Background

Plastic waste accumulation is an increasing problem on a global scale. Due to the non-biodegradable nature of plastics the trash that ends up in nature will remain there for thousands of years or until humans makes an effort to clean it up. In 2019 alone it was estimated that 460 million tons of plastic was produced. A significant amount of this would end up polluting the environment (OECD (2022)). Due to the fact that many of the largest cities in the world lie along waterways or rivers much of this trash will also find its way to the open ocean (Jambeck et al. (2015)). Here it will follow the currents across the planet,

accumulating in some areas such as the Great Pacific Garbage Patch, depositing along beaches or degrading into particulates by the wear and tear of the ocean. Even the most inaccessible places on Earth are not safe, with scientist finding plastic waste in places such as the Mariana trench (Gibbens (2021)) or the Antarctics (Doyle (2018)). Beside the obvious eyesore plastic waste in the ocean can have grave consequences, both now and in the future. In an extensive literature search conducted in 2015 (Gall and Thompson (2015)) it was found that 54 % of marine mammals, 56 % of seabirds and 100 % of sea turtles observed in the literature had either ingested or was entangled with plastic pollution. This was also a drastic increase from a similar study in 2007. Furthermore it impacts local communities in beach zones, many having the tourism industry as their main source of income. There are also unknown factors that might have consequences for generations to come. In recent years scientist have seen an increase in the prevalence of microplastics in human bodies. (Prata et al. (2020)) The long term effects of this accumulation is not well known, and could potentially pose a health risk. Due to these factors there is a large global incentive to both reduce the amount of waste reaching the ocean as well as collecting what is already present. Unfortunately the incentives are much smaller on a national scale, as investments into collecting infrastructure or trash handling facilities on land requires large economic commitments. Solutions that could reduce the cost, labor effort and maintainability over time are therefore a key component in making progress in these areas. One such solution is to use autonomous systems to collect trash at different stages in the cycle, with the meeting point of rivers and oceans naturally being one of the most viable areas. The field of autonomous USVs is still in its infancy, USVs in the niche field of collecting trash even more so. Several entities have taken on the challenge, such as the non-profit *The Ocean Cleanup* (Ocean_Cleanup (2023)). This thesis is written in cooperation with Clean sea Solutions AS, which aims to enable smaller entities such as companies or local government to have access to advanced water cleaning technology. In that pursuit a system for the USV-platform Otter by Maritime robotics is currently in development, with the goal being a fully autonomous vehicle that can locate, collect and deposit floating trash in harbour-areas while being aware of its surroundings.

1.2 Problem description

This thesis aims to explore the trash detection and tracking problem. This entails determining which sensor package is viable, how this sensor data should be processed and connecting the mathematical framework relating sensor measurements both local and global reference frames. A modular software system based on the findings is to be designed and implemented, and then tested on real life data captured from the USV platform. A fully autonomous USV specialized in surface cleaning will need several components to work in collaboration to achieve its goal. Beside the control elements common to all USVs such as propulsion, control design and control allocation, the system will also need a method to collect trash in the immediate vicinity. Furthermore, a sub-system that can direct the vehicle to the next piece of trash is needed. The first has been tackled in several different ways, ranging from grippers and nets to conveyor belts. The navigation component can be done by something as simple as a pre-determined path set by an operator under installation, but this solution does present several limitations. The first limitation is that the trash is not static on the surface, but will move with the currents. A rigid grid-path will therefore not be guaranteed to result in a successful catch. The second is that the system have little or no recourse if anything interferes with the planned path, which is to be expected in an active port. To work around these problems the USV needs to be able to be aware of its surroundings by using sensors, process the incoming data and make decisions based on some form of heuristic. In addition the USV should be able to determine if there is trash in the vicinity and keep a track of the trash over time.

1.3 Delimitations

The main delimitation of this thesis is the focus on the detection and tracking of floating trash. This is due to the many different areas that has to be covered to reach a functional system, many of which can be considered complex systems in their own right. There is also a delimitation regarding the implementation of object detection in a camera stream, as this is being developed in parallel at Clean Sea Solutions AS. Therefore, a suitable trash detector is assumed to be available and working. The intended final contribution of this thesis can be considered two distinct ”black boxes”, the first taking in sensor measurements, doing any necessary processing and returning detection coordinates. This is fed into the second box, the tracker, and returns relative position tracks in geodetic (global) or NED-coordinates. This methodology is illustrated in Figure 1.1.

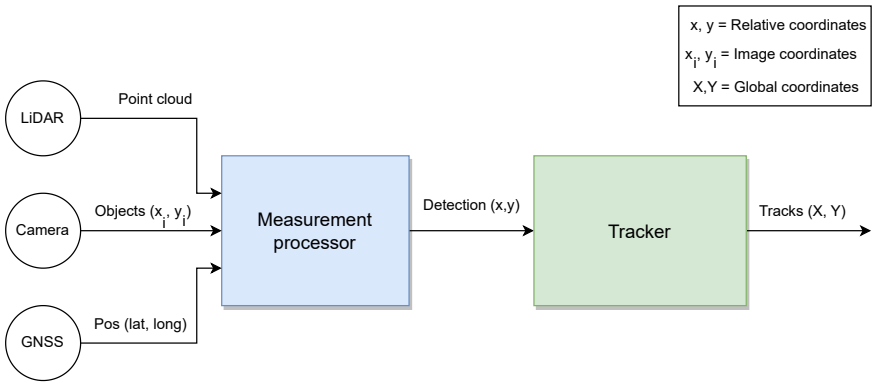


Figure 1.1: Black box overview of the intended system

1.4 Structure of the report

A brief description of each chapter is presented below:

- Chapter 2: Literature review
  Reviews some of the most relevant implementations of trash collecting USVs, as well as papers on the use of sensors and filters in a maritime environment, with focus on autonomous use-cases.
- Chapter 3: Theory
  Presents the theory needed to understand how the system is implemented, with a focus on following the path from raw sensor data to tracking output.
- Chapter 4: Methodology
  Presents the hardware used in the thesis as well as an overview of the datasets collected. Also presents the evaluation metrics that were used to measure the performance of the system.
- Chapter 5: Implementation
  Presents the main software tools used, some of the main design and implementation considerations and the code implementation of the main modules of the system.
- Chapter 6: Results
  Presents the results for sensor specific metrics and overall system performance.
- Chapter 7: Discussion
  Explores how the system performed according to evaluation metrics. Determines weaknesses and strengths of the system and identifies areas of improvement.

- Chapter 8: Conclusion and future work
  Summarizes the main results and findings of the thesis. Presents some ideas for how one could build upon the thesis to improve aspects such as performance, reliability and system validation.

2

Literature review

As the implementation of a USV for trash collection is multifaceted, a literature review of the field also has to explore several research fields. The main goal of this chapter is to identify existing solutions and what their their weaknesses and strengths are. Furthermore, technology that could be used as viable alternatives are explored, presented and discussed. The chapter first presents the scope of the review, then categorizes and presents current solutions, before finally diving into novel approaches. As the field of autonomous trash collecting USV's is niche the amount of directly related literature is limited. It is therefore viable to assess and present the main types of solutions that have been explored in various papers and which were available during the writing of this thesis. As sensors packages and their use-cases are extensive the viable technology aspect is limited to solutions that are proven to work in an USV setting, or are directly related to the viability of relevant sensors in a maritime setting.

2.1 Trash collector USVs

Several trash collector USVs have been produced and several trash collecting papers have been published. A natural separation presents itself based on the amount of external input that different solutions require as well as the system complexity. This thesis therefore separates between the three groups, **Remotely operated**, **Limited automation** and **Fully autonomous**, based on autonomy level both when looking at the solutions themselves and when analyzing other relevant technology that could be integrated into such a system. An illustration of the different categories can be seen in Figure 2.1.

Remotely operated is as the name suggests based on human operator inputs. In Akib

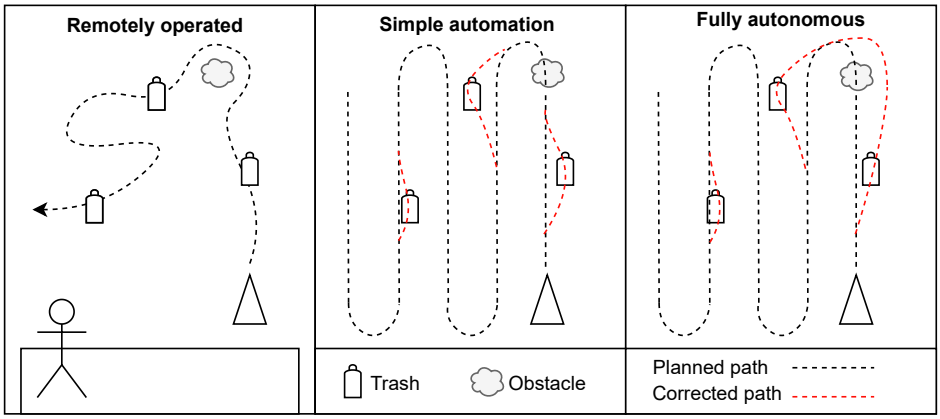


Figure 2.1: Illustration of the differences between the categories used to classify USV trash collection systems in the literature review. **Simple automation** does not include situational awareness. As such any deviation in the expected environment, such as an obstacle in the pre-planned path, can lead to system failure.

et al. (2019) a solution based on remote control over Bluetooth was prototyped, implemented and tested. Two other remotely operated prototypes were presented in Satheesh et al. (2020) and Khawaja et al. (2020). All three papers show that remotely operated USVs is a viable option to collect floating trash. However, the focus of the papers is mainly on creating an USV that can collect trash, and the drawbacks of the approach was not explored. Some obvious shortcomings can however be deduced from the papers. One of the main limitations of the remotely operated USVs is that they are not capable of making any decisions without direct human supervision, requiring a human operator at all times. Another drawback that the authors of Gao and Fu (2020) experienced was that there were limitations on range due to communication requirements. This limitation could also prove significant, especially if the range of the USV is to be outside of the line of sight of the operator, requiring a wireless video-stream.

Simple automation describes the USV solutions where the USV are capable of detecting and collecting floating trash in real time without outside intervention. They are separated from **Fully autonomous** due to lack of situational awareness and autonomous decision making beside actual trash capture. Li et al. (2020) presented a solution based on detecting trash using a camera and a modified YOLOv3 object detection method. The paper presents two experiments, the first showing that using YOLOv3 was a viable option for detecting and identifying floating trash. The second experiment was a field test where the goal was to detect, approach and grab trash. The approach was controlled by a simple algorithm, with the USV moving forward while adjusting the heading to keep the trash

aligned with the centre of the image. When the approach was completed the USV picked up the trash with a robot manipulator. The experiment was performed successfully, showing that trash could be collected according to this methodology. Although the experiment was a success several challenges present themselves. The first is that the USV had no further autonomous capabilities than detecting trash within field of view of the camera. If no trash was present it would ”cruise on the water surface with a random path.” This approach is therefore highly sensitive to unknown obstacles and reliant on trash within sight of the camera to operate as intended. Another challenge is dynamic changes in the environment such as weather, lighting conditions, boat traffic or other external disturbances. This can interfere with intended operation by obstructing the camera or moving the trash to be collected, neither of which this USV has any redundancy for.

Fully autonomous are the systems that integrates a full solution for litter collection without humans in the loop. This includes, but is not limited to, trash detection, trash collection, situational awareness, autonomous decision making and autonomous control. One such example was presented in Chang et al. (2021), where the trash detection was performed using a vision based system and collection was done using a simple heading controller. In addition an object avoidance system was integrated based on ultrasonic sensors and a water quality sensor suite was installed. Field tests were performed to validate the system. The obstacle avoidance was shown to perform as expected. The experiment to test the viability of the water surface cleaning method gave a collection rate of 70% or above depending on direction and distance from the image sensor, again pointing to vision based collection system having merit.

Another example was published in Zhang et al. (2021a), where the USV integrated path following and trash detection working in tandem to clean an area autonomously. The trash was collected using guiding rods that fed into a collection bag. LiDAR was used for obstacle avoidance, resulting in a solution that could account for its surroundings and cover a larger area. The USV was tested in a pool environment, with real river and reservoir environments left as future work.

The path planning method was further explored in Zhu et al. (2022), where measurements from camera and millimeter-wave radar were fused based on the procedure described in Cheng et al. (2021) and used for both object and obstacle detection. The path planning was separated into global and local path planning. The former let operators set the initial cleaning boundary points for the area, which was then processed by a proposed algorithm named water surface coverage path planning (WSCPP). This returned an optimized path for trash collection, accounting both for shortest path and amount of turns. The local path planning allowed the USV to diverge from the global path in the precence of

trash. A field experiment was conducted where the goal was to clean an area of about 4000 m^2. The experiment was performed both manually and with the USV. The efficiency was significantly better for the USV, reducing the task time from 2 hours down to 20 minutes. The explanation for this efficiency increase was not explicitly stated. One reason could be that the average speed of the USV was 1.2 m/s throughout the experiment, whereas a human collector might have had to stop to collect each piece of trash.

2.2 LiDAR detection and tracking

LiDAR technology can be applied in many different fields and environments. Some of the typical applications are situational awareness, SLAM and autonomous driving. Processing of LiDAR data is computationally intensive due to the massive amount of points created during one rotation, resulting in a *point cloud*. A point cloud from a dataset collected in this thesis is shown in Figure 2.2. This section focuses on the application of detecting and tracking objects using LiDAR, with a special focus on small objects.

One such example is Hammer et al. (2018), where the goal was to detect and track small UAVs using LiDAR. One main finding was that the vertical resolution of the LiDAR in use had large impact on the detection rate. It is immediately obvious that of the two sensors used, VLP-16 and HDL-64E, the former with lower vertical resolution had much worse performance, mostly detecting UAVs only at a distance of 15 m. The paper also implemented a Kalman filter for tracking, and concludes that detection and tracking of smaller objects could be a viable solution, although with limitations in range.

The HDL-64E sensor was also tested in Halterman and Bruch (2010), where the viability of the sensor for obstacle detection on an USV was explored. One important distinction between this and the tracking of UAVs is that all the objects of relevance is on a water surface with limited variation in height. In addition the objects being tracked are usually slower than UAVs. It was shown that LiDAR was able to detect objects such as buoys, semi-submerged rocks and floating kelp. It also showed that the water surface did not interact with the LiDAR in a way that returned detections, except when the water surface was disturbed by wakes from passing boats.

A more recent article, Jeong and Li (2021), used the VLP-16 for LiDAR-based in-water obstacle detection. The article focuses on the segmentation of obstacles in environments that are not known *a priori* using only LiDAR measurements. The measurements were first projected to a 2D spherical projection image, before being fed to a Breadth-First Search (BFS) algorithm. The BFS returns connected components which were then clustered, yielding obstacles. At the time the method was found to outperform in the mean

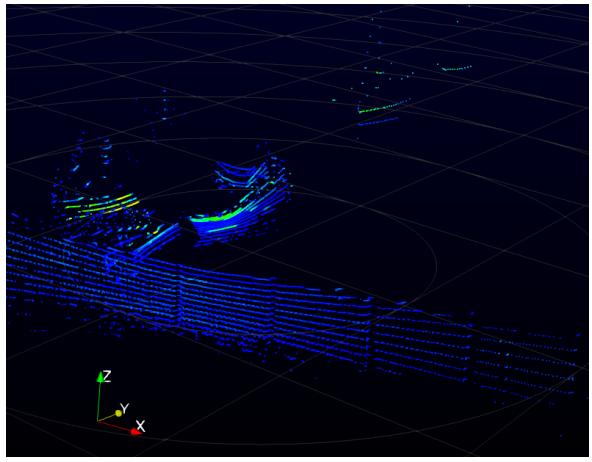


Figure 2.2: LiDAR point cloud from a single timestep of the dataset subsection 4.2.1. It captures the outline of several moored boats as well as the edge of a pier. Segmenting and tracking objects using based on these measurements is not a trivial task.

intersection over-union metric (mIoU) compared to other state-of-the-art methods within aquatic obstacle segmentation. The run time was also acceptable for real-time usage, with an average segmentation time of 72-297 ms depending on the setup.

2.3 Monocular distance estimation

To use a monocular camera to determine the relative position of an object it is necessary to convert from a 2D pixel coordinate to a 3D world frame.The general description for this task is *georeferencing*, and it is an ill-posed problem. To determine a 3D position through georeferencing it is a requirement that one or more of the degrees of freedom are removed. In the case of maritime applications this is possible if the assumption is made that the ocean surface is a plane, and that the projection of objects seen in the camera intersects with this plane. An illustration of the process is shown in Figure 2.3. One such application can be found in Helgesen et al. (2019), where a thermal camera was mounted on an fixed-wing UAV and used to find the earth-fixed coordinates of objects on the ocean surface, both static and moving. The results showed that the method was viable even at heights up to 350 m and several sources of error, with a mean accuracy in centimeters for the static objects and 2 meters for a moving object.

Øystein Kaarstad Helgesen et al. (2020) is another example of georeferencing in a maritime setting. The authors set out to extract range information of maritime vessels from both infrared (IR) and electro-optical (EO) cameras. The cameras were mounted

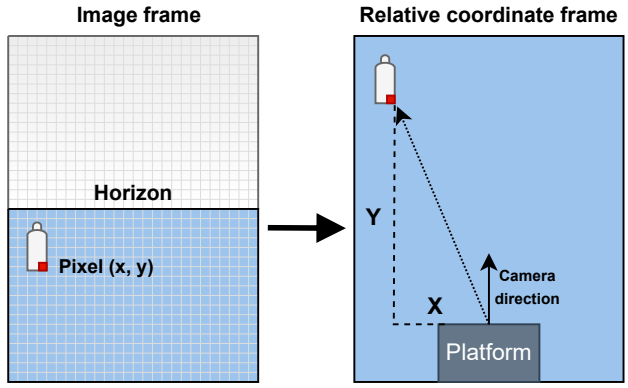


Figure 2.3: Illustration of the desired conversion from pixel to world coordinates in the case of forward looking USVs.

on a static base on land and aligned with the horizon. The calculated positions were fed into a Joint Integrated Probabilistic Data Association filter which then produced tracks of each vessel. Some of the main findings were the accuracy of the position estimated in optimal conditions was as good as a radar equivalent, but that performance degraded at long distances due to inaccuracies in the pixel coordinates of the bounding boxes. Another challenge that presented itself was false positives due to the object detection algorithm.

Another approach to georeferencing was performed in Gladstone et al. (2016). In this instance the object detection was performed using a method called Maximally Stable Extremal Regions (MSER), which finds regions in an image that remains stable over certain thresholds. A horizontal line detector was used on each image to extract a line segment. After selecting the region most likely to correspond to an object, the pixel furthest from the horizon was related to the extracted horizon line. Under the assumption that each pixel in an image represented the same angular offset in the vertical field of view, the distance could then be calculated using simple trigonometry. The implemented system was tested on moving surface vehicles and had an average error of 7.1% with a standard deviation of 5.8%. A noteworthy part of the results was that the position estimates experienced oscillations, which in parts could be explained to fluctuations in camera height due to waves. The authors suggest that this error could be reduced by measuring the height using an inertial navigation system.

An article directly related to distance estimation from an USV can be found in Woo and Kim (2015). The article set out to investigate a collision avoidance system based on information from a mounted camera. To find the position of vehicles on the surface a similar approach to the one in Gladstone et al. (2016) was developed and implemented. The

relative velocity was also estimated using the optical flow equation. Both measurements were then integrated using a Kalman filter. The corresponding tracks were used to determine if the USV was on a collision course using a fuzzy estimator. The system was tested in simulation and was proven to be able to detect and avoid vehicles on a collision course. The authors remark that the position estimate converges to zero as the relative distance decreases. Furthermore, the measurements were sensitive to target detection noise as well as angular and attitude noise. It is suggested that a physical implementation should take this into consideration by integrating a inertial navigation system (INS).

2.4 Sensor fusion in maritime environments

When a system includes several types of sensors a central issue is how these should be related to each other and what role they should perform. Sensor fusion is an umbrella term for a wide array of approaches to unify measurements from different sources. This is done to improve the accuracy or performance of the system through complementary information, as sensors have varying strengths and weaknesses. A situation where sensor fusion could improve the situational awareness is illustrated in Figure 2.4. In a maritime setting weaknesses could be present due to environmental disturbances, such as fog, rain or rough sea. A simple example of sensor fusion in such a setting can be found in Zhang et al. (2021b). The USV in the paper was equipped with a LiDAR and a camera which was used to extract relative position information from objects in the water. The measurements were fused by first projecting the a clustered point-cloud from the LiDAR corresponding to an object into the image plane. A bounding box was obtained and merged with a bounding box extracted using YOLOv3. A field test proved that the fusion algorithm had good detection effect and obstacle avoidance function. Another example of a simpler fusion method was seen in Stanislas and Dunbabin (2019). The sensor package consisted of radar, LiDAR and a camera, and was used for obstacle detection. The fusion was done by using a probability product rule to create a single sensor package at each timestep. This was then merged into a obstacle map using Bayes rule. This integration was tested on several datasets and was found to reduce classification performance and in some cases provide a more robust prediction.

One of the main use-cases of sensor fusion is within the field of target tracking. This usually requires more complex solutions than simple boundary box merging, as it is necessary to relate measurements in the time-domain. Hermann et al. (2015) is one such example, where a object detection system for a high-speed USV was developed. The sensors involved were GNSS, IMU, camera and radar, where the two first were necessary to

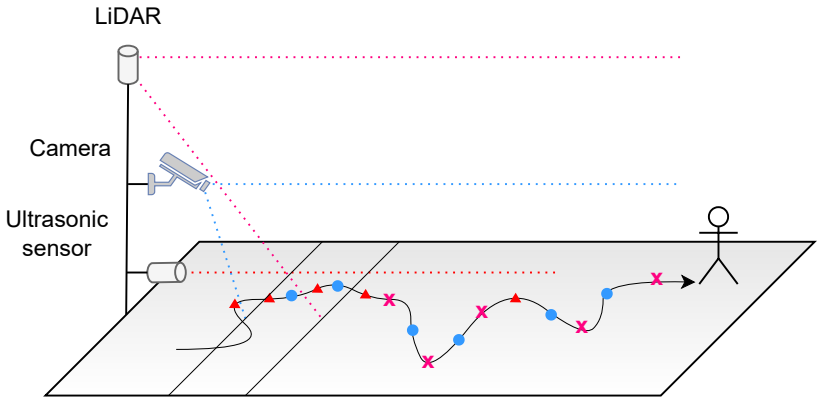


Figure 2.4: Illustration of how sensor fusion can be used to counteract weaknesses in specific sensors. The ultrasonic sensor can only measure when the person is directly in front, while both the LiDAR and camera have dead-zones. Utilizing measurements from all three sensors gives a much better track of where the person has moved.

estimate the pose and global position of the USV. The tracking of objects were performed by an Extended Kalman Filter (EKF). The system was tested on buoys and proved that it was possible to track the objects, even in cases where the radar temporarily lost track due to disturbances in pose. A Kalman filter was also used in Clunie et al. (2021), where LiDAR, radar and camera information was fused. A distinction from the former paper was that the defections were associated with a global-nearest neighbor approach and matched using a bipartite graph algorithm. These processed measurements were then fed to the filter. The field test performed showed that the sensor fusion worked adequately when the input was correct, but struggled in cases of false positives. This was a problem especially because of the camera object detection, which returned more false positives than true positives.

Another approach to multi-target tracking in a maritime setting was implemented in Haghbayan et al. (2018), where a Probabilistic Data Association Filter (PDA) was used to filter the incoming measurement originating from radar, LiDAR and cameras (IR, RGB). Each measurement was mapped to 2D radar coordinates before being filtered, and the corresponding result was a list of fused defections. The system was tested on a real dataset captured on a ferry. The paper presents a direct comparison between individual sensor performance and the filter approach. Here it is shown that the fusion led to a significant increase in true object detection as well as a strong reduction in false positives.

Han et al. (2020) extends further on the sensor fusion by introducing an intermediary step. Each sensor was assigned its own local tracking filter (EKF), which was then fed into a central tracking filter. The latter also received AIS measurements to relate the tracks to

a global coordinate system. The fusion worked as intended, with the LiDAR and cameras being able to provide enough measurements to keep tracks when objects entered the radar's blind-zone. It was found that the active sensors (radar, LiDAR) provided more reliable detection performance than the passive. As for many of the previous papers, rough sea conditions degraded performance.

2.5 Literature overview

As evidenced by the review several potential solutions and supporting technologies to detect and collect floating waste have been explored in recent years. LiDAR, cameras and radar are all sensors that have shown promise for object detection and situational awareness at different scale in maritime environments. In the specific case of USV trash collection the amount of complete solutions shown to work outside of very controlled environments are limited. The most complete solution reviewed being the SMURF in (Zhu et al. (2022)), due to both its autonomous functions as well as supporting framework easing introduction into new environments. Fusing measurements from different sensors have also been shown to have the potential to improve overall system performance for USVs. In relation to the scope of this thesis, where a LiDAR and camera is available for trash detection and tracking, there are two avenues that are not directly explored in the literature reviewed. The first is the use of LiDAR to detect trash specifically. In all but one example where LIDAR is present it is used to detect obstacles that are to be avoided, but the extension to detecting the trash is offloaded to other sensors. The second is the use of more advanced tracking algorithms to relate measurements stemming from trash in time. This could open up possibilities such as path optimization based on the predicted velocity of the trash or creation of surface gradient fields for more efficient path planning. Both of these avenues will be explored in this thesis.

3

Theory

This chapter presents relevant theory needed to understand the implemented system and the results in this thesis. The first section lays out the foundation for monocular distance estimation. It first presents camera fundamentals and the pinhole model, then presenting the simplifications that can be done due to the structure of the problem and finally arriving at a method of determining distance of objects in a single monocular image. The second section presents the theory behind extracting relative position information of floating trash on the water surface using LiDAR. This is done by filtering the raw point cloud data based on criteria such as distance, height and intensity. Further filtering is done by recognising land edges using a edge detector called the Line Segment Detector. The last section presents the Joint Probabilistic Data Association filter (JPDA), which is capable of tracking the detected trash under different circumstances using a mix of sensor measurements. Throughout this section bold upper-case notation ($\mathbf{A}$) denotes matrices, bold lower-case ($\mathbf{b}$) denotes vectors and regular letters denotes scalars.

3.1 Monocular distance estimation

To extract information from an image a mathematical model of how images are formed is needed. Although modern cameras are inherently complex a simple and idealized abstraction called the pinhole perspective model is often used as a viable approximation (Forsyth and Ponce (2012)). The model relates the image formation through perspective geometry. This section presents the fundamental mathematical model, how this model can be simplified due to known variables in the problem and how relative distance between camera and objects can be determined with these simplifications. Figure 3.1 shows the relevant coor-

dinate frames and their relation. The transformations between the frames will be explained in closer detail in this section.

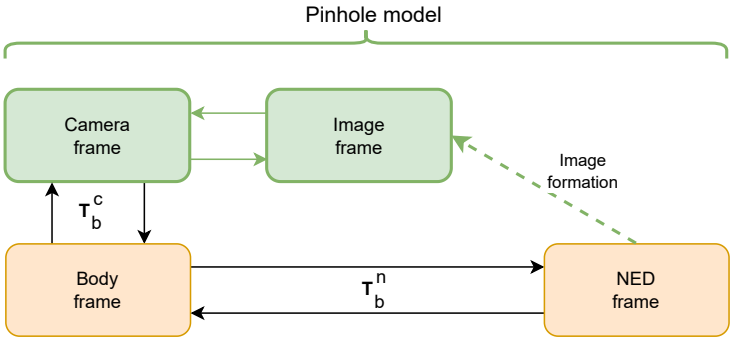


Figure 3.1: Diagram relating the different frames

3.1.1 Camera fundamentals

The pinhole perspective model, having roots in projective geometry, simplifies the concept of image formation. The geometrical relations of the model can be seen in Figure 3.2. The basis for modern image formation is that light reflected of objects within the cameras field-of-view is let through a small opening, illustrated as *pinhole* in the figure, and interacts with an exposed medium within the camera on the *Image plane*. For older cameras this medium was light sensitive film, but modern cameras uses an array of semiconductors that responds to exposure of photons. A common simplification is to use the *virtual projection plane/vertical image*, which is a flipped and equidistant plane in front of the pinhole. This removes the need to rotate the image. The distance from the pinhole to the image sensor (and virtual projection plane) is know as the *focal length*, marked by f in the figure. Using similar triangles one can represent the relationship between the distance Z from the object and the displacement in the x-axis X as

$$\frac{x}{f} = \frac{X}{Z} \tag{3.1}$$

$$\implies x = f * \frac{X}{Z} \tag{3.2}$$

The same logic can also be applied for the y-axis, leaving the two formulas:

$$x = f * \frac{X}{Z} \tag{3.3}$$

$$y = f * \frac{Y}{Z} \tag{3.4}$$

Where x and y represent the pixel coordinates in the image.

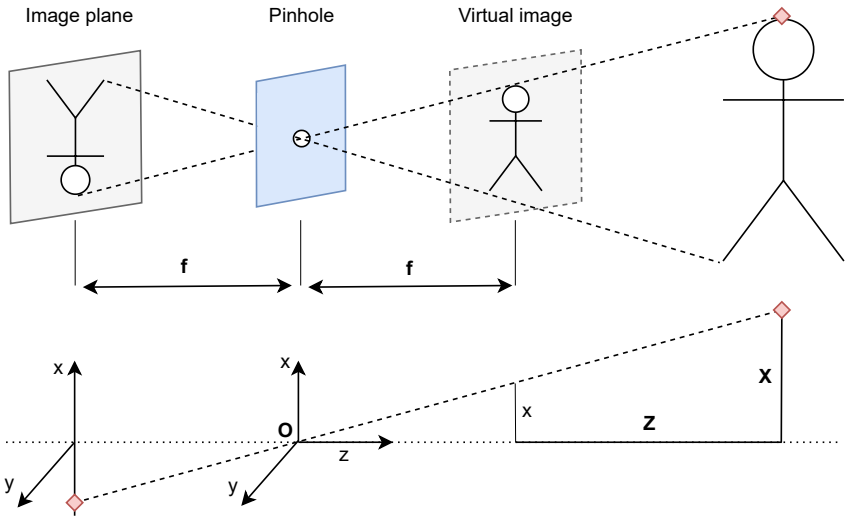


Figure 3.2: The geometrical relations of the pinhole model. Inspired by TTK4255, Lecture 3 - "Image Formation", Anette Stahl

3.1.2 World coordinates

Having established the pinhole model the next step is to relate the image to a world coordinate frame. To do this three coordinate systems are needed, shown in Figure 3.3. The first is the world coordinate frame, which functions as the "base" reference and can be arbitrarily defined. The second is the camera coordinate system, which is related to the world coordinate system through the a homogeneous transformation matrix of the lie group SE(3) (Briot and Khalil (2015)), consisting of a translation and a rotation in 3-D. Starting with the conversion from image to camera frame the first step is to use (3.3) from the pinhole-model, which when converted to homogeneous coordinates becomes (3.5). X, Y and Z represents the camera coordinates, f is the focal length and x and y is the image coordinates.

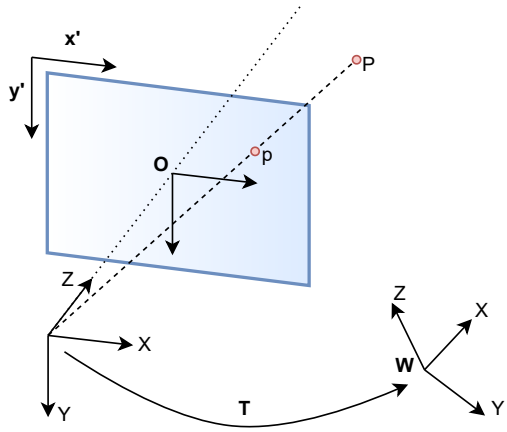


Figure 3.3: The coordinate system relations. Inspired by TTK4255, Lecture 3 - "Image Formation", Anette Stahl

$$Z_c * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3.5}$$

Due to the fact that the optical centre aligns with the centre of the image plane a conversion has to be done to find the corresponding pixel coordinates, which most commonly has the origin (0,0) in the left top corner of an image. The conversion is done by translating and scaling the coordinate. The latter is done to specify the relation between pixel array and the image. The translation is given by o_x and o_y while the scaling is given by s_x, s_y and s_θ. s_θ is only non-zero if the pixels are not rectangular. The conversion can be seen in (3.6), where x' and y' are the pixel coordinates.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.6}$$

Combining (3.6) and (3.5) yields (3.7).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} \tag{3.7}$$

To relate the camera coordinate vector to a world frame the a homogeneous transformation matrix of the lie group SE(3) can be used. The transformation can be represented on matrix form by (3.8):

$$\textbf{SE(3)} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \tag{3.8}$$

where the translation matrix is given by $\mathbf{T} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$. To follow established practices in marine environments, the zyx-convention is used for the rotation matrix **R** (Fossen (2011)). **R** is the product of consecutive rotations around the z, y, and x-axes. The angles corresponding to each rotation are given by ψ, θ and ϕ, and entitled yaw, pitch and roll respectively. The full rotation matrix from the camera frame to world frame R_c^w is then given by (3.9).

$$\mathbf{R}_c^w = \underbrace{\begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}_z(\psi)} \underbrace{\begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix}}_{\mathbf{R}_y(\theta)} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix}}_{\mathbf{R}_x(\phi)} \tag{3.9}$$

Mapping a homogeneous coordinate vector defined as $\begin{bmatrix} X_c & Y_c & Z_c & 1 \end{bmatrix}^T$ from the camera coordinate frame to the world frame can then be done by matrix multiplying as shown in (3.10), yielding the world coordinate vector on the form $\begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T$. The last row and column are needed to unify the framework when using homogeneous coordinates.

$$\underbrace{\begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) & t_x \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) & t_y \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}_c^w} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.10}$$

Combining the results to find the complete relationship between a pixel in the image and a point in the world coordinate system gives (3.11).

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.11}$$

3.1.3 Georeferencing

This thesis focuses on an USV that is floating on the surface of a body of water that is shielded from large disturbances due to waves (harbours, lakes and rivers). With this in mind some very valuable assumptions can be made. The first is that the roll and pitch of the USV are negligible, or that they can be measured and corrected by an on-board Inertial Measurement Unit (IMU). The second is that as the height of the camera above the surface is known. Finally, the surface itself can be assumed to be approximated as a flat plane under ideal circumstances. The method presented in Øystein Kaarstad Helgesen et al. (2020) is the chosen approach for georeferencing in this thesis. The choice was made based on shared characteristics in the problem structure as well as the availability of an established baseline which could be compared against. The method is illustrated in Figure 3.4. The main distinction between the referenced paper and the application in this thesis is related to the vector $\mathbf{t}_w^c$ marked in yellow on the right. When mounted to an USV the translation along the z-axis (height above water) will be lower and the corresponding vector will be smaller in magnitude.

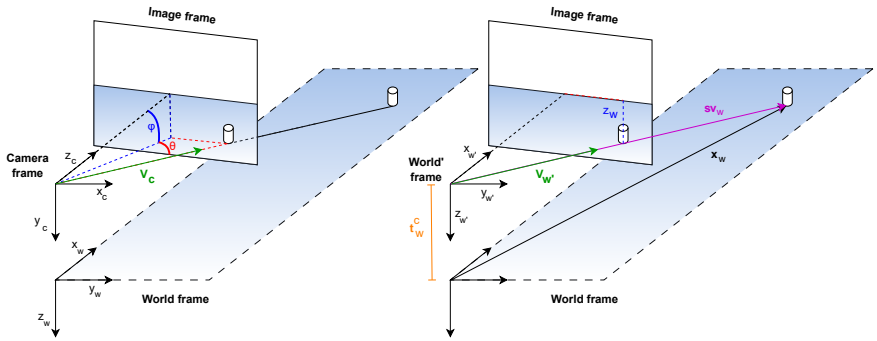


Figure 3.4: Illustration of the georeferencing method applied in this thesis.

When an image object detector, such as the YOLOv7 (You Only Look Once) model used in this thesis, detects an object in an image a bounding box covering the object is returned. The bounding box is given in pixel coordinates. The coordinate of interest depends on the orientation of the camera. A camera mounted straight down towards the

ground would give correct results using the centre of the box. In the case of a forward looking camera the pixel of interest will be on a y-coordinate along the intersection between the box and the water surface. This will naturally be where the object intersects the surface-plane and the z-coordinate therefore is equal to zero. Choosing the central box coordinate along the x-axis then yields the pixel coordinates $\begin{bmatrix} X_c & Y_c \end{bmatrix}$.

The georeferencing starts by finding the bearing θ_c and elevation φ_c of the pixel in the camera frame relative to the image centre. The formulas are shown in (3.12) and (3.13) respectively. P_x and P_y represent the image resolution of the camera given in pixels, while F_x and F_y represents the Field of view (FOV) in radians.

$$\theta_c = \frac{x_c - P_x/2}{P_x} F_x \tag{3.12}$$

$$\varphi_c = \frac{y_c - P_y/2}{P_y} F_y \tag{3.13}$$

Using these angles it is now possible to relate the detection to the camera coordinate by establishing a vector $\mathbf{v}^c$. The z component is chosen as the unit vector z = 1. The relation is given by (3.14) and seen on the left side of Figure 3.4.

$$\mathbf{v}_c = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix} = \begin{bmatrix} tan(\theta_c) & tan(\varphi_c) & 1 \end{bmatrix} \tag{3.14}$$

This vector is then converted to an intermediary world reference frame, *w'*, that shares the axis alignment with the world frame but is not translated along the z-axis. This keeps the vectors camera origin. The transformation is done using the transformation matrix defined in (3.10). The transformation is formulated as (3.15).

$$\mathbf{v}_w = \mathbf{T}_c^w \mathbf{v}_c \tag{3.15}$$

With the world frame aligned as depicted, where the camera frame z-axis is aligned with the world frame y-axis, the rotation matrix will consist of a 90° rotation around the y-axis followed by a -90° rotation around the z-axis. The translation *t* is given by the camera offset from the world frame origin in the xy-plane (world frame).

It is necessary to find a scale factor *s*, so that the vector $\mathbf{v}'_w$ has $z'_w = t^w_{c_z}$. Scaling $\mathbf{v}_c$ with *s* will yield the vector $s\mathbf{v}_w$ illustrated in purple. The scale factor can be found according to (3.16).

$$s = -\frac{t^w_{c_z}}{z^w} \tag{3.16}$$

After retrieving the scale factor the objects position can be determined according to

(3.17).

$$\mathbf{x}_w = \mathbf{t}_c^w + s\mathbf{v}_w \tag{3.17}$$

Since the z-component of $\mathbf{v}_w$ was scaled to match $t_{c_z}^w$ the resulting vector $\mathbf{x}_w$ should be on the form $\begin{bmatrix} x & y & 0 \end{bmatrix}^T$. This leaves the x and y components, which corresponds to the position of the object in the world frame.

3.2 Coordinate frames

When measurements from several sensors are to be unified in a common framework it is a necessity to establish a physical relationship between the different origins. This becomes even more important when the mounting platform is subject to changes in both pose and position, as is the case for a moving USV. This section presents the main coordinate *frames* used throughout this thesis and their relationship. Figure 3.5 illustrates the relationships between frames on block diagram form.

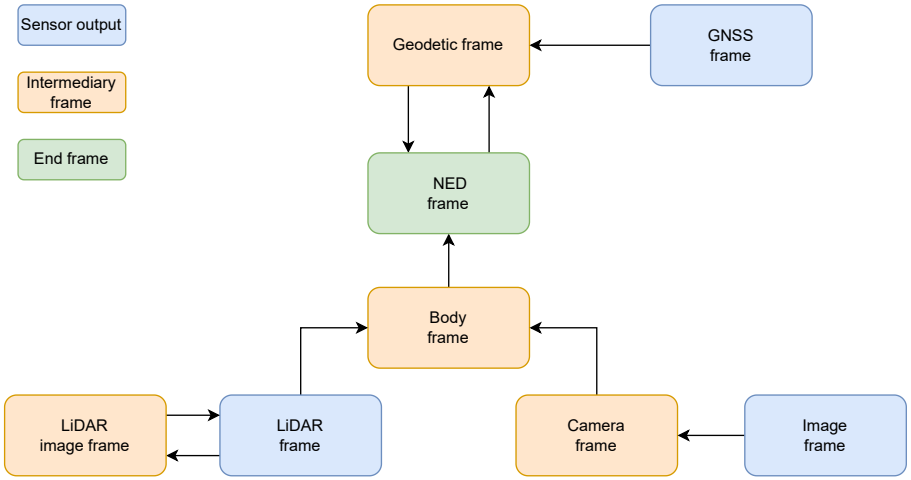


Figure 3.5: A block diagram showing the relationship between the different frames encountered in this thesis

The **geodetic frame** is a representation of earth where geographical coordinates on the form (latitude, longitude) maps to a specific position on earth. The North-east-down, **NED**, frame is the chosen world frame in this thesis. This is the frame where all processed measurements as well as tracks are mapped and are ready to be distributed outside the implemented system. The origin of the NED frame is given by a geographical coordinate

and therefore pinned to the geodetic frame, with the North and east axis at the origin aligned with the geodetics'. The NED frame is a tangent plane and therefore only accurate in a limited area around the geodetic origin. The **Relative USV frame** is given by the USV's offset in yaw, pitch and roll as well as the translation along the xy-plane, all relative to the NED frame. The **Camera frame** and **LiDAR frame** are both given by a translation in the yz-plane from the relative USV frame. This is set by the distance between the origin of the relative USV frame and the sensor. An offset in roll, pitch and yaw due to alignment errors might also have to be accounted for. The **Image frame** is related to the camera frame through the pinhole model. The **LiDAR image frame** relates 2D-LiDAR measurements to a pixel array used during the filtering process. Finally, the **GNSS frame**, where GNSS measurements are received, is related to the geodetic frame by a translation in the z-coordinate equal to the receivers height above the surface.

3.3 Small object LiDAR tracking

This section presents the the working principle of LiDAR as well as the methods used to filter the relevant information from the noise in the measurements. The filtering process is tailored to the use-case of detecting floating trash, which is distinct from the object detection and avoidance commonly implemented in an USV setting. Where object avoidance deals with identifying objects above a certain size that can endanger the craft, trash detection seeks to only be left with measurements from objects small enough to be captured. This thesis proposes a pipeline for separating out the wanted measurements. In addition to being to able to dynamically process LiDAR in novel surroundings it has the benefit of returning measurements on a format that conforms with the one-shot assumption of the JPDA, further explained in Section 3.4. The filtering is done in three steps. The first is **preliminary filtering** using simple thresholds to determine which measurements that are of interests. The second step is **filtering**, which is done to remove measurements stemming from harbour edges and other obstacles such as boats. The final step is **clustering**, which is performed to reduce noise and unify the measurements with the JPDA filter described in Section 3.4. An overview of the proposed pipeline to convert raw LiDAR data into usable LiDAR measurements is shown in Section 3.6.

3.3.1 LiDAR

Light Detection and Ranging, LiDAR, is a sensor technology based on lights ability to reflect of objects. By sending out highly concentrated laser light and measuring characteristics such as light wave shift, intensity and delay in transit it is possible to calculate

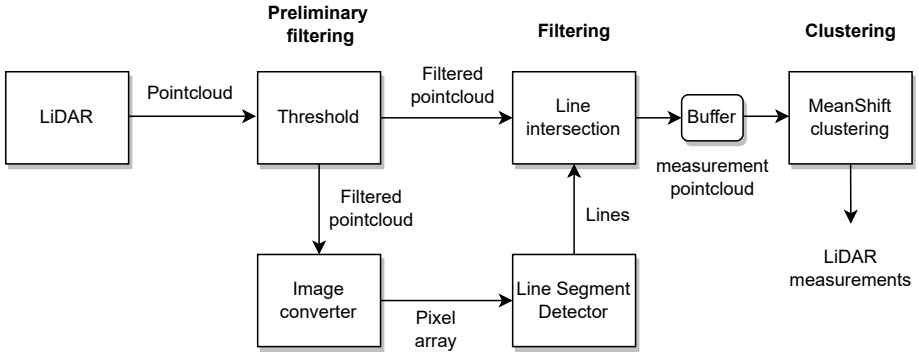


Figure 3.6: The full pipeline from raw data supplied by the LiDAR to JPDA-compliant LiDAR measurements. The LiDAR measurements stemming from the MeanShift clustering is ready to be passed to the JPDA filter to create tracks

a position and heading of the object reflecting the laser light. Most commercial LiDAR solutions consists of a mechanical laser array rotating several times a second, making it possible to get information about the surroundings with a 360° view. Some limitations do however apply. A key challenge for this thesis is the density of the lasers in the grid. The horizontal resolution is determined by the amount of samples taken on a single rotation, and is usually in the range of less then a degree. The vertical resolution is usually much less refined, as it is determined by the field of view as well as distance between and number of lasers in the grid. Figure 3.7 shows how the light from the laser array will spread over a given vertical distance. The first thing to note is that there is a *Dead zone* below the sensor in the area before the lowest laser intersects the ground. No information can be gathered here, which means that the LiDARs operating range has a minimum distance. To calculate where the laser intersects the ground one can use (3.18).

$$\text{dist}_i = h * \tan(90^\circ - \frac{\text{FOV}}{2} + i * res) \tag{3.18}$$

where *h* is the height above ground, FOV is the field of view of the sensor in the vertical plane, *res* is the vertical resolution and *i* is the number of the ray indexed from zero.

The second challenge can be seen where the two rays intersects the rectangular box in Figure 3.7. This presents the worst-case scenario where an object is just behind a lasers detection range. Using the formula (3.19)

$$\min\_h_i = \frac{\text{dist}_i - \text{dist}_{i-1}}{\tan(90^\circ - \frac{FOV}{2} + i * res)} \tag{3.19}$$

one can determine the height needed to intersect the next ray and with that be observable. These results are calculated for the rays below the horizon (8 individual rays) for a LiDAR with FOV of 30 degrees at a height of 1 m, corresponding to the LiDAR VLP-16 which is used in this thesis. The results can be seen in Table 3.1. From the table on can see that the rays closer to the horizon have larger distances between them and correspondingly require a larger height to intersect with objects.

Ray	GID (m)	h (m)
0	3.73	-
1	4.33	0.14
2	5.14	0.16
3	6.31	0.19
4	8.14	0.22
5	11.43	0.29
6	19.08	0.40
7	57.28	0.8

Table 3.1: The calculated relative ground intersection distance (GID) and worst case height for each ray of the VLP-16

One aspect to note is that objects slightly within the Dead zone might be detectable depending in the height. Reformulating (3.19) one can find the distance of which an object of i.e. 10 cm would intersect the first ray: $\text{dist}_{i-1} = 3.73m - 0.1 * \tan(75°) = 3.35m$.

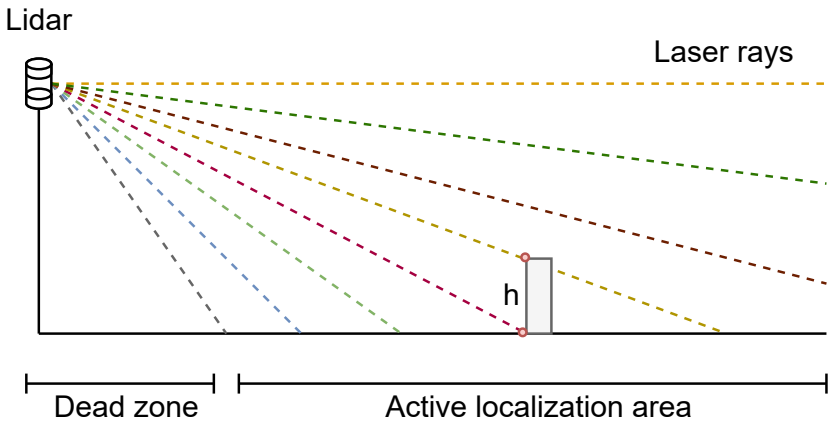


Figure 3.7: Illustration of the vertical resolution of LiDAR and the problems that arise. *h* corresponds to the height in Table 3.1, which in the illustrated case would be h = 0.29 m

3.3.2 Preliminary filtering

This thesis focuses on the detection of trash floating on the surface of relatively controlled areas of water such as harbours, rivers and lakes. This presents the opportunity to simplify and filter the raw LiDAR data to remove unwanted noise and detections. The first natural step is to filter any measurements that are not within a thin 3D-slice parallel to the water surface. This ensures that any measurement that stems from boats, buildings, harbour walls etc. above a certain height is removed. The filtering step is shown in Figure 3.8. The second filtering step follows naturally from Table 3.1, where it was shown that the further from the source the object is the less likely it is to be detected. There is therefore diminishing returns on probing too far, and filtering measurements outside a radius determined by calculated results and real-life testing can be a valid approach. The final step is based on the fact that the final product of this thesis is to determine the relative position of the floating trash. As the USV and the trash is on the same plane (water surface), the z-coordinate is redundant and might be discarded completely. This also simplifies the integration of different sensors as only the relative x and y coordinates of each measurement is needed.

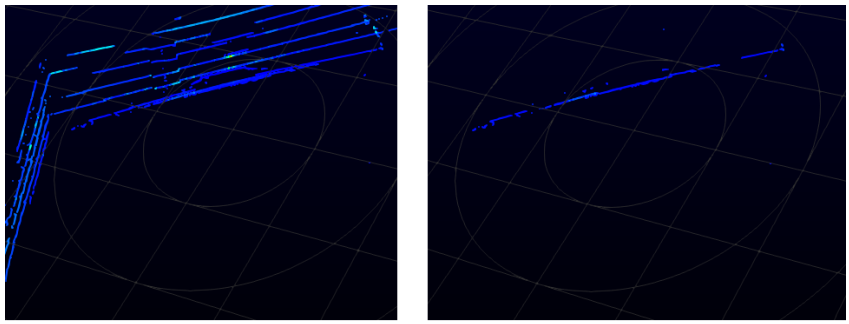


Figure 3.8: Before and after preliminary filtering. LiDAR measurements above a set height and beyond a set radius are discarded.

3.3.3 Line extraction

As the USV moves in its surroundings it will encounter a variety of obstacles, from harbour walls to moored ships. One solution to filter out some of the measurements stemming from these objects is to use a mask based on harbour maps. However, this does not account for any dynamic changes in the environment, such as ships mooring at new locations or the tide revealing partly submerged rocks. To do so a method better suited to a changing environment should be chosen. This thesis handles the problem by projecting the 3D-LiDAR measurements to a 2D-plane, also making the assumption that all objects that are larger

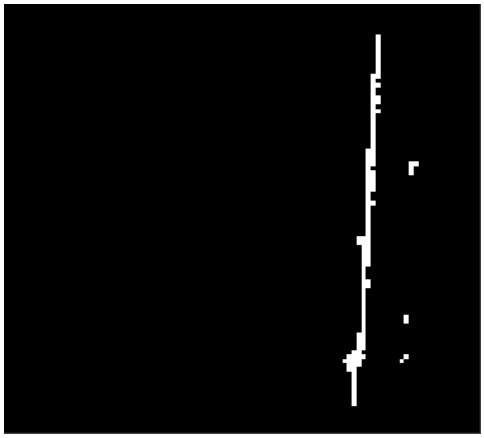


Figure 3.9: LiDAR image created by casting LiDAR measurements to a discrete array. Each white pixel corresponds to a measurement.

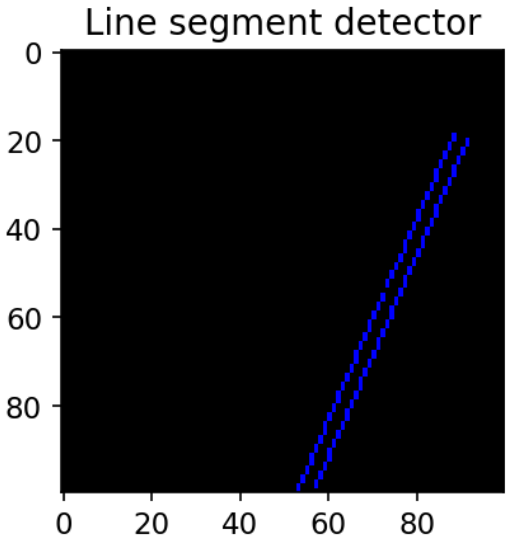


Figure 3.10: Line segment extracted from the LiDAR-image. Note the two lines corresponding to the edge from black to white and white to black respectively.

than the trash to collect can be represented by lines or line segments. Having projected the LiDAR to 2D leaves a x and y-coordinate for each point in the pointcloud at each time-step. To extract the line segments a process called the Line Segment Detector (Morel et al. (2012)) is used. This process receives an image and returns the start and end coordinate of each line segment found. For a deeper dive into the working details the reader is referred to (Vormdal (2022), chapter 3). To utilize the method the measurements have to be unified

in an image format. To do this one starts by rounding the coordinates to the nearest integer (yielded as floats with high precision from the VLP-16). This yields a discrete set of measurement values in the range $X \in [-radius, radius], Y \in [-radius, radius]$. The standard format for an image of equal size is $X \in [0, 2 * radius], Y \in [0, 2 * radius]$. To convert a discrete 2D-map of LiDAR measurements is therefore as trivial as adding the radius to each measurement and "coloring" the pixel corresponding to the new x and y values. However, due to the way the LSD grows regions it is preferable that the individual pixels constituting a line are adjacent. This can be solved by reducing the resolution of a corresponding pixel-array to a suitable size, making sure that the LiDAR measurements are calculated to fit the chosen size. A Gaussian blurring effect can also be added to counteract discontinous jumps. A converted LiDAR image is shown in Figure 3.9. A output image after running the LSD is shown in Figure 3.10. After retrieving the line segments the next step is to convert them back to the relative measurement framework. This can be done by doing the conversion process in reverse.

3.3.4 Line filtering

The final step in the filtering process is determining which points are behind the line segments. The process is shown in Figure 3.11. The first step is to create *virtual line segments* between the LiDAR position (i.e. the origin (x,y) = (0,0) in the relative frame) and the measurement coordinate. It is then possible to calculate if each virtual line segment intersects any of the line segments found with the LSD using Algorithm 1, which is implemented based on Antonio (1992). Here get_t and get_u are calculated according to (3.20) and (3.21). The subscripts are found from the start and end point of the two line segments being compared at each iteration, written as $\mathbf{L}_1 = \Big[[x_1, y_1], [x_2, y_2]\Big]^T$ and $\mathbf{L}_2 = \Big[[x_3, y_3], [x_4, y_4]\Big]^T$

For the application of determining if a point is behind a line segment it is only necessary to determine if two segments intersect or not, which means that process of finding the exact intersection point can be skipped.

$$t = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{3.20}$$

$$u = \frac{(x_1 - x_3)(y_1 - y_2) - (y_1 - y_3)(x_1 - x_2)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{3.21}$$

The intended result in the USV frame is shown in Figure 3.12.

Algorithm 1 Line segment intersect algorithm

$measurements \leftarrow get\_measurements(frame)$
$lines \leftarrow LSD(image)$
$non\_intersecing\_apoints \leftarrow []$
for line in lines **do**
  for p in measurements **do**
    $v\_line \leftarrow ((0,0),(p\_x,p\_y))$
    $t \leftarrow get\_t(line, v\_line)$
    $u \leftarrow get\_u(line, v\_line)$
    if $!(0 \leq u \leq 1)$ and $(0 \leq t \leq 1)$ **then**
      $non\_intersecing\_points.append(p)$
    end if
  end for
end for
Return $non\_intersecing\_points$

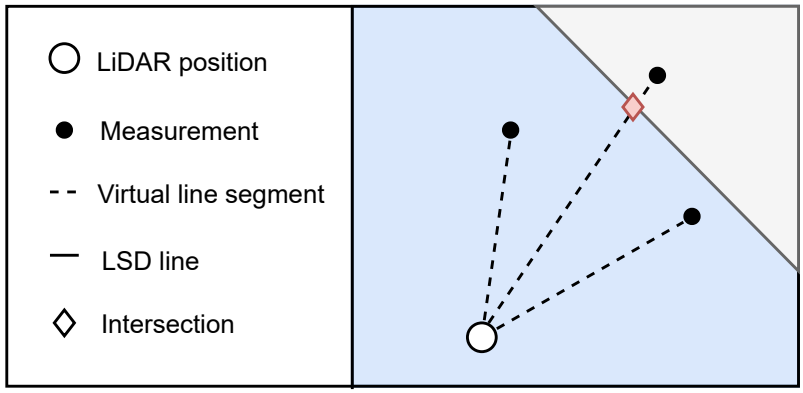


Figure 3.11: Line segment intersection

3.3.5 Buffering and Mean Shift clustering

The LiDAR measurements that are left after the line intersect filtering have some characteristics that are unwanted. The first is that several measurements can occur from a single object. The second is that the filtering process does not remove noise due to laser-to-surface interaction. To counteract this a buffering and clustering step can be used. The buffer and cluster process is visualised in Figure 3.13.

The buffering serves the purpose of reducing the impact of the low vertical resolution discussed in subsection 3.3.1. This is done by leveraging the fact that the USV will be moving during operation and that the LiDAR can rotate at a high frequency. If the USV is moving at a constant velocity of 1 m/s and the LiDAR rotates at a frequency of 10 Hz, each

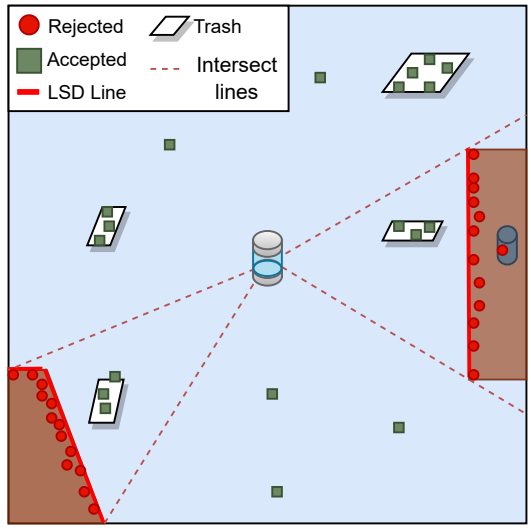


Figure 3.12: Rejected and accepted LiDAR measurements after running 1. The *intersect lines* marks the relationship between the LiDAR and the LSD line end-points.

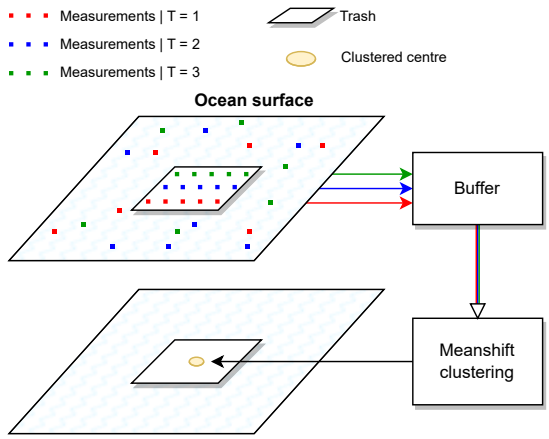


Figure 3.13: The buffering and clustering process. Three line intersect filtered pointclouds are buffered to utilize the movement of the USV. The result is a single LiDAR measurement for each observed object, while also filtering noise from the surface. T denotes the current time-step

laser ray will have its intersection point with the surface moved 10 cm in the direction of movement. This means that an object caught in-between two rays might move into view at the next time step, or that an object already in view might still be in view if large enough.

After a user-defined amount of samples is buffered the data can then be passed to a clustering algorithm. The purpose of the clustering is to find a central point for each

cluster of measurements. The clustering method used in this thesis is the Mean Shift clustering algorithm (Comaniciu and Meer (2002)). The choice is based on four advantageous attributes the method offers. The first is that the algorithm is *centroid*-based. This is of interest as each cluster center ends up being a mean of all the members in the cluster, which is especially relevant in the case of multiple scan lines being returned from a single object. The second attribute is that the amount of clusters can be determined dynamically. As the amount of objects within range varies as the USV moves in the environment this is key to map measurements to corresponding clusters. The third attribute is that very little parameter tuning is required when using the implementation found in Pedregosa et al. (2011). This increases robustness as a set of parameters tuned for a specific environment might not be viable in other circumstances. The density of trash, amount of noise from the surface or other variable factors can impact what is the ”right” parameters. Finally, the scikit-implementation enables *orphaning*, filtering out measurements not connected to a cluster. This removes ”one-off” noise from the surface. A drawback of the method is that it suffers from limited scaleability. Due to the limited amount of measurements let through the filtering stages this was deemed an acceptable compromise. The final processing step is to remove every cluster consisting of less than a user-defined threshold of points. This further removes noise and ensures that only non-transient measurements remain. The remaining measurements are then ready to be passed to the tracking filter.

3.4 Joint Probabilistic Data Association filter - JPDA

To unify the measurements from the sensors and keep track of objects over time a tracking filter is necessary. The chosen filter for this thesis is the Joint Probabilistic Data Association filter (JPDA), first presented in Fortmann et al. (1983). The theory in this section is based on Brekke (2020). The JPDA filter enables tracking of multiple objects simultaneously, attributing measurements to tracked objects according to the normalized conditional probabilities of every track and arriving at a globally consistent solution at each time-step. It was chosen as it allows for tracking of objects in close proximity as opposed to a single-target tracking filter such as the Probabilistic Density Association filter (PDAF). One of the main weaknesses, track-coalescence (merging of different tracks), is also not a large concern and might in fact be beneficial. This is due to certain types of trash tendency to entangle (i.e. seaweed, plastic nets). It is also proven to be computationally efficient enough to run in real-time on common computing hardware. This section presents the main components of the JPDA-filter, including the chosen measurement and noise models. The filter was chosen as a proof-of-concept for tracking of floating trash, and some

simplifications that will be presented were deemed acceptable for this purpose. Note that the filter is processed in the NED frame, which means that the GNSS position already have been integrated in the measurements.

3.4.1 Structure

The main structure of the JPDA filter is based on thirteen underlying assumptions. The first two are a cornerstone for most multi-target tracking methods. Number one is that each target generates at most one measurement. Number two is that any measurement comes from at most one target. Although being large simplifications this has a positive impact on computational complexity and stops the filter from attributing all measurements to a single cluster. The next six attributes are general for the prevailing paradigm within the field of multi-target tracking. The final five are JPDA specific. Brekke (2020) lists the assumptions as:

- M1 New targets are born according to a Poisson process with intensity μ(x).
- M2 Existing targets survive from time step *k-1* to *k* with probability $P_s(x_{k-1})$.
- M3 The motion of a surviving target is given by $f_x(x_k|x_{k-1})$
- M4 A target with state x_k generates a measurement z_k with probability $P_D(x_k)$
- M5 Clutter measurements occur according to a Poisson process with intensity $\lambda(z)$.
- M6 The measurements of a detected target is related to the state according to $f_z(z_k|x_k)$.
- M7 The number n of targets is constant and known (i.e zero birth intensity and death probability).
- M8 The single target motion model and the likelihood are Gaussian-linear:

$$f_x(x_k|x_{k-1}) = \mathcal{N}(x_k; Fx_{k_1}, Q) \tag{3.22}$$

$$f_z(z_k; x_k) = \mathcal{N}(z_k; Hx_k, R) \tag{3.23}$$

- M9 At the end of the previous estimation cycle, the posterior densities of the targets are independent and Gaussian

$$p^t_{k-1}(x^t_{k-1}) = (x^t_{k-1}; \hat{x}^t_{k-1}, P^t_{k-1}). \tag{3.24}$$

- M10 Target t has a constant detection probability P^t_d

- M11 The clutter Poisson process has constant intensity λ

The JPDA extends and generalized the Probabilistic Data association filter (PDAF) which is used for single target tracking (Brekke (2020), page 135). Understanding the PDAF is therefore key to understanding the JPDA.

3.4.2 PDAF

This section present the PDAF filter as seen in (bar shalom and Daum (2010)). To understand the PDAF it is first necessary to have visited the underlying issue of associating measurements to tracks. For a track given by the state vector **x** and a chosen transition model one can predict where the next measurement $\hat{Z}$ will be found. However, due to measurement uncertainty one cannot guarantee that the prediction and actual measurement will align. The can be accounted for by establishing a *validation gate*, which assigns a region around the predicted measurement where there is a high probability that a measurement will be detected. However, due to clutter several measurement might fall within the same validation gate, or two targets might have validation gates that overlap. The situation is illustrated in Figure 3.14, where the measurement Z_2 can be associated with both $\hat{Z}_1$ and $\hat{Z}_2$. Furthermore the measurement Z_3, which stems from clutter, can be attributed to $\hat{Z}_2$. The PDAF uses a Minimum Mean Square Error (MMSE) approach to associate these measurements. The conditional mean of each target state is found according to (3.25), where Z is the measurement data, A_i is all events in a set of M mutually exclusive association events. β_i represents the conditional probabilities for each event given the measurements, denoted $\beta_i \triangleq P\{A_i|Z\}$.

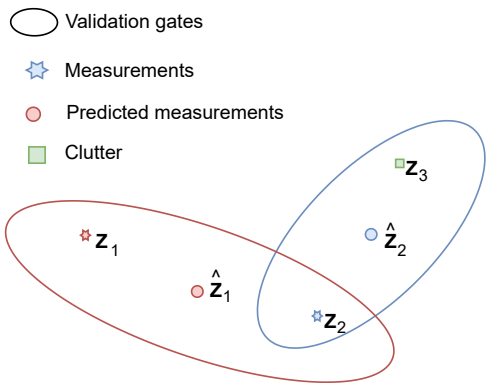


Figure 3.14: The probabilistic data association problem.

$$\hat{x}^{\text{MMSE}} = E[x|Z] = E\{E[x|A, Z]|Z\} \tag{3.25}$$

$$= \sum_{A_i \in A} E[x|A_i, Z]P\{A_i|Z\} \tag{3.26}$$

$$\triangleq \sum_{i=1}^{M} \hat{x}_i \beta_i \tag{3.27}$$

As the assumptions M3, M6, M8 and M9 hints at, the underlying model for the PDAF is the standard Kalman filter. For brevity the reader is referred to bar shalom and Daum (2010), page 10, for the integration of the PDA and Kalman filter into the PDAF.

3.4.3 Extension to multiple targets

The JPDA extends on the PDAF by considering multiple targets and their corresponding tracks during both association and track management.

3.4.4 Transition model

The kinematic model used in the prediction step (3.22) of the underlying Kalman filter is the **constant velocity model**, which can be seen in (3.28).

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{G}\mathbf{n} \tag{3.28}$$

For the 2D case where an object can move in the xy-plane the state vector is given by $\mathbf{x} = \begin{bmatrix} x & v_x & y & v_y \end{bmatrix}^T$. x and y corresponds to the objects position and v_x and v_y the objects velocity. The **A** and **G** matrix is given by 83.29) and where **n** is the process noise, assumed to be white.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{G} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{3.29}$$

3.4.5 Measurement model

The measurement model used in this thesis, corresponding to (3.22), is given by

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k \tag{3.30}$$

where $\mathbf{w}_k$ is a zero-mean Gaussian with a covariance matrix given by $\mathbf{R}$. Using the state vector $\mathbf{x} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ H is then given by (3.31)

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.31}$$

This linear model is used for measurements both from the LiDAR and the camera. In both cases it is a simplification of a nonlinear conversion which might degrade filter performance. For the camera the amount of distance each pixel covers increases as an object travels towards the horizon. Optimally this should be reflected in the measurement noise having bias. The LiDAR measurements also have noise bias, as they are collected in a polar coordinate system and converted to cartesian coordinates. The simplification is justified by two observations:

- The chosen detection range is limited to 10 m and the uncertainty of the measurements within this range should be negligible compared to other sources of uncertainty. GNSS noise, USV roll, pitch and yaw errors, inaccurate bounding boxes and inaccurate clusters are all sources that could dominate.
- The accuracy requirements both in regard to measurements and tracks are limited due to the use-case. The USV can be assumed to be moving slowly, and losing track or spawning false tracks, while not optimal, is not critical for the continous operation of the USV. The latest iteration of the intended garbage collection system had a capture diameter of 2 meters at the time of writing.

3.4.6 Initiator

The JPDA implementation used in this thesis overrides assumption M1 and M7 for the initiation of tracks. They are overridden as initialization is done utilizing a procedure consisting of a separate tracking filter tuned for spawning new tracks. The filter inherits the transition and measurement model, but differs by using a global nearest neighbour data associator (GNN) and time-step deleter. For each received measurement the filter tries to associate it with a track. If no association can be made a temporary track with a high initial uncertainty in velocity is created. The GNN associates measurements based on a distance hypothesiser, which yields what measurement is spatially closest to a tracks Kalman prediction. Tracks are created and transferred to the JPDA filter after a user-defined amount of measurements have been associated to the same track. If a measurement is not associated with the threshold within a set amount of timesteps the track is removed. The overruling ensures that the system can handle a dynamic environment where the amount of trash

within range might vary over time, spawning new tracks as necessary.

3.4.7 Deleter

Assumption M2 is also overridden to improve tracking performance. The deletion of tracks is done using a combination of a covariance based and a time based deleter. The latter is the same as for the initiator, but with a more relaxed time-constraint. The former removes tracks that have too high covariance. The combination was chosen to minimize unwanted behaviours in the tracks while also keeping the need for reestablishing tracks to a minimum. If the error between two subsequent measurements corresponding to a single object is large, the velocity of the object will scale accordingly. The next Kalman prediction can then be so far offset that new measurements fall outside the validation gate and cannot be associated. This leads to a ”loose” track, only being deleted when the time step deleter activates. Fortunately, the covariance of a ”loose” track increases rapidly, which can be caught by the covariance filter. In the case where measurements are close to perfectly aligned and the velocity therefore is close to zero the covariance might not approach the limit in a reasonable time. This can lead to tracks persisting long after being passed by the USV even when no new measurements are present. The time step deleter counters this by removing these tracks after a user-defined amount of time without new measurements.

4

Methodology

To develop the trash detection system and evaluate its performance three separate instances of data were acquired from the Otter platform in real-life testing scenarios. Each collection of data consisted of LiDAR, video recording and GNSS measurements which included both position and heading. This chapter presents the hardware used, how the data acquisition was performed and the datasets themselves. In addition the evaluation metrics used to measure performance and establishing a baseline for comparisons with existing solutions are presented and explained.

4.1 Hardware

This section presents the hardware used in this thesis. This includes both the computing platforms and the platform which the data was collected on, the Otter developed by Maritime Robotics. In addition the specific sensors with corresponding transformation matrices from the USVs body frame origin will be presented. For the data processing a standard computer platform running windows was used, as the final integration with the Otter hardware as well as real-time restrictions were left to future work.

4.1.1 PC

The first piece of hardware was an ordinary laptop with Windows 10 installed. It was used for developing and running the system, as well as visualizing the data. To interact with ROS *Windows Subsystem for Linux* (WSL) was also installed.

- CPU: Intel Core i5-8250 (1.60 GHz)

- RAM: 8 GB
- Operating system: Windows 10 and WSL

4.1.2 Otter platform

The Otter platform is a fully integrated USV platform intended for maritime environments. Produced by Maritime Robotics with the intention of being a multipurpose platform, it can be fitted with a wide array of sensor packages ranging from active sonar to 360° cameras and LiDAR. The platform is built with a catamaran-form factor, increasing stability and enabling a payload to be attached between the hulls to interact with the environment below the water surface. The bridge between the hulls contain the computing hardware, as well as acting as a base for the sensor suite. The Otter used in this project is depicted in Figure 4.1. The USV specifications are as follows:

Figure 4.1: Clean Sea Solutions Otter platform

- CPU: Intel Core i7-1270p (3.50 GHz)
- RAM: 8 GB
- Weight: 65 kg
- Operational time: 20 hrs
- Dimensions: 108x106.5x200cm
- Sensors: LiDAR, Front-facing camera, Global Navigation Satellite System (GNSS)

Figure 4.2 shows the relative distances that relate sensor measurements, where the origin of the body frame is given by the centre of the USV according to the dimension

specifications. Only the YZ-plane is shown, as every sensor is placed along the centre plane where x = 0.

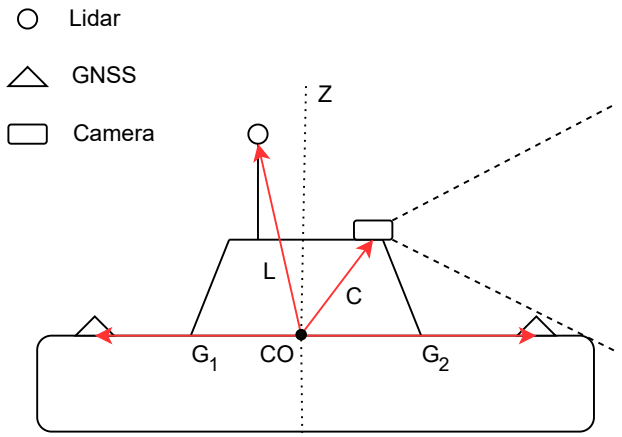


Figure 4.2: Sensor layout on for the Clean Sea Solution's Otter. CO is the Coordinate origin, L is the LiDAR lever arm, C is the camera lever arm and G_1 and G_2 are the lever arms for the GNSS receiver pair.

The sensors offset in angles were approximated in post-processing using the available data. The reason it could not be estimated during data-collection was a combination of limited time with the testing platform and a lack of benchmarks to determine the offsets. The LiDAR offset was visually estimated in LiDARview when the Otter drove straight towards a harbour wall, then turning left. It was found to be negligible compared to other sources of error in system. The camera could be aligned after the fact by comparing the actual location horizon and front of the USV with the intended. Although both procedures improves the accuracy they are far from a perfect solution. Unaligned sensors are therefore a weakness that have the potential to degrade results.

4.1.3 VLP-16

The VLP-16, also referred to as the ”puck”, is a LiDAR sensor produced and sold by Velodyne. Its small form factor and relatively low cost makes it a viable and accessible candidate for platforms that does not have the highest requirements for performance. With a range of 100 meters, 360 degree field of view in the horizontal plane and real-time capabilities it is well suited for smaller autonomous systems requiring situational awareness. The most relevant specifications for this thesis, based on (Velodyne), are listed below:

- Vertical channels: 16

- Range accuracy: $\pm$ 3 cm
- FOV (Vertical): $\pm$15 ° (30°)
- Vertical resolution: 2 °
- Rotation rate: 5-20 Hz

The sensor is installed on an arm extended vertically from the central base of the Otter, and is parallel to the horizon. This allows the LiDAR to have an uninterrupted view of the surroundings, but also means that only eight of the channels are directed towards the sea surface. As discussed in Section 3.3.1, the total vertical distance and angular distance leave a ”dead” zone in a circle around the USV, extending to a radius of 3.35 meters assuming no object is protruding more than 10 cm above the surface.

Combining the translation from the base with the angle offset found experimentally the total transformation matrix between the body frame and the LiDAR is given by (4.1), given in meters.

$$\mathbf{T}_{lid}^{b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.28 \\ 0 & 0 & 1 & 0.57 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

4.1.4 HIKVISION DS-2CD2045FWD-I

The HIKVISION DS-2CD2045FWD-I is a network camera produced by Hikvision. It is water resistant, performs well in dark conditions and can handle strong backlighting. The latter is especially useful in a maritime environment where the ocean can reflect the sun or other strong light sources. The focal length is adjustable, meaning that the FOV can be chosen based the on application. In the use-case of trash detection a wide FOV is preferable to cover more area, and the focal length and corresponding Field Of View (FOV) are therefore given based on this criteria. The camera specifications are presented below:

- Resolution: 2688x1520
- Focal width: 2.8 mm-12mm
- Vertical FOV: 109°
- Horizontal FOV: 60°

- Max Frames Per Second (FPS): 30

As the camera is aligned with the principal axis the rotation matrix is given by the identity matrix and the full transformation matrix is shown in (4.2), given in meters.

$$T_{cam}^{b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -0.04 \\ 0 & 0 & 1 & 0.27 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

4.1.5 ANN-MB-00-00

The GNSS equipment consisted of two ANN-MB-00-00 Multi-band GNSS antennas (Ublox (2023)), installed along the chosen principal axis of the Otter (y-axis pointing forwards). This ensured that a heading could be calculated by extracting a vector between the two positions. As (Kartverket) has both real and virtual RTK stations covering Norway and stores this data, it is possible to correct the GNSS measurements obtained during data collection in post-processing. However, due to the data-format available during testing, which data only included time, latitude and longitude, RTK correction was not integrated in the results. The position output from the GNSS pair is in the beginning of the vector starting at the GNSS antenna mounted on the back section of the Otter. The centre of origin (CO) is chosen as the central point between the receivers, given by an 94 cm offset forward of the two GNSS receivers. They are mounted along the principal axis pointing forward and the transformation is therefore (4.3), again given in meters. Note that the CO is approximately 0.30 m above the water surface. This can be discarded as all measurements are already in the xy-plane when the GNSS measurements are integrated in the system.

$$T_{GNSS}^{b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.94 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

4.2 Datasets and acquisition

Three instances of data were acquired for this thesis. All three were based on real sensor measurements collected from the Otter platform, but the mounting platform was alternated due to hardware availability. The first dataset was collected by remotely controlling the Otter in the harbour at Brattørkaia, Trondheim. The two latter only utilized the

upper mounting of the Otter, therefore being confined to land on a stationary rig. The acquired data consists of sensor streams from three sensors, Velodyne VLP16, Hikvision DS-2CD2045FWD-I and ANN-MB-00-00, corresponding to the sensors in Section 4.1. Furthermore a Samsung S10 was used to establish a ground truth for the final dataset D3. This section presents each dataset, what the intent was behind each and how this was handled in the acquisition. The acquisitions were done at three different dates, and some variations between the sets can be observed. This is in part due to environmental noise (such as lighting conditions and wind) as well as owing to the fact that weaknesses with the methods and repeatability in the preliminary gathering were corrected in subsequent acquisitions.

4.2.1 Preliminary data - D1

The first dataset was intended as a preliminary baseline to identify viable avenues of approach to solving the overall problem description. It was also needed to develop the system, as information about data formats, pre-processing steps and error handling was critical to ensure function in real-life scenarios. Dataset 1, henceforth referred to as D1, were collected on 31.01.2023. The sensor settings related to timing for the capture are presented below, where RPS is Rotations Per Second and MPS is Measurements Per Second:

- Camera FPS: 4 (0.25 sec)
- LiDAR RPS: 10 (0.1 sec)
- GNSS MPS: 20 (0.05 sec)

The dataset is split in two segments. The first capture was performed with a focus on LiDAR and camera detection viability. The sensor data was an uninterrupted stream of 17 minutes and 50 seconds, where both general navigation as well as specific experiments were carried out using a remote controller from the pier. The general navigation consisted of manually steering the Otter along different parts of the harbour. This was done to ensure that the sensors would be exposed to variations in speed, variations in distance to the edges of the pier as well as water disturbances. After this was performed a set of trash commonly found in waterways were introduced one at the time in the vicinity of the USV. The trash selected was intended to give a baseline of what could and could not be detected by LiDAR, both due to differing cross-section above the surface as well as reflective properties. The selection is shown in Figure 4.3.

The USV was then steered so that trash could be observed on approach, standstill at different ranges, passing as well as a overrun. An illustration of this process is shown in

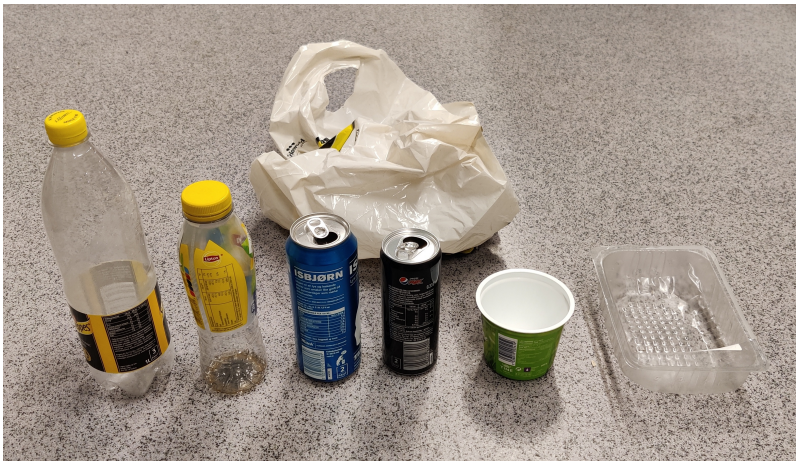


Figure 4.3: The different types of trash introduced in D1

Figure 4.4

Figure 4.4: An example of the trash being introduced in the vicinity of the USV

The second segment included a GNSS data stream. The main intent was to have all sources of data needed to run a tracking filter over time. It was performed with a generic approach, more akin to what the USV could observe in normal operation. The USV was controlled along the pier in a zig-zag pattern, being steered closer when coming upon trash and other floating objects such as kelp that were already present in the harbour. This approach also meant that the LiDAR was exposed to other edges than the pier, as the harbour had several docked boats of varying size. The total runtime of the second capture was 21 minutes.

4.2.2 Georeferencing validation set - D2

The second dataset was acquired to validate the camera georeferencing accuracy. It was performed indoors with the upper section of the Otter including the proper mounting of the LiDAR and camera. The experimental setup is shown in Figure 4.5. The camera distance above ground was 70.5 cm. Bottles were placed within view of the camera at different locations to test the accuracy from short to long range with different offsets (6 locations in total). As the system was stationary during the test the LiDAR was used as ground truth. The positions were determined by visual identification in LiDARView of the relevant measurements and subsequent distance extraction. The process is shown in Figure 4.6. The bounding boxes were manually set as the camera trash detection module were not able to distinguish the bottles in the indoor environment. The labeling was done on the full resolution image (2688x1520 px) to get the most accurate result, and a separate data set was created by adding white gaussian noise. This was done to simulate the lower resolution of the bounding boxes originating from the camera trash detector. The noise addition also served to test the sensitivity of the georeferencing method. One of the labeled images are shown in Figure 4.7.

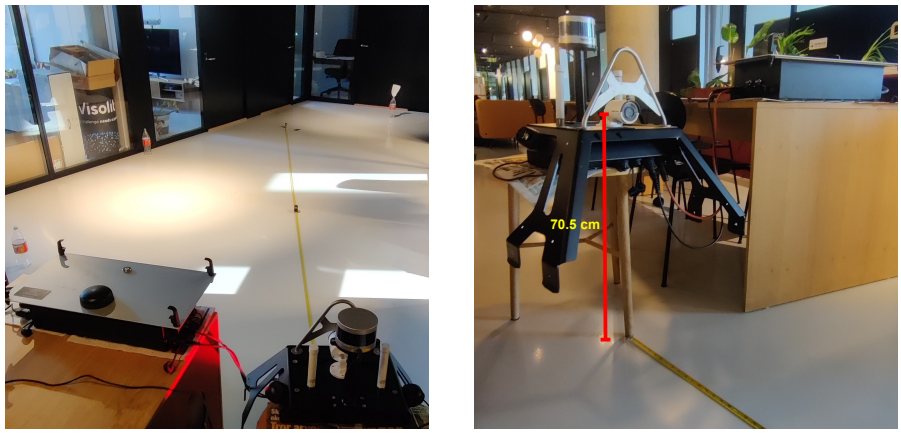


Figure 4.5: Experimental setup used to acquire dataset D2.

4.2.3 Track validation set - D3

The final dataset was acquired 16.06.2023 at Brattørkaia, Trondheim. This set was intended as a validation set for the accuracy of the processed LiDAR measurements and tracks produced by the JPDA filter. Some adjustments to the capture method were done to increase the accuracy of the measurements, as D1 had some weaknesses that were iden-

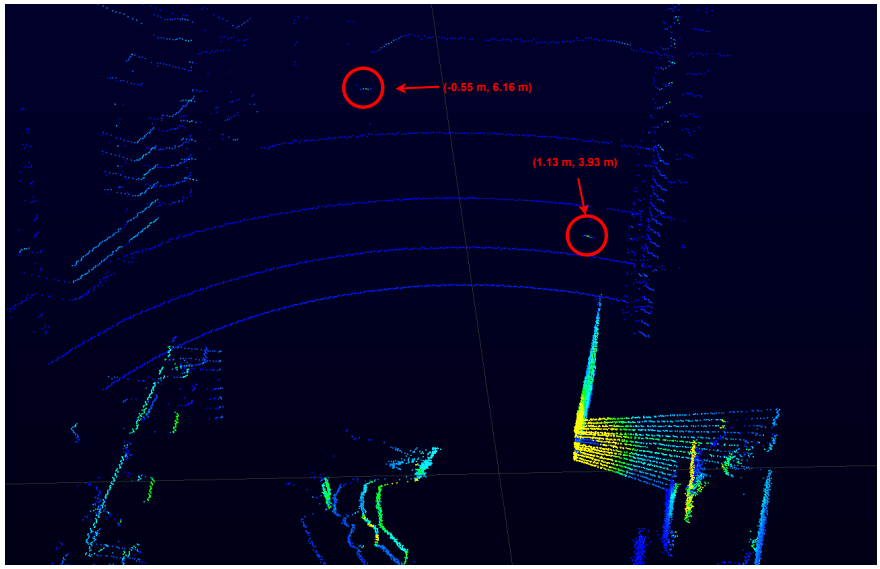


Figure 4.6: Extracting ground truth from LiDARView. The objects were removed after extraction and and confirmed to disappear from the view to ensure proper association.

tified during development. The main distinction was that all the measurements from the sensor suite were pipe-lined to a ROS-core running on a laptop and captured using the rosbag command. This allowed metadata from the camera stream to be collected and for the system to be integrated with the ROS ecosystem. The former was important to ensure that a frame could be tied to a specific timestamp. The timestamps for D1 had to be extrapolated from the start of the recording based on the frame rate, leading to large synchronization issues. The experimental setup consisted of the upper section of the Otter (Figure 4.8a) placed at the end of the pier, a Samsung S10 with GNSS enabled enclosed in a waterproof encasing (Figure 4.8b), several types of trash (Figure 4.8c) and a fishing rod (Figure 4.9b). Three segment of data was collected for each type of trash. The distance from the camera to the ocean surface was measured for each segment. The collection was performed by throwing the trash out in the water on the right side of the sensor platform, outside of sensing range. The distances thrown from the pier was varied to cover larger areas of the active tracking area. The fishing rod was then moved to a perpendicular part of the pier and the fishing line slowly retracted. This ensured that the trash ended up in range for both the camera and LiDAR. The process is illustrated in Figure 4.10. A final segment for each type of trash was captured where two pieces of stationary trash was introduced before starting the test. The trash with GPS was then thrown within the sensing range of the sensors and pulled in. This was performed to evaluate a setting with multiple targets.

Figure 4.7: Manually labeled image from D2

The total amount of segments were (4 types * 3 distances) + 4 stationary = 16.

4.3 Evaluation

This section presents the evaluation metrics used in this thesis. The metrics are split into two categories, the first being sensor specific and the second being overall system performance.

4.3.1 Sensor specific metrics

For each raw sensor stream being fed into the system a corresponding processed stream should be passed on to the JPDA filter. The LiDAR and georeferencing streams have different weaknesses and strengths, and the metrics used to validate the measurements should reflect this.

The georeferencing stream is based on a pre-trained YOLOv7 model outside the scope of this thesis. Metrics related to detection rates is therefore skipped. D2 was specifically acquired to test the precision of the georeferencing. The ground truth of each bottle is known and the georeferenced coordinate can be extracted from the bounding boxes. The metric chosen to validate this stream is therefore the Root Mean Square Error (RMSE), given by (4.4).

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(\hat{\mathbf{x}} - \mathbf{x})^2}{N}} \tag{4.4}$$

The RMSE yields the accuracy of the predicted position $\hat{\mathbf{x}}$ compared to the ground-

(a) The placement of the sensing platform during the capture of D3

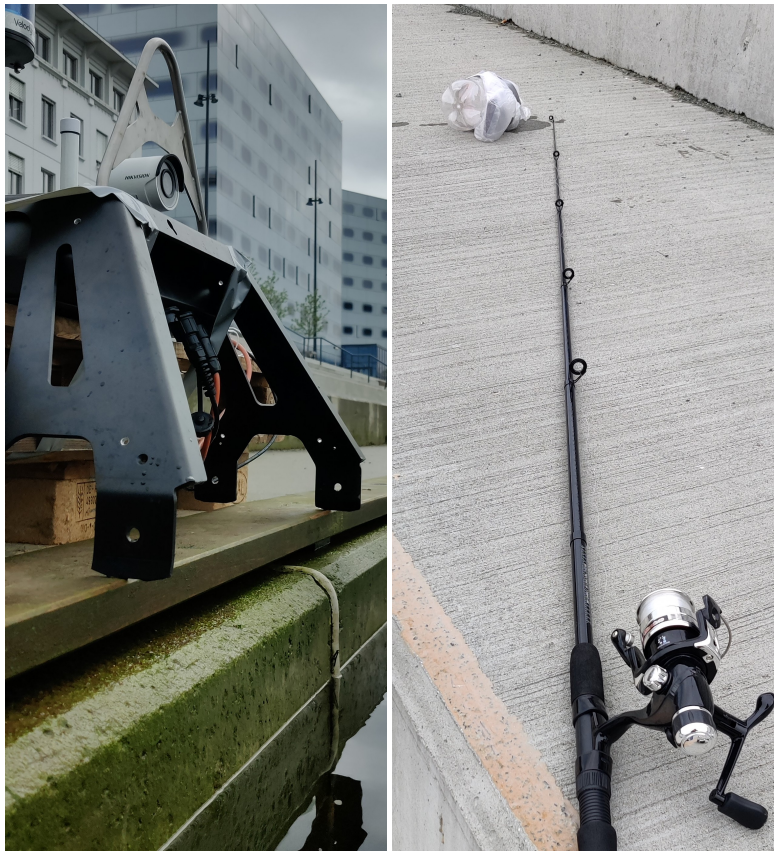
(b) The Samsung S10 used to establish the ground-truth of the trash

(c) The different types of trash tested, henceforth referred to as bag, plastic container, box, bottle

Figure 4.8: Test setup for D3

truth **x**. To see the effect of the undistortion and test the sensitivity to noise four scenarios were calculated (Distorted, No noise), (Undistorted, No noise), (Undistorted, $\pm 1^\circ$ vertical camera offset) and (Undistorted, ± 2 px vertical offset).

The LiDAR measurements are very precise, but lacks the ability to determine what physical object the measurement stems from. The LiDAR processing in this thesis has a binary outcome for each measurement received: **Valid detection** passed to the JPDA filter, or **Invalid detection** which is rejected. The most relevant metrics to test therefore becomes the ones related to detection rates. During the initial development, when only dataset D1 was available, a qualitative approach had to be used. This was both due to the LiDAR processing not being fully implemented, and that the data segment where trash was introduced did not include GNSS data. The latter meant that even after the system was completed the measurements from different time-steps could not be related. This is one



(a) The platforms offset from the water surface. Variable due to the tide.

(b) The fishing rod attached to the trash and Samsung S10

Figure 4.9: Height above water surface and the fishing rod-solution used in D3

of the base assumptions of the finished LiDAR processor, and running the process without the GNSS would therefore yield erroneous results. The first metric was therefore based on visual, manual confirmation in LiDARView. For each piece of trash introduced, 50 consecutive frames (5 seconds) of LiDAR measurements were visually inspected. Each frame was given a score of 1 if one or more points stemming from the trash was observed. No observed detection was given a 0. The score was then summed and averaged, yielding a True positive (TP) and False negative (FN) rate. In addition the largest amount of consecutive False negatives was recorded for each piece of trash. The video-stream was used to ensure that the LiDAR measurement under observation was assigned to the right object by comparing timestamps and visually confirming. A frame of LiDAR with the trash present and its corresponding visual confirmation is shown in Figure 4.11b and Figure 4.11a.

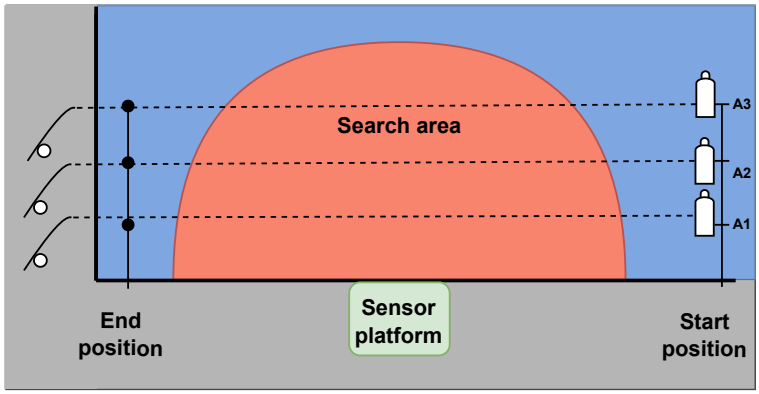


Figure 4.10: The capture procedure for each segment in D3. The naming convention A1-A3 is also used to separate the results from the trials

As the position of the sensor suite is static during D3 and ground truth data have been captured a quantitative approach should be avaiable. This is again done by calculating the RMSE for each measurement - ground truth pair.

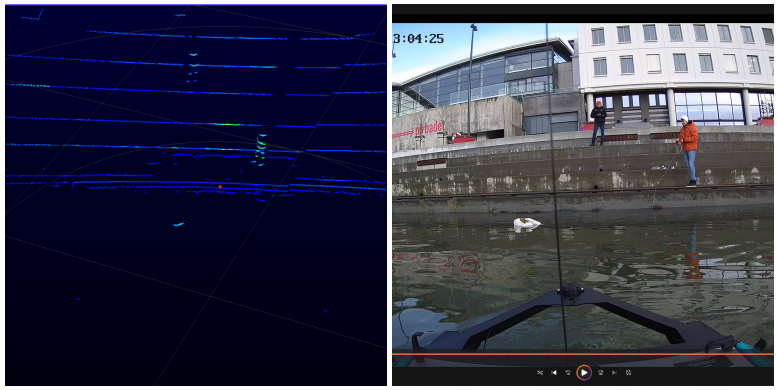
4.3.2 System performance metrics

Several system performance metrics are viable for testing the performance of the full system. The first two, Average Normalized estimation error squared (ANEES) and Average Normalized Innovation Squared (ANIS), are used for measuring filter consistency(Brekke (2020), page 69). For any time-step the NEES and NIS can be calculated according to (4.5) and (4.6) respectively. $(\hat{\mathbf{x}}_k - \mathbf{x}_k)$ is the state error of a track at time-step k, $\mathbf{P}_k^{-1}$ is the estimated covariance at time-step k, ε_k^v is the measurement residual and $\mathbf{S}_k^{-1}$ is the innovation covariance. Averaging over time yields ANEES and ANIS. To use these metrics it is necessary to construct a upper and lower bound based on a chosen confidence interval. The bounds can be calculated as the inverse χ^2 cumulative distribution function with $N_d/2$ and $N_s/2$ degrees of freedom respectively. N_d is the amount of realizations used to find ANEES and N_s is the amount of realizations used to find NIS.

$$\varepsilon_k = (\hat{\mathbf{x}}_k - \mathbf{x}_k)^T \mathbf{P}_k^{-1} (\hat{\mathbf{x}}_k - \mathbf{x}_k) \tag{4.5}$$

$$\varepsilon_k^v = \boldsymbol{\nu}^T \mathbf{S}_k^{-1} \boldsymbol{\nu}_k \tag{4.6}$$

ANEES is a valuable metric as it indicates if the state errors of the tracks is of the same magnitude as the state covariance. If ANEES is high it is a sign that the filter is overcon-



(a) LiDARview of one frame with measurement from plastic bag visible.

(b) The corresponding video frame used to match object and measurements.

Figure 4.11: The visual detection and confirmation process.

fident in its state estimate. ANIS enables testing of the innovation covariance consistency. The metric indicates if the predicted measurement error is of the same magnitude as the innovation covariance. A low ANIS value indicates that the filter is underconfident in its measurements.

For the JPDA the performance metrics evaluated is the Track probability of detection (TPD), track time (TT) and the track fragmentation rate (TRF). TPD is given by tracked targets divided by total number of targets, TT is the time period of the longest track segment and TFR is defined as the amount of tracks necessary to track a single target.

Finally, the full proof-of-concept of the system is evaluated. This is based on the second segment in D1 where GNSS measurements are available for USV positioning. This allows the system to process real-life data continuously over a longer period, in a dynamic environment where both trash and obstacles are present. As no ground-truth was captured the performance is evaluated by looking at detections that can be visually confirmed, the ability to filter out measurements from the surroundings and the overall tracking performance.

5

Implementation

This chapter presents how the full pipeline from raw sensor data to track output for detected trash is implemented. Section 5.1 presents the software tools used, Section 5.2 presents some of the main implementation choices that were made during development, while Section 5.3 presents the system architecture and the main modules directly related to the theory in Chapter 3.

5.1 Software tools

This section presents the programming language and software needed to implement and run the system.

5.1.1 ROS

Robot Operating System (ROS) is an established open-source framework for development of robotic applications (OpenRobotics (a)). Due to its modular and distributed approach to software hierarchy, where each process is a *Node*, it is well suited to create standalone packages that can be integrated into existing systems across different platforms. Each node can *publish* information to *topics*, and listen on topic by *subscribing* to them. This allows information to flow between nodes and lets each node process its task independently. ROS is also language agnostic, which means that programming languages that usually are non-compatible are able to communicate and interact over a common interface.

For this thesis the main use of ROS is to interact with the USV over a shared framework, which can handle asynchronous data-streams and relate each measurement with

common timestamps. This was done utilizing three packages, the first two being *video_stream_opencv* (OpenRobotics (b)) and *velodyne* (O'Quin) while the final being an internal package from Clean Sea Solutions. The latter intercepts the GNSS measurements sent by the GNSS receivers to the on-board computer and republished them on a topic. The former two nodes publishes to two other topics, namely */camera/image_raw* and */velodyne_packets*. All three can be recorded in the same file using the *rosbag* command, creating a rosbag-file. The rosbag can then be replayed later, emulating the captured data-streams by republishing all data to the original topics in chronological order.

To ease future integration into the ROS ecosystem the software in this thesis is developed with the ROS methodology in mind. Each sensor-stream is therefore assigned its own module, which is implemented so that it can process one measurement at the time as they arrive. After processing the measurement the module publishes a corresponding data-packet and keeps any internal states for the next measurement. Furthermore, the order of processing is based on the corresponding timestamps given by ROS for each measurement, with the oldest taking priority.

5.1.2 Programming

The main programming language used in this thesis is Python. The was in large parts due to ROS being language agnostic, which leaves it up to the developer to choose a programming language based on the needs of the project. For this project two main arguments laid the foundation for using Python. Foremost, the ease of prototyping and inclusion of open-source packages speeds up the development process. This is important when the overarching scope is large and the development time is limited. The access to open-source packages meant that subsystems which implementation fell outside of the scope could be offloaded to established frameworks. A comprehensive list of all packages imported and what purpose they served can be found in Table 5.1. The second argument is that the real-time processing speed was a secondary priority during the first iteration of this project. The main goal was to develop and test the viability of a trash detection system based on a fusion of sensors. It did not matter how fast the system ran if no meaningful information could be extracted from it. It was therefore decided that the first version of the system was to be developed in Python. Having established a proof-of-concept one could then revisit the notion of rewriting the code in a faster language (i.e. C++). This would however fall outside the scope of this thesis and be left as future work, if the implementation showed potential but real time capabilities were determined to be lacking.

Package	Offloaded task	Source
PyTorch	Trash object-detection module	Paszke et al. (2019)
Stonesoup	JPDA implementation	Hiles (2023)
velodyne_decoder	Velodyne packet decoder	Valgur (2023)
PIL	Image conversion	Clark (2015)
Pymap3d	Coordinate conversion	Scivision (2023)
Plotly	Plotting of JPDA	Inc. (2015)
Matplotlib	General plotting	Hunter (2007)
NumPy	Linear algebra	Harris et al. (2020)
Rosbag	rosbag support	Field et al. (2023)
scikit learn	Meanshift clustering	Buitinck et al. (2013)
OpenCV	LSD implementation	OpenCV (2015)
FFmpeg	mp4 reader	Tomar (2006)
vision_opencv	ROS image to CV2 image	Mihelich and Bowman (2023)

Table 5.1: The imported python packages, their use-case and their citations

5.2 Design and implementation

This section presents some of the main design and implementation challenges that had to be solved during the development of the system. Section 5.2.1 explains how the camera intrinsic and distortion parameters of the camera were found. Section 5.2.2 describes how measurements on different data formats were unified. Section 5.2.3 presents how issues related to synchronization was handled.

5.2.1 Camera calibration

Although the pinhole camera model from Section 3.3 yields a decent approximation of image formation for standard cameras, it does not account for different types of distortion in the image. This occurs due to the camera lens not being *rectilinear*, which leads to straight lines projected onto an image not staying straight. To correct for this it is necessary to find find the cameras unique intrinsic matrix and distortion parameters. One of the most common methods for camera calibration was first presented in Zhang (2000). A series of pictures taken with the camera to be calibrated is needed to perform the calibration, each image with a black and white checkerboard visible. The dimensions of the checkerboard and the physical size of each square is also necessary. The actual calibration was done according to OpenCV (2014). Figure 5.1 shows one of the 40 images used during calibration including the checkerboard detection. Equation 5.1 presents the intrinsic camera matrix and Equation 5.2 presents the distortion parameters. Figure 5.2a and Figure 5.2b shows a comparison of a image before and after undistortion. Lines that are supposed to

be straight have been straightened out after undistorting, with the clearest example being the window frame. The reprojection error was 0.114 pixels. This means that the points reprojected back to the scene using the calibrated matrix yields a sub-pixel accurate result. An important observation is that the undistortion scales and crops the image. This could potentially have an impact on the georeferencing process where pixel offsets lead to errors and parts of the image are left out. However, when the cropping was not performed the returned image was unacceptable, as seen in Figure 5.2c. This was identified as originating mainly due to a large k_3 parameter (-3.706).

Figure 5.1: The checkerboard detection for one of the images used for calibration.

$$I_{pin} = \begin{bmatrix} 2753 & 0 & 1344 \\ 0 & 2784 & 748 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

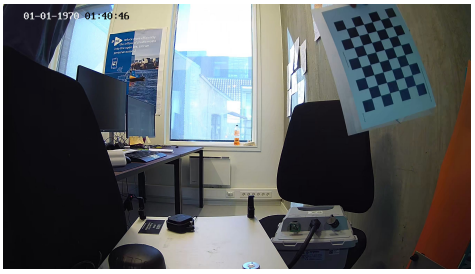
$$D_{pin} = \begin{bmatrix} -0.949 & 2.273 & 0.0283 & -0.009 & -3.706 \end{bmatrix} \tag{5.2}$$

The HikVision camera is defined as a bullet camera and should fall within the ”pinhole”-category. However, due to the large horizontal FOV another calibration method could also be viable, namely the fisheye model (Kannala and Brandt (2006)). The camera was therefore also calibrated according to this model (Using the procedure and code outlined in Jiang (2017)). Equation 5.3 presents the corresponding intrinsic matrix, Equation 5.4 presents the distortion parameters and Figure 5.2d shows the undistorted image. To iden-

tify if any of the camera calibration improved the performance the georeferencing validation set was processed both with and without undistortion, using both the cropped pinhole model and the fisheye model. The calibration determined to be best was then deployed on to produce the final results.

$$I_{fish} = \begin{bmatrix} 1673 & 0 & 1293 \\ 0 & 1686 & 768 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3}$$

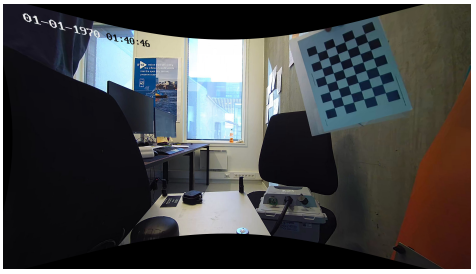
$$D_{fish} = \begin{bmatrix} -0.174 & 0.191 & -0.290 & 0.204 \end{bmatrix} \tag{5.4}$$



(a) Image before undistortion. The distortion effect can cleary be seen along the window frame

(b) Image after undistortion. The rectifying effect is most visible along the window frame.

(c) Undistortion without cropping. Setting the k_3 parameter to zero was the only way to rectify this.



(d) Image after undistortion using the fisheye model. Black borders are introduced to avoid cropping.

Figure 5.2: The different camera calibrations

5.2.2 Datastream unification

One central issue when developing a multi-sensor system is how to process the incoming data streams. In this thesis the processed measurement data is to be fed into the JPDA filter described in Section 3.4. This opens up the opportunity to feed the filter with measurements as they arrive. This is relatively trivial in ROS with its topic subscription and rosbag format. In this case measurements are published in chronological order and can be decoded using packages designed to bridge the gap between ROS and python. When the data is recorded using other formats, which was the case for dataset D1, there is however software overhead. In this case it is necessary to determine in which order the data arrived, after first stripping the relevant information and unifying the format. Two methods is considered for this thesis. The first is to do information extraction, merging and sorting in a separate process and saving the output to file in chronological order. The second option is to handle the streams dynamically as the measurements are being received. In this case only one measurement from each sensor is considered at a time and the measurement propagated to the system is the oldest of the three. As the intention is to connect the software to live sensors the latter, dynamic approach was deemed advantageous, as developing the system around this format means that the integration with the ROS ecosystem can be done almost seamlessly. The former would require an intermediary step that would have to perform many of the same tasks as the dynamic choice.

The dynamic solution interfacing with both rosbags and the alternative data formats in D1 is implemented by using pythonic data generators to simulate the data streams. These generators are memory efficient way to access and handle large amounts of data. This stems from the fact that a generator only processes the next iteration in its loop when called with the function *next(generator)*. This iteration can include loading the next member of a data structure from a file, as opposed to loading the file in full.

5.2.3 Synchronization

Another central issue of multi-sensor systems is how the synchronization should be handled. If there is a time offset between actual capture time and the related timestamp this can have dire consequences for the accuracy of the system, as the translation between frames described in Section 3.2 depends on both the distance traveled and the heading. Errors in the latter also results in increasingly worse errors for measurements as the distance increases. To illustrate how severe this problem can become one can look at a scenario where the USV starts a turn at a constant rate of 10° per second and the heading measurements are sampled at 10 Hz. Then assume that the USV is observing an object at

the edge of the permitted range with LiDAR, with a NED coordinate given by (x,y) = (0 m, 10 m). If the heading measurement and LiDAR timestamp is out of sync with 100 ms, i.e. due to being stamped at measurement initiation versus storage time, a 1° error in heading would be realized, which yields a new coordinate of: $x = 10 * cos(1°) = 10$ m , $y = 10 * sin(1°) = 0.17$ m. It is clear that majority of the error manifests itself perpendicular to the heading, with a discrepancy of 17 cm along the x-axis in this specific case. The situation is illustrated in Figure 5.3, where the USV and a frame aligned measurement are shown at two subsequent time steps. The red X illustrates the position in NED of the measurement right before being fed into the JPDA, while the blue circle signifies the actual object being measured. The discrepancy between the object and the aligned measurement at the next time step is shown on the right hand side. The sensor has received a measurement that is translated according to the actual heading change, but since the heading is lagging behind the assumed heading is still directly north. No correction is therefore performed and the JPDA is fed erroneous information. In the worst case this can lead to a bad velocity estimate and track loss. This example only shows the impact of an error in orientation, but these problems also extend to GNSS position, either over or under-compensating displacement depending on the offset direction.

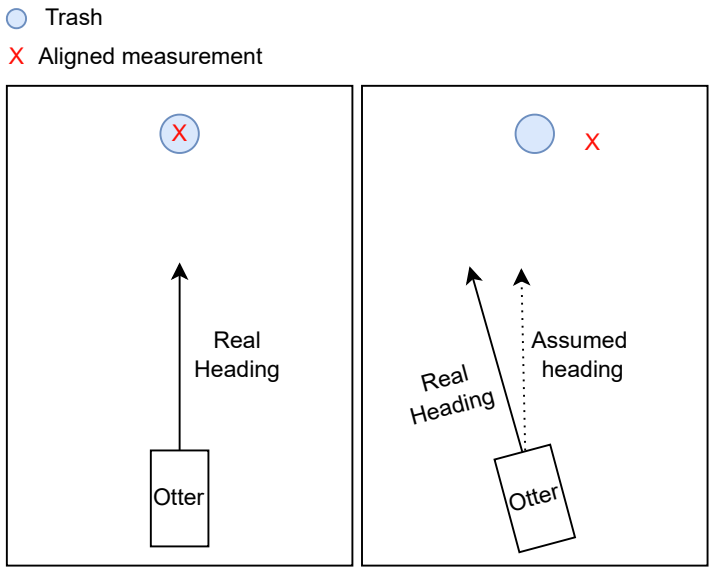


Figure 5.3: Graphical representation of the synchronization issues discussed in Section 5.2.3, given in the NED frame. The figure is for illustration purposes only and the measurement offsets and angles are not to scale.

With the large impact these issues can have on the accuracy and performance of the system it is clear that steps should be taken to mitigate the risk of asynchronous data. The first and foremost approach is to stamp all measurements according to the same internal clock whenever possible. This minimizes outside effects such as transmission delay and clock offsets. However, this is not always possible due to different measurement origins. This is the case for the data collection system used in this thesis. The Otter is connected wirelessly to a radio receiver that is then connected to a laptop. The recording for all three datasets were done on this laptop, but both how the data was captured (D1 differs from D2 and D3) and what hardware was available led to limitations that had to be handled manually.

The GNSS receivers measurements are intrinsically linked to the atomic clocks in the satellite network, meaning that the sensors passes on an ”internal” stamp at the time of calculation, The VLP-16 also yields an ”internal” stamp, which is connected to the clock on the Otter. The Hikvision-camera is the weakest link, not being able to embed timestamps in the metadata of each frame. This results in the timestamp not being resolved before arriving to the module responsible for storing or propagating the image into the system. In the case of dataset D1 the only timestamp available is the time the file was written. This means that every timestamp-image frame pair has to be extrapolated based on the initial timestamp, adding the frame number multiplied by the time difference between two frames. For the longer datasets in D1 this leads to further complications, as the theoretical offset between two frames (0.25 s at 4 FPS) and the actual offset differs slightly. Although giving a good initial guess it is therefore a necessity to tune the offset manually given a chosen starting time in a capture file.

For D2 and D3 the ROS framework is used, enabling the stamping of each measurement according to the same internal clock on the laptop running the recording setup. The clearest advantage is that each image frame has its own associated timestamp, removing the need for extrapolation. It does however expose measurements from all sensors to the transmission delay from the Otter to the laptop. Furthermore, the way the camera transmits the image frames by publishing to a local web-page, which the laptop then is able to interface with to fetch the image. This also introduces a delay, which in addition to being of varying length between captures also is introduced from the beginning of a capture. This means that every rosbag capture requires manual tuning to synchronize the LiDAR and camera measurements. To counteract these shortcomings in synchronization a rework of the collection system both on the software and hardware side is needed, which was found to be outside the scope of this thesis.

5.3 System architecture

This section presents how the system architecture and the modules it consists of are implemented. Section 5.3.1 presents the overarching structure and information flow. The next sections dives deeper into the implementation of each individual module and how they relate to the theory presented in Chapter 3. Code lines marked with # represents the pseudocode of the actual code, replaced for code readability. For brevity only the main modules related to the theory section are shown. The reader is referred to (Vormdal (2023)) for the full source code of the system.

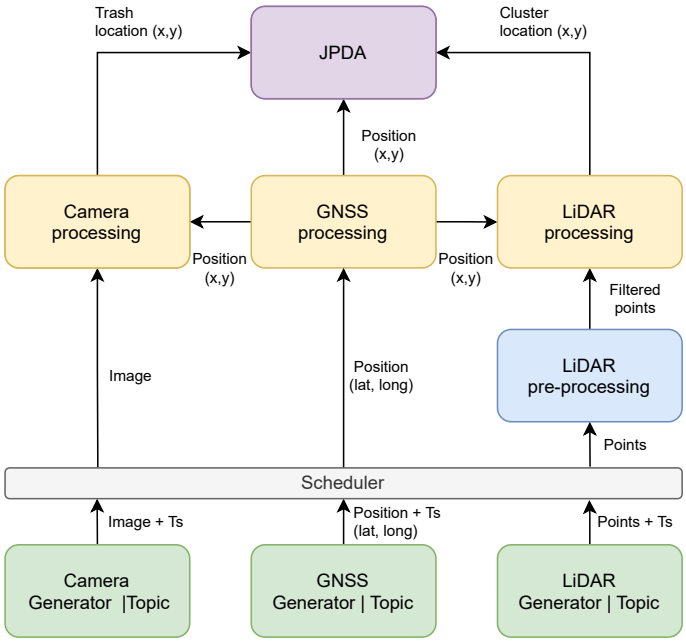


Figure 5.4: Visualization of the modular system architecture. Also presents the information flow for data streams based on both data generators and ROS topics.

5.3.1 Overview

An overview of the system architecture, including the different modules and their relations, is illustrated in Figure 5.4. The lowest layer consists of three modules corresponding to one sensor each, and is tasked with distributing measurements as they arrive. These measurements either arrive by subscribing to ROS topics as in D2 and D3, or by iterating step by step through each measurement file, as collected in D1. If the data-streams originates

from ROS topics the measurements are packaged on format that can be easily converted to python objects using readily available packages. This means that the data can be passed straight to the scheduler, which is tasked with deciding which measurement is next in line for processing. In the case of measurements from D1, the generators are also responsible for formatting data to the same format as received from a rosbag. The scheduler passes the next measurement in line to the correct processing module. Each processing module then converts the raw data to measurements conforming with the JPDA assumptions and passes them to the filter. The JPDA filter then returns tracks of the observed trash.

5.3.2 LiDAR generator

The generator for the LiDAR data is shown in Code 5.1. This is the simplest generator as LiDAR packets either originated from .pcap or ros velodyne_packets, both of which could be read using the python package velodyne_decoder referenced in Section 5.1.2. The generator takes the file to read from and the start stamp as inputs. *ros* is a control variable selecting the input type. *frame[0]* returns the time stamp of the current data packet, and while the start stamp is higher the packets are discarded. This ”rolls” the data packets forward until the wanted timestamp is reached, and is a feature in every implemented generator. The first data packet with a time stamp after the start time will go to *yield*, where it wills stay until the function is called with *next()*.

```
def get_raw_lidar_data(rosbag, topic, start_stamp, ros = False):
    config = vd.Config(model='VLP-16', rpm=600)
    if ros == True:
        for stamp, points, _ in vd.read_bag(rosbag, config, topic):
            stamp = stamp.to_sec()
            if start_stamp > stamp:
                continue
            yield [stamp, points]
    else:
        for stamp, points in vd.read_pcap(raw_lidar_data, config):
            if start_stamp > stamp:
                continue
            yield [stamp, points]
```

Code 5.1: The LiDAR data generator

5.3.3 Camera generator

The generator for the camera is shown in Code 5.2. The intrinsic and distortion matrices are cast in *intr* and *dist*. If the *video* being passed is in the form of a rosbag the correct

topic is read and each image is converted to a python-friendly format before being undistorted. The timestamp is corrected with the user defined *offset* and the generator yields the corrected data. If the video is on a .mp4 format as in D1, the metadata *end_time* stamp, video *duration* and number of frames *frame_num* is extracted and used to calculate the correct frame corresponding to the chosen start stamp. The video is rolled forward to this frame and the remainder of frames are iterated over, keeping track of the new stamp offset. Each frame is again undistorted before being yielded by the generator.

```
def get_camera_frame(video, start_stamp, video_path = None, ros = False,
    topic = None, offset = 0):
    intr = C_matrix #
    dist = D_matrix #

    if ros == True:
        for _, msg, t in video.read_messages(topics=topic):
            t = t.to_sec()-offset
            cv_img = bridge.compressed_imgmsg_to_cv2(msg, "passthrough")
            cv_img = cv2.undistort(cv_img, intr, dist, None, None)
            cv_img = cv2.UMat.get(cv_img)
            if start_stamp > t:
                continue
            yield [t,cv_img]
    else:
        meta_data = ffmpeg.probe(video_path)
        end_time, duration, frame_num = get_metadata()                    #
        video_offset = extrapolate_offset(start_stamp, end_time,
                                          duration, frame_num) #
        video.set(cv2.CAP_PROP_POS_FRAMES, video_offset)
        success = True
        for i in range(frame_num):
            success, img = video.read()
            stamp = stamp + 0.25
            if start_stamp > stamp or not success:
                continue
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            cv_img = cv2.undistort(cv_img, intr, dist, None, None)
            cv_img = cv2.UMat.get(cv_img)
            yield [stamp, cv_img]
```

Code 5.2: The camera data generator

5.3.4 GNSS generator

The implementation of the GNSS generator is shown in Code 5.3. The largest distinction from the other generators is that the reference point in geodetic coordinates is subject to

change depending on what is being visualized. The variables *relative_pos*, *set_start_pos* and *start_pos* determines where the geodetic origin of the NED-plane is. In the case of the JPDA visualization this is set as a pre-defined point at a corner of Brattørkaia, while for the visualizations of D1 it is set as the geodetic coordinate at the chosen initial time-step. The function *g2n* stands for *geodetic2ned* and is imported from the package pymap3d. In the case of D1 the GNSS data is supplied as a .txt-file, requiring reading the formats line by line. GPGGA is the header for lines containing positioning data. For each package of positioning data there is also a corresponding GPHDT package containing the heading.

```
def get_position(data_stream, start_stamp, relative_pos, date = None,
                                        ros = False, topic = None):
    set_start_pos = False
    heading = 0
    if ros == True:
      for _, msg, t in data_stream.read_messages(topics=topic):
          t = t.to_sec
          if start_stamp > t:
            continue
          heading, lat, lon = get_gnss_ros(msg) #
          if set_start_pos == False:
              start_pos = [ned_origin[0], ned_origin[1]] #
              set_start_pos = True
          ned = g2n(lat, lon, 0, start_pos[0], start_pos[1]) #
          yield [t, [ned[0], ned[1], heading]]
    else:
        file = open(data_stream, 'r')
        frames = file.readlines()
        for i, frame in enumerate(frames):
            if frame[1:6] == 'GPGGA':
                time, lat, lon = strip_and_format_gnss_string(frame)  #
                heading = find_corresponding_gphdt(frame)             #
                timestamp = time_to_ts(time)                          #
                if start_stamp > timestamp:
                    continue
                if set_start_pos == False:
                    if relative_pos == True:
                        start_pos= [float(lat), float(lon)]
                    else:
                        start_pos = [ned_origin[0], ned_origin[1]] #
                    set_start_pos = True
                ned = g2n(lat, lon, 0, start_pos[0], start_pos[1]) #
                yield [timestamp, [ned[0], ned[1], heading]]
```

Code 5.3: The GNSS data generator

5.3.5 LiDAR processing

Although being abstracted out into two separate modules in the system overview the LiDAR pre-processing and processing is integrated into one overarching function, shown in Code 5.4. The LiDAR pre-processing is done by direct filtering, which means that every point outside set thresholds are rejected. This reduces the processing demand significantly, which is crucial for the processing step. An added advantage is that it can be run in O(n)-time, since it only requires one iteration through the point-cloud list. The processing steps described in Section 3.3.1 are done on line 16-20. The function *update_lines* is an addition not covered in theory. It was added to counteract a weakness of the line segments returned from the LSD, which in some instances returns several line segments where there should only be one, leaving gaps. Each line segment detected is therefore passed on to the next three subsequent time-steps. This requires that each line segment is oriented according to the current heading and position and that old segments are removed as they pass out of scope. To improve performance segments that are fully covered by other segments are also found using the line intersection method and removed.

```
1  def get_lidar_measurements(detector, lidar_data, position_delta, radius,
       intensity, heigth, current_lines):
2      if np.size(lidar_data) == 0:
3          return None, None, None, None
4      frame_points = []
5      unique_points = np.unique(lidar_data, axis=0)
6
7      for point in unique_points:
8          if np.linalg.norm([point[0], point[1]])<= radius and point[2] <
       heigth:
9              frame_points.append(point)
10     frame_points = np.array(frame_points)
11     lines = []
12     new_lines = []
13     lidar_measurements = []
14
15     if np.size(frame_points) != 0:
16         lidar_image = lidar_to_image(frame_points)
17         lidar_image = cv2.GaussianBlur(lidar_image,(3,3),0)
18         new_lines = detector.detect(lidar_image)
19         lines = update_lines(new_lines, current_lines, position_delta)
20         lidar_measurements = clean_on_line_intersect(lines, frame_points)
21
22     return lidar_measurements, lines, frame_points
```

Code 5.4: The overarching LiDAR processor

5.3.6 Camera processing

The camera processing function is shown in Code 5.5. The image being processed is first being resized to a resolution that is similar to the one used to train the object detection model as to not impact the detection performance. Although the model was trained on 640x480 the image is resized to 672x380 in order to keep the original format intact, circumventing the stretching of pixels. *model()* is the initialized detector object, returning bounding boxes for each object detected in a frame. Each bounding box is scaled back to the original resolution. The scaling and rescaling process introduces a $\pm$ 2 pixel systemic error. It was deemed an acceptable compromise to not degrade the performance of the object detector, which even in best case scenarios frequently experienced bad detection rates. Detections that are above the horizon, below a lower confidence threshold or that are in the area of the image where the USV is visible are removed on line 10-12. *find_box_coordinate* calculates the horizontal centre of the bounding box and returns the box coordinate used for georeferencing. Each box is then georeferenced using the implementation shown in Code 5.6, corresponding to the theory presented in Section 3.1.3.

```
1  def detect_trash(image, model, rot, heigth):
2      image_res = cv2.resize(image, (672, 380))
3      predictions = model(image_res)
4      boxes = []
5      detections = []
6      world_coords = []
7      for row in predictions.pandas().xyxy[0].itertuples():
8          ymin, ymax = int(row.ymin *4), int(row.ymax *4)
9          xmin, xmax = int(row.xmin *4), int(row.xmax *4)
10         if row.confidence > 0.35 and ymin > 725 and ymax > 725:
11             if bbox in usv_rect: #
12               continue
13             detections.append([xmin, ymin,xmax, ymax])
14             box = find_box_coordinate(ymax, xmin, xmax)
15             boxes.append(box)
16     for b in boxes:
17         R = rotation_matrix(rad(-90.0+rot[0]), rot[1], rad(90.0+rot[2])) #
18         world_coord = georeference(b[1],b[2], R, [0.0,0.0, heigth])
19         if world_coord[0] < 10.0:
20             world_coords.append(world_coord)
21     return [detections, world_coords, boxes]
```

Code 5.5: The overarching camera processor

```
def georeference(u,v, R, t_wc):
    theta = ((u - 1344) / 2688)*np.radians(109)
    psi =  ((v - 760) / 1520)*np.radians(60)
    v_c = [np.tan(theta), np.tan(psi), 1]
    v_w = R@v_c + np.array([t_wc[0], t_wc[1], 0])
    s = -t_wc[2] / v_w[2]
    x_w = t_wc + s*v_w
    return x_w
```

Code 5.6: The georeferencing function

5.3.7 JPDA

The JPDA implementation is presented in Code 5.7. It is almost entirely based on predefined objects and functions from the package Stonesoup. The majority of the work in this thesis with regard to the JPDA consists of choosing the right components for the trash detection problem, tuning the parameters to ensure proper tracking and integrating the framework into the developed system.

Line 1-8 defines the underlying Kalman filter used to estimate each state vector corresponding to a trash object. The transition model is the CV model, where the white noise-component corresponding to the expected acceleration of each object is passed as an argument. The value of 0.001 was determined experimentally and was found to yield satisfactory results related to tracking performance and corresponding metrics. A low value indicates that the trash is expected to undergo little to no acceleration. This fits well with the assumption that the floating trash is mainly moved by environmental forces such as current and wind. Three measurement models were used, one for track initialization and one for each sensor. The *noise_covariance* parameter sets the expected uncertainty for each dimension of the measurements. In the case of the initiator (visible on line 4), this is set low (0.01 m) to ensure that only subsequent measurements actually measured to be within a certain distance can initialize tracks. The uncertainty for the sensors is set higher at [0.5, 0.5 m] for the camera and [0.25 m, 0.25 m] for the LiDAR. The higher value for the camera originates from the fact that the camera calculates distance through several steps where sources of error can be present, while the LiDAR directly measures distance. The values being higher than for the initializer is selected to relax the constraints for measurement-to-track association for already established tracks. This counteracts new measurements falling outside a narrow validation gate and being considered for new tracks. It also has the added benefit of adding an appropriate uncertainty to the track itself, which will scale when the track is lost. Three track deleters are cast, the first being time based with a narrow time window, used for initialization. This is done to filter out clutter and ensure that a

track is only accepted when the inital measurements are close in time as well as space.

The two other deleters on line 13-14 are used for the main filter and corresponds to the discussion in Section 3.4.7. Line 16-27 corresponds to the initator discussed in Section 3.4.6. The initial state vector is cast with uncertainty in the velocity components [0.3 m 0.3 m] corresponding to the process noise. This increases the size of the validation gate. The remaining parameters are set to zero as they are overwritten by the initial measurement state. The selected minimum associated measurements to pass a track to the main filter is set as 2 in *min_points*. This is based on the assumption that clutter is unlikely to appear close in time and distance over a short period. This does not account for disturbances such as wakes or other temporary water disturbances which might yield false measurements. In practice it was however observed to work as intended in the testing environment.

```
1  transition_model = CombinedLinearGaussianTransitionModel(
2                       [ConstantVelocity(0.001),ConstantVelocity(0.001)])
3
4  measurement_model = LinearGaussian(ndim_state=4, mapping=[0,2],
5                                     noise_covar=np.diag([0.1**2, 0.1**2]))
6
7  predictor    = KalmanPredictor(transition_model)
8  updater      = KalmanUpdater(measurement_model)
9  hypothesiser = PDAHypothesiser(predictor=predictor,
10                         updater = updater, clutter_spatial_density=0.125)
11 data_associator   = JPDA(hypothesiser=hypothesiser)
12 deleter_init      = UpdateTimeStepsDeleter(time_steps_since_update=3)
13 deleter_2         = UpdateTimeStepsDeleter(time_steps_since_update=35)
14 deleter           = CovarianceBasedDeleter(covar_trace_thresh=1.5)
15 multi_del     = CompositeDeleter([deleter, deleter_2], intersect = False)
16 init_hypothesiser = DistanceHypothesiser(predictor, updater,
17                     measure=Mahalanobis(), missed_distance=0.3)
18
19 init_data_associator = GNNWith2DAssignment(init_hypothesiser)
20
21 initiator = MultiMeasurementInitiator(
22     prior_state=GaussianState([[0], [0], [0], [0]], np.diag([0, 0.3, 0,
       0.3])),
23     measurement_model=measurement_model,
24     deleter=deleter_init,
25     data_associator=init_data_associator,
26     updater=updater,
27     min_points=2,)
```

Code 5.7: The JPDA implementation

Finally, the integration with the implemented system is shown in Code 5.8. The input *detector_generator* is a wrapper generator that chronologically yields processed measurements and their corresponding time-step. Each measurement are packaged as a Stonesoup *Detection* object with the sensor specific measurement model. The returned *tracker* object can then be called to produce tracks and associated information.

```
def track(detector_generator):
    tracker = MultiTargetMixtureTracker(
        initiator=initiator,
        deleter=multi_del,
        detector=detector_generator,
        data_associator=data_associator,
        updater=updater,)
    return tracker
```

Code 5.8: The JPDA integration

6

Results

6.1 Preliminary data - D1

This section presents the results obtained from the Preliminary dataset D1. Section 6.1.1 presents the results from the visual detection test described in Section 4.3.1. Section 6.1.2 presents the full implemented system running on a section of the second data segment of D1, as described in Section 4.3.2.

6.1.1 LiDAR detection

The results from the visual detection test can be seen in Table 6.1. TP is true positive, FN is False negative, CN is consecutive negatives and TP rate is the average rate of True positive over 50 samples. Note that two of the pieces of trash introduced in D1 (Figure 4.3), namely the 0.5 l bottle second from the left and the black aluminium can fourth from the left are not present. This is due to an error in the video capture procedure. The error meant that the two first pieces of trash never were visible in the video, consequently leaving no way to visually match measurement and object after the fact.

Metric	Bottle	Can	Beaker	Clear plastic	Bag
TP	43	42	50	30	47
FN	7	8	0	20	3
CN	2	4	0	15	3
TP rate	86%	84%	100%	60%	94%

Table 6.1: The results from the visual detection test, ordered as in Table 6.1, with the exception of the two pieces not present.

6.1.2 Global tracking

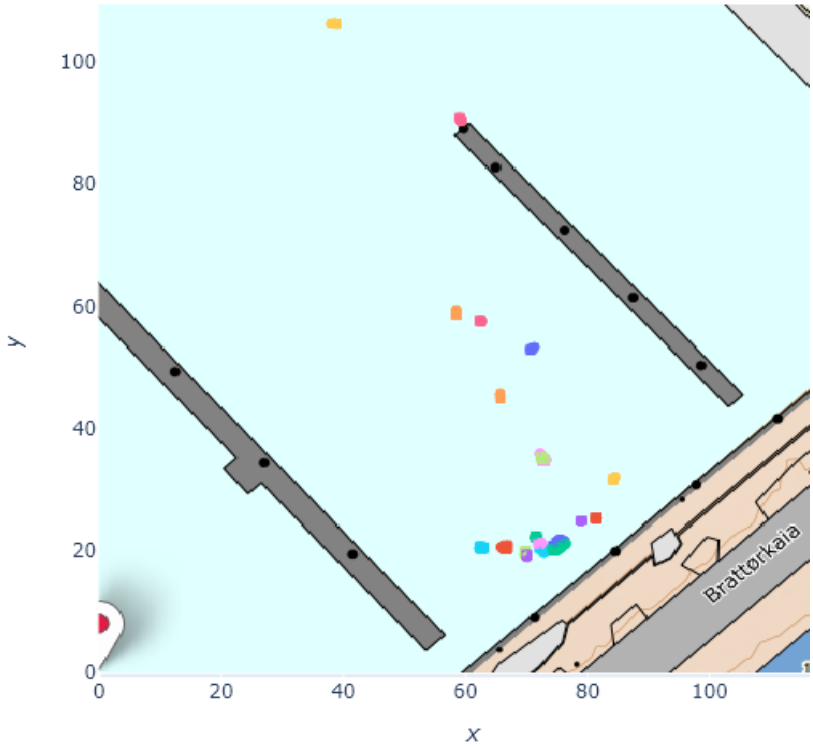


Figure 6.1: Global NED track results overlaid a map of Brattørkaia. Each color represents a unique track.

The full result from the global tracking test can be seen in Figure 6.1. Each color represents a unique track. The semi-transparent circles illustrates the associated covariance for the track at each time-step. The underlying map is set as the NED plane, with the origin corresponding with the geographic location with latitude = 63.4386345 and longitude = 10.3985848. The tracks positions on the map are determined by first calculating the distance in NED between the origin and the USV geodetic coordinate at a given timestep. The relative distance between the USV and a given track is then rotated by the current heading and added to the former result. This means that each track should be placed consistent with its actual geodetic position during the data acquisition. An animation of the underlying data processing can be seen by viewing the file *Global_tracking_animation.mp4* at (Vormdal (2023)) (also embedded in the README). The results were extracted from the

dataset D1, where GNSS data for the USV was available. The test ran for 100 seconds. The section was chosen as it was a representative sample of what the USV might encounter in an harbour environment. This included floating trash, organics, piers, floating jetties and larger boats. This allowed for verification of the main components of the system. Before running the system on the section the camera was manually synchronized to the rest of the data by comparing known movement from the GNSS with the video stream.

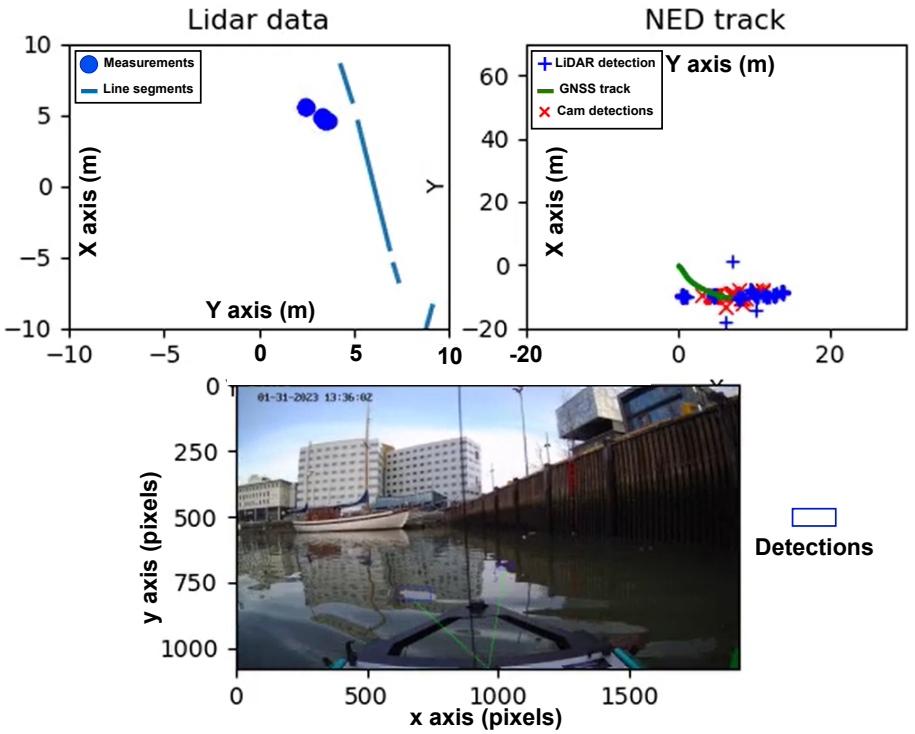
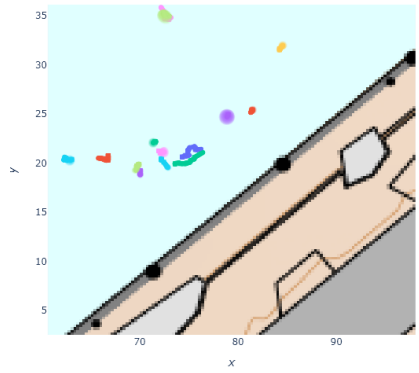


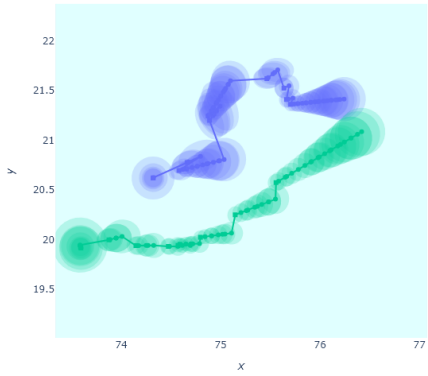
Figure 6.2: A snapshot of the different sensors and corresponding NED track at a single timestep.

Figure 6.2 shows a snapshot of the system as it is processing the sensor measurements. The top left image show the LiDAR processing, where the extracted line segments are illustrated by blue lines and the measurements being passed to the clustering is marked by blue dots. The top right image shows the NED track, where all sensor measurements observed up until that point are present. The lower image shows the camera frame, where the object detector shows the current detections with blue boxes. The latter also illustrates the problem of false positives, where the reflection in the water has been identified as trash. The low resolution is due to being part of an animation that was poorly optimized. Storing at higher resolution resulted in the system freezing due to working memory con-

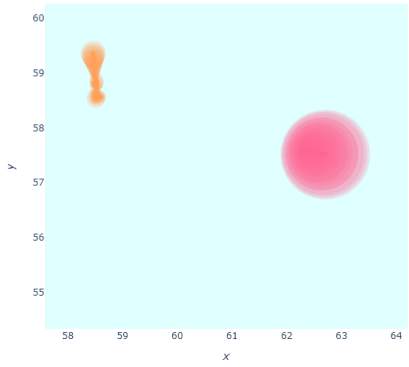
straints. Figure 6.3 presents a closer view of some of the more interesting tracks that were established.



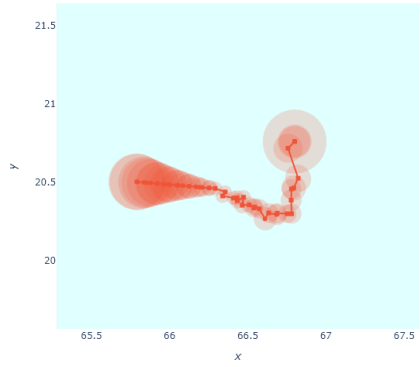
(a) Closer look at the tracks in the vicinity of a harbour wall



(b) Track where the time based deleter led to track deletion



(c) Track where the covariance based deleter led to deletion



(d) Track consisting of measurements from both camera and LiDAR

Figure 6.3: A selection of tracks showing some of the main attributes of the JPDA filter

6.2 Georeferencing validation set - D2

This sections presents the results from processing of the dataset D2, first described in Section 4.2.2. The six annotated images, their labels and the pixel coordinates of the corresponding bounding boxes are shown in Figure 6.4. The first row corresponds to

the distorted images, the second to undistorted using the pinhole model and the last to undistorted using the fisheye model. Note that Box 3 is not present in the right image, due to the cropping effect of the undistortion step. Table 6.2 gives the ground truth of each object extracted from the LiDAR, with the x-axis pointing into the image and the y-axis to the right. Table 6.3, Table 6.4 and Table 6.5 yields the georeferencing results for each calibration.

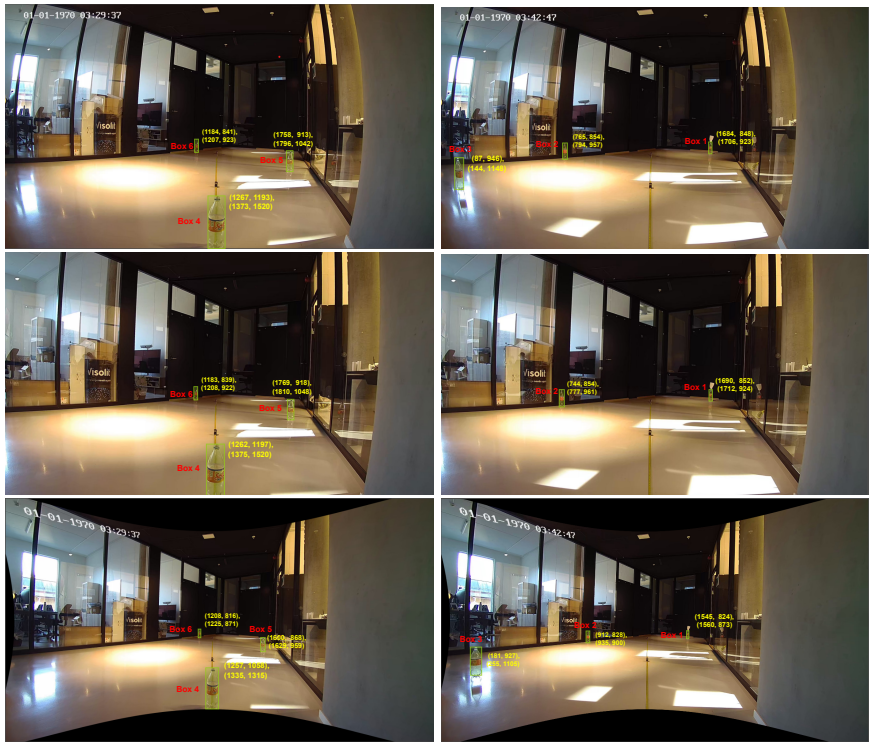


Figure 6.4: The six images used in the georeferencing experiment. Both the box labels (Box 1-6) and the bounding boxes upper left and bottom right corner pixel coordinates are overlaid.

GT (m)	Box 1	Box 2	Box 3	Box4	Box 5	Box 6
x	6.75	4.73	2.18	1.46	3.93	6.19
y	1.56	-1.75	-1.95	0.00	1.13	-0.55

Table 6.2: LiDAR Ground Truth (GT) for each box, extracted manually

Since the camera calibration with the lowest RMSE was the pinhole model this was the method chosen for the sensitivity analysis. This still holds true after accounting for the ”outlier” of Box 3, which when removed yields RMSE = 0.32 for the Distorted images.

Distorted (m)	Box 1	Box 2	Box 3	Box4	Box 5	Box 6
x (geo)	6.49	5.40	2.81	1.46	3.82	6.49
y (geo)	1.59	-2.20	-3.04	0.02	1.13	-0.66
x (err)	0.26	0.67	0.91	0.00	0.10	0.30
y (err)	0.03	0.45	1.09	0.02	0.00	0.11
RMSE	0.49					

Table 6.3: Distorted, No noise The georeferencing result from distorted image without noise. Error is given in absolute value

Pinhole (m)	Box 1	Box 2	Box 3	Box4	Box 5	Box 6
x (geo)	6.45	5.30	-	1.46	3.74	6.53
y (geo)	1.60	-2.22	-	0.02	1.24	-0.66
x (err)	0.30	0.57	-	0.00	0.19	0.34
y (err)	0.04	0.47	-	0.02	0.11	0.11
RMSE	0.32					

Table 6.4: Undistorted, pinhole The georeferencing result from undistorted image using the pinhole model (with cropping). Error is given in absolute value

Fisheye (m)	Box 1	Box 2	Box 3	Box4	Box 5	Box 6
x (geo)	9.28	7.53	3.15	1.99	5.35	9.44
y (geo)	1.04	-2.23	-2.98	-0.06	0.99	-0.83
x (err)	2.79	2.8	0.97	0.53	1.42	3.25
y (err)	0.52	0.48	1.03	0.06	0.14	0.28
RMSE	1.73					

Table 6.5: Undistorted, Fisheye The georeferencing result from undistorted image using the fisheye model. Error is presented in absolute value

Table 6.6 presents the change in calculated distances when a $\pm$ 2 pixel error is introduced on the vertical component (y). The reason for only testing on this component is that it will dominate at larger distances. This error is introduced to simulate the systemic error introduced from low bounding box resolution. Table 6.7 presents the change in calculated distances when introduced to a $\pm$ 1° in vertical camera offset.

Pinhole (Δ m)	Box 1	Box 2	Box4	Box 5	Box 6
Δ x (+2px)	0.07	0.05	0.01	0.02	0.08
Δ y (+2px)	0.02	0.03	0.00	0.01	0.00
Δ x (-2px)	0.15	0.05	0.00	0.03	0.08
Δ y (-2px)	0.02	0.02	0.00	0.01	0.01

Table 6.6: The georeferencing sensitivity to errors in the bounding boxes. Each value signifies the absolute distance between the original calculated value and the value with error present

Pinhole (Δ m)	Box 1	Box 2	Box4	Box 5	Box 6
Δ x (+1°)	1.15	0.74	0.05	0.35	1.18
Δ y (+1°)	0.30	0.33	0.00	0.12	0.13
Δ x (-1°)	0.84	0.58	0.05	0.28	0.86
Δ y (-1°)	0.20	0.25	0.00	0.10	0.09

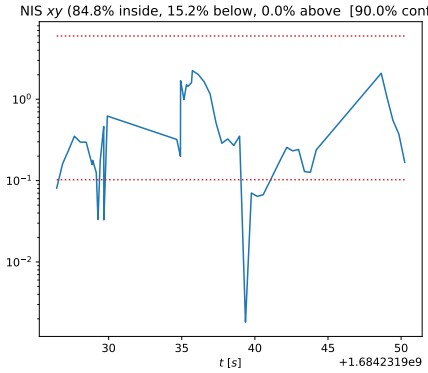
Table 6.7: The georeferencing sensitivity to errors vertical angle. Each value signifies the absolute distance between the original calculated value and the value with error present

6.3 Track validation set - D3

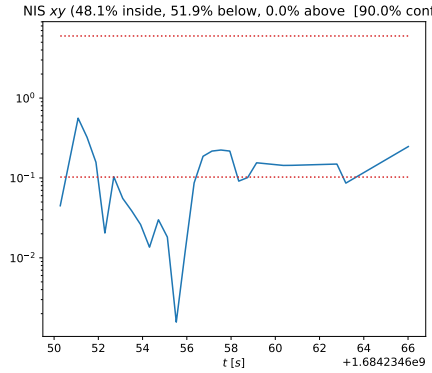
This section presents the results from the track validation set D3. Due to unforeseen problems with the captured ground-truth data, no RMSE, NEES or ANEES values are present in the results. In addition the trials related to the aluminium box are left out. Both issues are thoroughly covered in Section 7.3.1. Each trial is marked with A1-A4, where the three first corresponds to the trials with increasing distance from the sensor platform shown in Figure 4.10. A4 corresponds to the trials where two other bottles were introduced in addition to the trash being tested. These was intended as a qualitative stress test to observe the filters ability to track multiple tracks and observe track confusion tendencies, and evaluation metrics were therefore not extracted. Each subsection covers a specific piece of trash being tested, presenting the measurements originating from the senors in each trial, the JPDA track created based on this measurements and the corresponding JPDA evaluation metrics. Note that the JPDA plots are overlaid an actual map of Brattørkaia, related through the GNSS coordinate of the USV. The gray section in each image is a series of steps leading into the water, which at the beginning of data acquisition was covered with water. Due to the tide one step emerged from the water at the time, requiring the sensor platform to be moved to a lower step several times to stay close to the water surface. This is why some of the tracks are illustrated as being on land and why the USV location differs between trials. The offset from the origin in the x and y axis also stems from this, with the origin being located in the bottom left corner of the full scale map of Brattørkaia.

6.3.1 Evaluation metrics

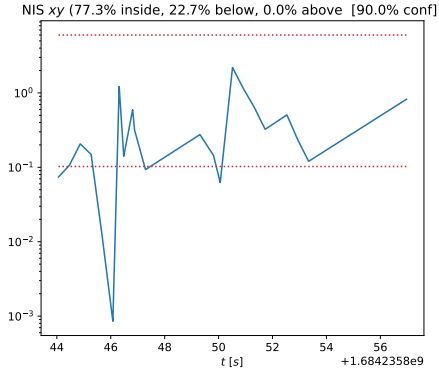
This section presents the evaluation metrics results that could be extracted from the trials. The 90 % confidence interval for the NIS given by two degrees of freedom is I = [0.102, 5.991]. NIS plots for the first trial for each piece of trash is shown in Figure 6.5. Table 6.8, Table 6.9 and Table 6.10 presents the evaluation metrics for each trial.



(a) NIS plot for Bottle (A1)



(b) NIS plot for Bag (A1)



(c) NIS plot for Container (A1)

Figure 6.5: Plots of NIS for the first trial of each type of trash

Bottle	ANIS	TPD	TT (s)	TRF
A1	0.556	2	31.87	1
A2	0.440	3	15.74	2
A3	0.206	2	23.80	1

Table 6.8: ANIS, TPD, TT and TRF for bottle tracks.

Bag	ANIS	TPD	TT (s)	TRF
A1	0.127	3	22.99	2
A2	0.254	1	17.75	1
A3	0.273	2	8.99	2

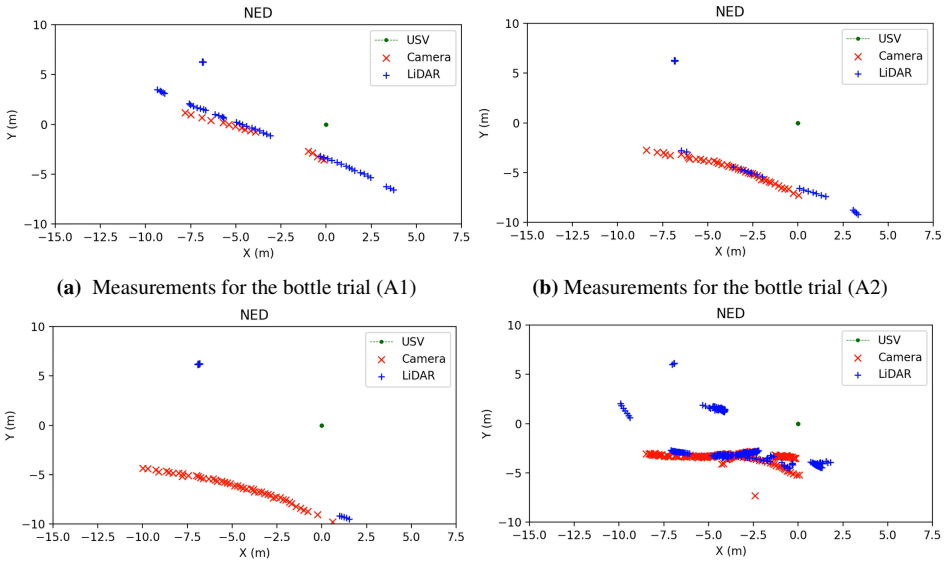
Table 6.9: ANIS, TPD, TT and TRF for bag tracks

Container	ANIS	TPD	TT (s)	TRF
A1	0.42	10	18.65	5
A2	-	7	-	3
A3	-	4	-	1

Table 6.10: ANIS, TPD, TT and TRF for container tracks

6.3.2 Plots

This section presents the plots of the measurements for each trial in addition to the corresponding JPDA-tracks. Each unique track is marked in a different color. The semi-transparent circles illustrates the associated covariance for the track at each time-step. The USV is illustrated at its geodetic coordinate for each trial, with the arrow signifying the heading during the test. Each object was introduced south-east of the USV position and reeled in towards north-west.



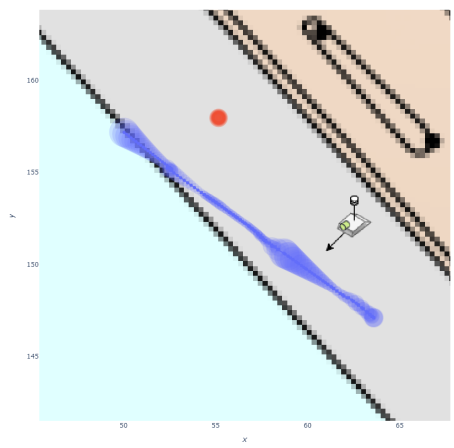
(a) Measurements for the bottle trial (A1)

(b) Measurements for the bottle trial (A2)

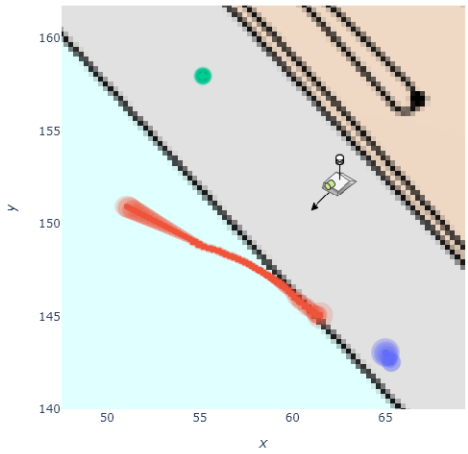
(c) Measurements for the bottle trial (A3)

(d) Measurements for the bottle trial (A4)

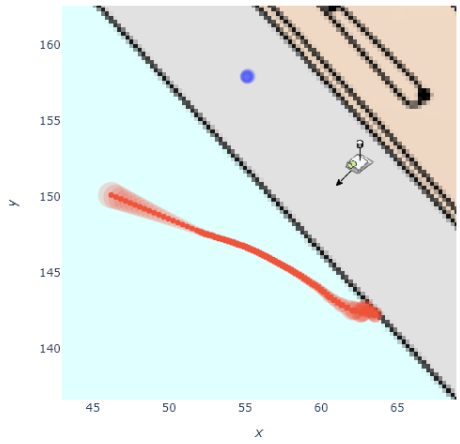
Figure 6.6: Plots of measurements for the bottle trials.



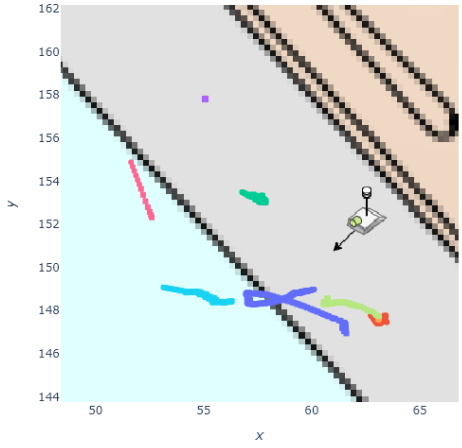
(a) JPDA track for bottle trial (A1)



(b) PDA track for bottle trial (A2)

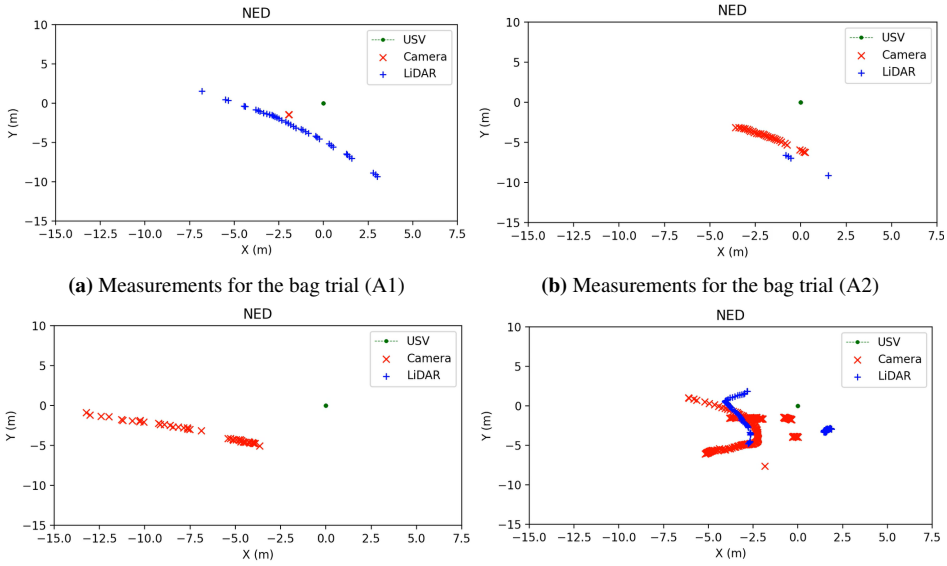


(c) JPDA track for bottle trial (A3)



(d) JPDA track for bottle trial (A4)

Figure 6.7: Plots of JPDA tracks for the bottle trials.



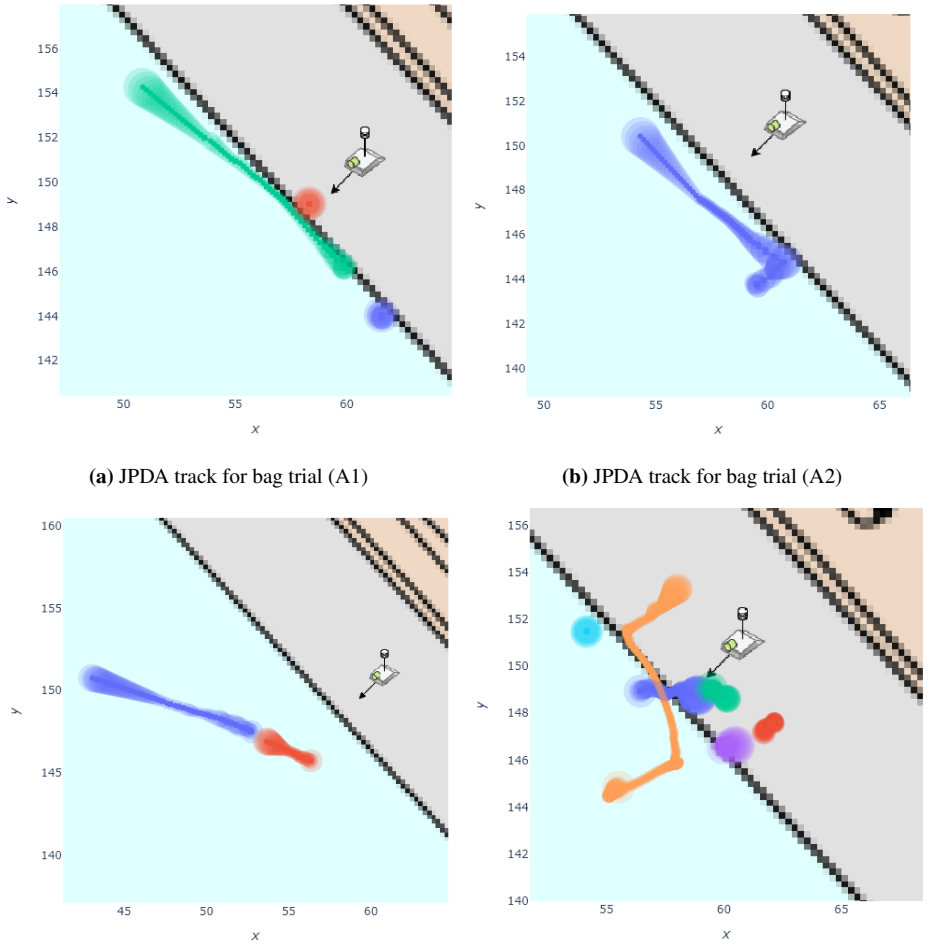
(a) Measurements for the bag trial (A1)

(b) Measurements for the bag trial (A2)

(c) Measurements for the bag trial (A3)

(d) Measurements for the bag trial (A4)

Figure 6.8: Plots of measurements for the bag trials.



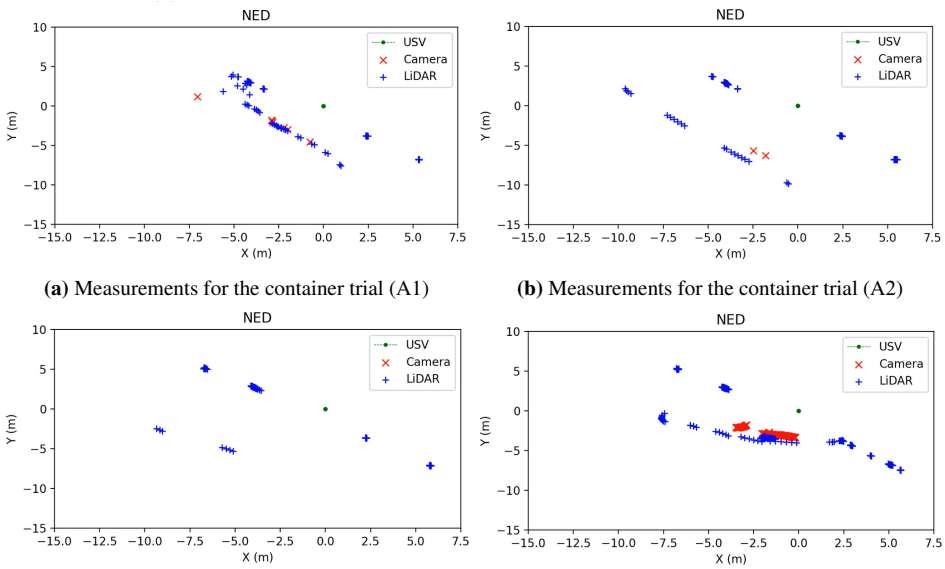
(a) JPDA track for bag trial (A1)

(b) JPDA track for bag trial (A2)

(c) JPDA track for bag trial (A3)

(d) JPDA track for bag trial (A4)

Figure 6.9: Plots of JPDA tracks for the bag trials.



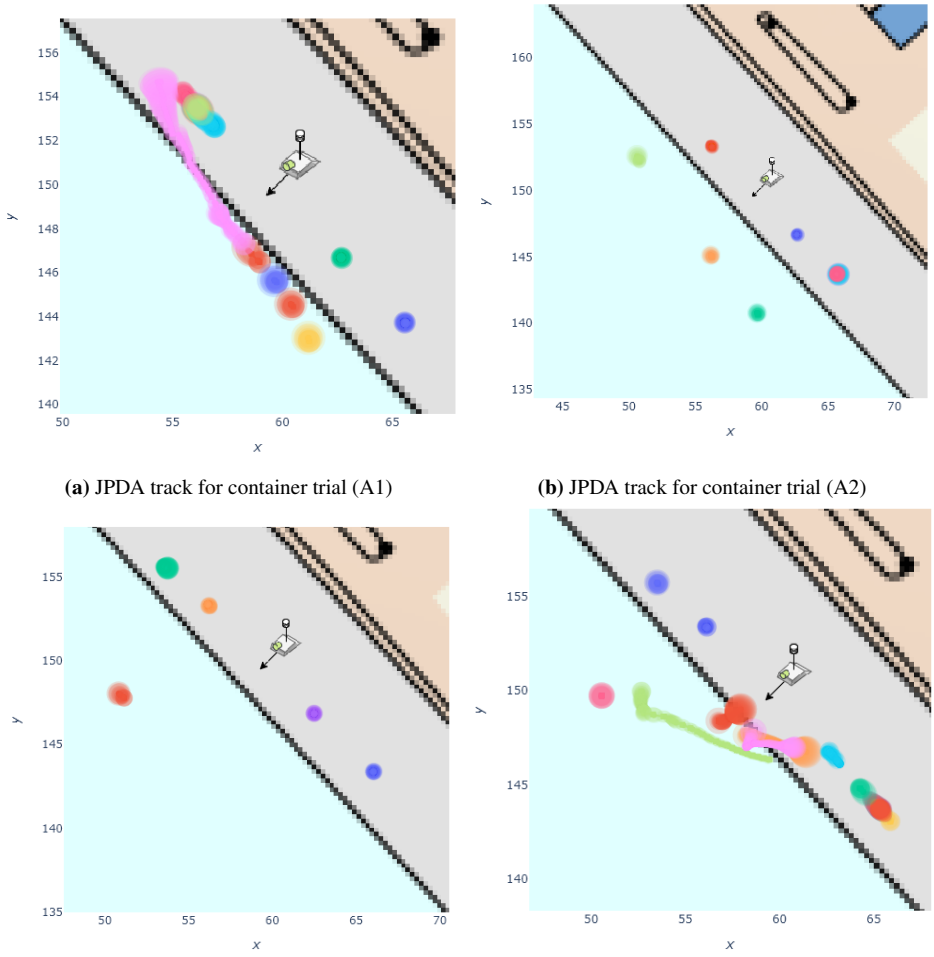
(a) Measurements for the container trial (A1)

(b) Measurements for the container trial (A2)

(c) Measurements for the container trial (A3)

(d) Measurements for the container trial (A4)

Figure 6.10: Plots of measurements for the container trials.



(a) JPDA track for container trial (A1)

(b) JPDA track for container trial (A2)

(c) JPDA track for container trial (A3)

(d) JPDA track for container trial (A4)

Figure 6.11: Plots of JPDA tracks for the container trials.

7

Discussion

This chapter presents and discusses the main findings of this thesis. Section 7.1 discusses the results related to the dataset D1. Section 7.2 evaluates the viability of the georeferencing method and its shortcomings based on dataset D2. Finally, Section 7.3 presents and discusses the LiDAR accuracy and tracking performance of the JPDA based on dataset D3.

7.1 Preliminary data - D1

7.1.1 LiDAR detection

The visual detection test gives valuable insight into using a LiDAR for detection of small objects in the water. First of all, all types of trash introduced were observed by the LiDAR when within the pre-determined 10 meter radius. This, combined with the fact that the lowest detection rate was 60% suggests that using a LiDAR for trash detection has the potential to yield correct measurements. Another observation is that the false negatives has a tendency to arrive consecutively, suggesting a systemic error. The most likely reason for this result is the issues related to low vertical resolution, first discussed in Section 3.3.1. The extreme case of 15 consecutive negatives in the case of the clear plastic container requires closer inspection, as there could be several contributing factors. The major distinction between this object and the other trash is that it is transparent. This means that the reflective properties also are different, and it is plausible that the laser might scatter at the right angles of reflection instead of being directed back towards the LiDAR. This is supported by the fact that the trash that should have the highest reflection coefficients, the

green and white beaker and white plastic bag, has the best detection rates. A caveat is that the during testing of the beaker the Otter was mostly turning, having very little translation. This can have led to an over-representation of the LiDARs detection rate at a specific range, yielding positively skewed results. This is likely not the case for the clear plastic, which data was recorded in an instance where the USV was moving continuously.

In summary the visual detection results show that the types of trash introduced can be detected in the chosen range. They do however reveal some of the weaknesses of using LiDAR, both because of missed detections and the inability to determine what the measurements stems from.

7.1.2 Full system test

The full scale system test shown in Figure 6.1 yields promising results. Starting with a general assessment of the systems detections throughout the run there are two observations that immediately stands out. The first is that the system has created tracks on the water surface, where several also lasted over extended periods of time. This suggests that the system manages to detect and track something over time, but the classification of what is being tracked cannot be determined directly, due to the lack of a controlled environment. Using the video-stream and corresponding detections as a baseline for what is floating in the area does however point to several of the tracks correctly originating from trash. This is especially evident in another snapshot of the process, shown in Figure 7.1. The camera frame shows an object being considered, a piece of styrofoam, directly in front of the USV. Starting with the detection marked by a blue bounding box in the image it is clear that the corresponding measurement marked in red is appropriately placed as the USV is moving south-east. The position of the detected trash is further collaborated through the filtered LiDAR measurements in the top left. Here the cluster straight ahead of the USV (x = 0m, y = 5 m) overlaps with the camera measurement when transformed to NED coordinates. Looking closer at the NED plot also reveals that no measurements from land have been passed on to the JPDA filter. This ties in with the second observation that clearly stands out in the JPDA-track plot. No measurements from the surroundings have been converted into tracks, with the sole exception of the track at approximately (x = 60, y = 90). Combining this with the fact that measurements from obstacles such as harbour walls were in fact observed and converted into line segments, as shown in Figure 6.2, points to the fact that the LiDAR filtering process worked as intended. In the same figure it is also clear that the LiDAR has observed two objects that are not currently visible through the corresponding image. Forwarding the video stream reveals thata cluster of seaweed is the likely candidate for the detections at around (x = -2.5, y = 7.5). The cluster at approximately (x = 3, y =

4) can however not be attributed to anything, either originating from a false positive or already being out of frame.

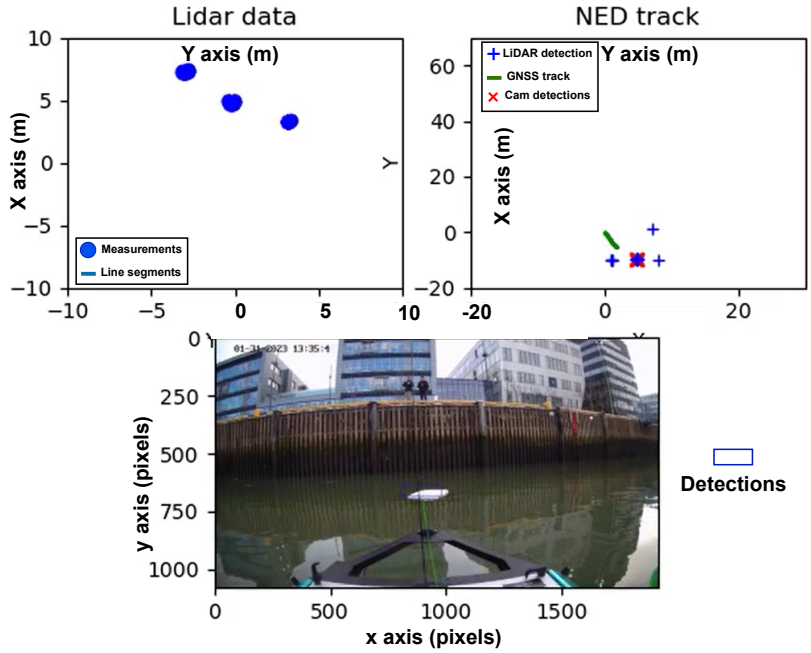


Figure 7.1: A snapshot showing the detection of a piece of trash and corresponding measurements in NED.

Zooming in on the tracks in the global test, shown in Figure 6.3 gives some other valuable insights. Figure 6.3a shows a closer view at some of the tracks in the vicinity of the harbour wall. The first thing of note is the amount of tracks being created, with several of them persisting for an extended amount of time. This points to the measurements being consistent and close enough in proximity both in time and displacement to be viable for the creation of tracks. It also shows that the filter manages to associate measurements to several tracks in the same time period. It is also clear that tracks are not allowed to deviate too much from its original position before being removed. This fits well with the fact that the trash is close to static and that it falls outside range as the USV is moving beyond the area.

Figure 6.3b shows a situation where two tracks are deleted when the allotted amount of timesteps without a new measurement has passed. As the covariance was low right before track loss, possibly due to an overconfidence in the measurements, the tracks were allowed to drift for an extended amount of time before being removed. Depending on the situation

and use-case this might be desirable. In a scenario where the trash is slowly drifting with a ocean current the CV-model might be relied upon enough to predict the position for an extended time. This would allow for the track to be picked up when the USV is back in range. On the other hand the focus on re-establishing tracks might be misguided. The track loss is often due to the USV moving away from the object in question, also meaning that the USV is unlikely to try to collect the trash in the immediate future. Reestablishing a new track when the USV is facing the object, i.e. if the USV is following a coverage path, might be a equally good solution. The tracks themselves are also interesting. Both are covering relatively large distances in the context of floating trash, and it seems likely that this originates from a systemic error rather than abnormally fast-floating objects. In both cases there are at least two perceivable ”jumps” where the filter has joined together measurements after a period of track loss. A plausible explanation could therefore be that measurements from several objects in close proximity have been attributed to a single track. Another could be that some GNSS measurements from the USV were lost or corrupted, leading to the USV being perceived as at rest for a short period. The LiDAR and camera inherently account for the USV displacement, but to relate relative measurements to the global map the USV fix has to be accurate.

Figure 6.3c show a comparison of a track being deleted by the time based deleter and a track being deleted by the covariance based deleter. The latter, illustrated in pink, shows that the covariance of the track expands until reaching a diameter of 1.5 m, consequently being removed. The reason for this large uncertainity could be that the track was only barely let through the initialization filter. The initial point of the track is inserted with a plausible uncertainty, set through the initialization filters measurement model. If no subsequent measurements were associated the covariance scales rapidly.

Figure 6.3d shows a track manually confirmed to consist of measurements from both camera and LiDAR. At the initial point at around (x = 66.75, y = 20.75) the measurements stems from the camera. As the USV closes in on the object (the styrofoam piece shown in Figure 7.1), the georeferencing first undershoots the distance before converging in the area where LiDAR measurements starts appearing. The track is then lost and the time based deleter removes the track. Two findings can be extracted from this. The first is that the georeferencing can be sensitive to the distance from the object. The second is that the JPDA manages to combine measurements of one object from both sensors into a single track.

From the results of the full system test it is evident that the system developed is capable of detecting small objects on the water surface, while filtering out disturbances and measurements from the surroundings. However, due to testing in a relatively uncontrolled

environment and lacking ground truth, the accuracy of measurements and tracks cannot be determined beyond visual confirmation. To further prove the viability of both sensors and the JPDA in the detection and tracking of floating trash the datasets D2 and D3 and corresponding results were acquired.

7.2 Georeferencing validation set - D2

7.2.1 Calibration

The calibration process for this camera was not trivial. Due to the fact that the measurements from the camera would effect the performance downstream, large efforts were made to ensure that the camera performed to its full potential. After observing the uncropped image from the pinhole model (Figure 5.2c) several steps were tried to rectify the situation. First, different subsets of images were used for calibration to see if the model was overfit in any way. This can happen if the checkerboards are too similar in pose and position, leading to the distortion in an small area being extrapolated to the full image. This approach did not result in any significant improvement. The wrapped image could be rectified by setting the k_3-parameter in the distortion coefficients to zero, but this left the image close to the original. The k_3-parameter is the fourth term in the radial distortion model used in the openCV model, given by $x_{distorted} = x(1 + k1r2 + k2r4 + k3r6)$ (OpenCV (2014)). A large value suggests that the radial distortion along the image edges is high. Visually it is clear that there is some distortion along the edges, but the wrapping effect suggests that this parameter was overestimated. The next approach was therefore to collect a new calibration set to ensure that the results were not due to a bad sample. Another set of 150 checkerboard images were captured in a different setting, with a more representative sample along the edge of the image. Each image was manually screened to make sure that the checkerboard was properly detected. The calibration was then run on both the full set and new subsets. This yielded similar results as the first calibration attempts, again with a large k_3-parameter. Finally, calibrating using the fisheye model was tried. The resulting calibration shown in Figure 5.2d showed promise in the way that it managed to undistort the image without the need for cropping. The drawback is that black borders are introduced, which could throw off the calculated distances for objects that are close to the bottom of the image. Cropping the image to remove the black borders yields the same situation as in the pinhole-calibration case, where Box 3 disappears.

7.2.2 Results

Beginning with a general comparison of the different images used to georeference (Figure 6.4) there are several observations that are of interest. One of the most obvious might be that the cropping effect of the pinhole model loses information in the image, by removing Box 3 and its corresponding bottle completely. Calculating the distance the camera should be able to see horizontally at a distance of 2.18 meters yields $tan(54.5) * 2.18 = 3.06m$. This means that the bottle should be visible in the frame. At the same time both the fisheye model and uncalibrated camera yielded inaccurate distances for Box 3, with errors in the range of 28 % - 53 % from ground truth. The choice of which camera calibration is best for the purpose of georeferencing therefore also depends on the weight put on accuracy versus false-negatives. Another aspect that is visually apparent is that the fisheye calibration "squeezes" the objects towards the image centre. Every bounding box coordinate is closer to the centre than the corresponding box in the other image instances. This introduces a significant error in the calculated x-distance for Box 4 which is not present in the other images. This, coupled with a much higher RMSE at 1.73 meters and overall worse accuracy, points to the fisheye calibration being a bad choice in the case of georeferencing. Although appearing to undistort the image in a satisfactory manner, it does conflict with the pinhole model used as a foundation for the georeferencing method. Better results could possibly have been achieved by creating a model based on image formation in a fisheye lens. However, as the georeferencing module was only one module of many in the system this was left as future work.

Comparing the results from the pinhole calibration with the results from the distorted image also yields interesting observations. The first thing to note is that the errors are remarkably similar overall. This is surprising considering the cropping of the pinhole images, which has the effect that the pixels need to increase in size to still take up the full resolution of 2688x1520. It is possible that most of the upsized pixels are found along the vertical edges of the image, where the cropping seems most severe when comparing the first four images in Figure 6.4. This theory is also supported by the fact that the white cupboard on the left is stretched more compared to the bounding boxes (Chosen as reference as it shares a similar vertical size to Box 4). The vertical stretch of the cupboard was measured to be about 35 pixels, while none of the bounding boxes experienced stretching above 8 pixels.

Another interesting result is that Box 1 and Box 6 had an estimated x-component that ended up being closer to the average of the two corresponding ground truth values (6.47 m). In other words both methods overestimated the position of Box 6 and underestimated Box 1. The most likely explanation for this is that the camera has an unaccounted error in

roll around the x-axis. This would lead to object on one side being overestimated and one being underestimated. Looking at the rest of the boxes one can see that Box 5 (right side) also is smaller than ground truth, while Box 3 and 1 (left side) are larger. Introducing a 2° rotation along the x-axis, tuned so that the x-component of Box 1 has a 1 cm offset in its x-component, yields a new RMSE error of 0.37 for the uncalibrated scenario and 0.20 for the pinhole calibrated method. Furthermore the x-component of Box 6 decreased as expected, with new values at 6.75 and 6.71 meters. This suggests that this alignment issue was a large contributor to the overall error.

7.2.3 Sensitivity

Expanding on the fact that an unaccounted 2° rotation could cause large discrepancies, the sensitivity test in Table 6.6 and Table 6.6 shows how sensitive the setup is to noise in general. This is especially true for offsets in the orientation of the camera, with the largest displacement being 1.18 m along the x-axis. Only the vertical offsets were used, as although both x and y distance scales with tan(θ), the angles are restricted to different subsets of the set A $\in$ (-90°,90°). An object in the bottom of the image corresponds to a vertical angle of $\frac{FOV_v}{2} = 30°$ and at the horizon to 90°, yielding the subset B = [30°,90°]. The horizontal angle is constrained by $\frac{FOV_h}{2}$ which yields a subset C = [-54.5°,54.5°]

7.2.4 Measurement model considerations

The shortcomings of the system presented in this section has direct implications for the viability as a sensor in the system. Although the RMSE for the uncalibrated and pinhole alternatives can be deemed acceptable for the purpose of locating the trash on the water surface, assumptions that were made during implementation should be revisited. The decision to treat the noise of the georeferencing in the CV-model as uniformly distributed is tenuous at best. However, due to the limited detection range before the performance starts to degrade it might still have merit. One way to counteract the impact of a bad measurement model could be to increase the uncertainty in only the x-component. This would lead the underlying Kalman filter to put less weight on that specific part of the measurement when iterating.

The sensitivity issue is also important to be aware of, as the USV is subject to wind and waves which can disturb the pose. To counteract this there are several methods that could be considered. The first is integrating an Inertial Measurement Unit (IMU) in the system, which allows for continuous estimation and correction of pose. However, how accurate such a system would have to be is also a consideration. With a 1° error potentially leading

to errors in the magnitude of meters the added complexity and cost might not be worth the effort. The second clear candidate is to increase the height of the camera. During testing the camera was 70.5 cm above ground looking at objects up to 6.75 m in front. This yields a max angle of $atan(\frac{6.75}{0.705}) = 84.0°$, where a $+1°$ shift gives a new distance of $tan(85°) * 0.705 = 8.06m$. Doubling the height yields $atan(\frac{6.75}{1.41}) = 78.2°$, where a $+1°$ shift gives $tan(79.2°) * 1.41 = 7.39m$. It is clear that just by raising the camera the sensitivity decreases. A trade-off between sensitivity to noise and the practicality of mounting the camera higher can therefore be considered. Another aspect of sensitivity that should be considered is the static error that can be introduced easily by aligning the camera and USV improperly. Even in a controlled environment, with a solid surface as baseline, the alignment effort proved to be inaccurate enough to introduce significant error.

To summarize, the georeferencing model and setup used in this thesis might be too sensitive for the application. When aligned properly and under calm conditions the method might be viable, but outside this scope the resulting measurements might ending up degrading overall system performance. The method was originally used for the tracking of marine vessels from land, which allowed both for the camera to be installed higher and the pose to be known exactly. One way to leverage the position information extracted could be to only use it attribute classification of of objects to the JPDA tracks. This would cover one of the main weaknesses of using the LiDAR, namely the inability to segment the detected objects, while also circumventing the introduction of uncertain measurements in the filter.

7.3 Track validation set - D3

7.3.1 D3 test setup

During processing of the dataset D3 it became clear that there were two problems with the experimental setup that led to issues with the test results. The issues were severe enough that they could not be rectified without a complete recapture of the dataset, which was not possible due to both limited access to the test equipment and time-constraints. This impacted both the accuracy of the sensor measurements as well as the ground truth used to validate the system performance. The first and most detrimental issue was the degradation of the ground truth tracks. Although returning a decent geographical fix when on land, which was inspected during testing, the ground truth measurements when the objects were throw in the water were few to non-existent. Closer inspection of the gps-data revealed that the periods where the drop-outs where the most prominent were the periods when the trash was being actively pulled by the fishing-rod. Observing the video-stream from one of the trials yields a plausible explanation. A frame from one of the trials with a bottle

is shown in Figure 7.2. It is clear that the phone attached to the bottle is at least partly submerged during the trial. This could potentially degrade the signal strength arriving at the receiver, as the signal has to travel through another medium than air.

Figure 7.2: Submerged GNSS logger (phone) during bottle trial

Furthermore, the ground truth measurements that were actually captured could not be compared to the sensor measurements or tracks in any meaningful way. Each ground truth measurement was compared to each tracks corresponding measurement in a ± 1 second time window. An upper limit of 5 m error between each component in the sensor measurement state vector and corresponding GNSS measurement was imposed. This would have set an upper limit on the RMSE and NEES-values, but was deemed an acceptable compromise as such a high value in practice means that the sensor was completely unable to track in the chosen radius of 10 m. Throughout all trials this yielded at most two matches, both with the corresponding RMSE at around 2 meters. There are several plausible explanations for this. It is likely that the delay issues experienced when aligning sensor measurements also extends to the difference between LiDAR capture time and stamping by the laptop. If the radio-connection had a delay over 1 second the sensor measurements would fall behind the corresponding ground truth track and introduce a systemic error. Accounting for this offset is also not trivial as it is originates from a wireless connection that can have variable delay. Another contributor could be inaccuracies in the mapping from the body frame to the NED frame. Deviations in heading or USV geodetic location could lead to bad alignment between tracks and ground truth. Both the USV position and ground truth was captured on .gpx format and could therefore not be RTK corrected. The horizontal accuracy of normal GNSS is usually in meters, which could make the offsets

even larger. Overall the data degradation led to a decision to leave out the metrics based on ground truth, namely NEES, ANEES and RMSE.

The second issue was the impact the tide had on the experiments. Throughout the testing the water-level was steadily decreasing, requiring constant monitoring. Furthermore, submerged concrete steps were revealed and the USV had to be relocated closer to the surface at two occasions. In addition to introducing a source of error on the georeferenecing as the height varied, it also heavily impacted the final trials where the aluminium box was introduced. Here partly submerged rocks along the harbour wall were so prominent that they overshadowed the object being tested. This was further exacerbated by the fact that the buoyancy of the box was overcome by the weight of the waterproofed phone, leaving the combination mostly submerged. This trial was therefore removed from the results. The trial do however point to one important consideration, namely that the system has no way of filtering out static objects right on the water surface that are small enough to not be caught by the line segment filtering. Trying to catch such an object could in the worst case lead to the USV being damanged or stuck.

7.3.2 Evaluation metrics

Starting with the evaluation metrics presented in Table 6.8, Table 6.9 and Table 6.10 the first thing of note is that almost all tracks exists for an extended period of time. The two notable exceptions being for trial A2 and A3 for the plastic container, where no proper track can be attributed. This could tie in to the reflective properties discussed in Section 7.1.1. It is also possible that the attached phone again reduced the footprint of the container by weighing it down. The track fragmentation rate is low overall, pointing towards the system being able to establish and keep track of the objects to a satisfactory degree. The outlier is Figure 6.11a which was tracked by 5 distinct tracks. The corresponding measurements in Figure 6.10a shows that the measurements in the beginning of the track comes in small clusters with distinct jumps between them. It is therefore likely that the bare minimum of measurements needed to establish a track has been achieved, but that the velocity could not be determined accurately enough to connect the tracks. A possible solution to this could be to increase the threshold of measurements needed to establish a track. This might however degrade the performance in other instances. The track probability of detection (TPD) is positively skewed in all trials. For the bottle trials this likely due to a mix of track fragmentation and a persistent false positive at around (x = 55, y = 157), i.e. seen in Figure 6.7c. For the bag trials the positive skew seems to stem from purely track fragmentation, possibly with the exception of Figure 6.9a where a misaligned camera measurement as seen in Figure 6.8a. For the container trials the high positive skew seems

to again be a combination of false positives and track fragmentation. The semi-submerged rocks discussed in Section 7.3.1 could be a potential reason for the former.

Looking at filter consistency all the ANIS values are within the confidence interval, although closer to the lower limit. This suggests that the predicted measurement error is of the same magnitude as the innovation covariance, with the measurements leaning towards being somewhat underconfident. This is likely an artefact of the tuning process using the first trial of the bottle test (A1). It was found that better results could be achieved by setting the uncertainty in the measurement model for both sensors relatively high. The underconfidence is supported by the NIS plots shown in Figure 6.5. It is evident that the NIS over time is consistently closer the lower limit, especially in the case of the Bag (A1) trial. The most interesting part of the plots is that they all have a sharp negative spike. This suggests that either the measurement residual was small or the innovation covariance large.

The ANIS and NIS values can also be seen in connection with the chosen transition model, which was tuned to be very conservative with regards to acceleration. As the measurements from the sensor were subject to a wide array of sources of error two subsequent measurements from the same object could end up with a significant distance between them. Trusting these measurements too much would yield a high velocity vector and the track would start drifting. By having a higher confidence in the underlying CV-model this effect is reduced. As no NEES could be calculated it is not possible to determine quantitatively if the transition model is consistent. However, looking at a track where track loss is present such as Figure 6.7b might give an indication. The velocity right before track loss has been determined by a significant sample of measurements. From the moment of track loss the covariance starts to increase, but not fast enough to be caught by the time-based deleter. This suggests that the transition model does not add a lot of uncertainty, pointing towards it being confident.

7.3.3 Measurement and JPDA plots

Delving into the measurements and corresponding JPDA tracks in Figure 6.6 - Figure 6.11 gives a clearer picture of the performance of both the individual sensors and the JPDA filter. Figure 6.6a and the corresponding JPDA track in Figure 6.7a presents a situation where the object being tracked most likely fell within the ”dead-zone” of the system. This is evident by the lack of measurements in the middle of the track. The tracker does however manage to re-establish the track when new measurements are apparent. This points to the filter being robust in the presence of varying amount of measurements available, at least when the movement is uniform. The measurement plot also shows that the sensors

agree on the position of the object. Figure 6.6b and Figure 6.7b presents a case where the two sensors compliment each other well. In the areas where LiDAR measurements are not present the camera ”fills in” the gap, resulting in a singular track. This showcases the strength of fusing measurements from different sensors to improve performance. Figure 6.6c and Figure 6.7c shows a case where the LiDAR is unable to see the object except for in the beginning, while the camera produces a coherent track on its own. The initialization is likely to have stemmed from the LiDAR measurements as the covariance in the bottom right of the JPDA plot is increasing, until being connected to the camera measurements. Figure 6.6d and Figure 6.7d shows that in an environment with several objects to be tracked the filter manages to create and hold tracks over time. However, some of the weaknesses of the JPDA also become apparent. The path the bottle was pulled, observed in the video stream, suggests that the pink, light blue and parts of the dark blue track actually stems form the same object. the fact that the dark blue track changes direction and starts traveling towards the USV implies that the track has been confused with one of the other bottles. It is also plausible that the end of the dark blue track is followed by the beginning of the lime-green track. In situations where the correct attribution of tracks is critical, such as in the tracking of aerial vehicles, track confusion is an important factor. It can be argued to be of less importance in a trash collection setting, where every detected object is to be collected and the order is determined by a higher-level heuristic.

Figure 6.8a and Figure 6.9a presents a case where the LiDAR is the only sensor contributing to the track. The only camera measurement present is also offset from the LiDAR measurements. A plausible explanation for the offset could be that the heigth above the surface was inaccurate due to the tide. Figure 6.8b andFigure 6.8c shows the opposite scenario, where camera measurements are the ones relied upon to establish tracks. The offset between camera and LiDAR is also present in and Figure 6.9b, which leads to the track first heading north-east before aligning with the actual north-west direction. Figure 6.8d and Figure 6.9d, where the yellow track corresponds to the bag tied to the fishing rod. Firstly, it is clear that track confusion has not taken place. This might be due to the velocity vectors for the yellow and blue track pointing in different directions. It could also be due to timing, with one of the tracks passing the intersection before the other. Another interesting observation is that the filter manages to catch even abrupt changes in direction.

Continuing with the container measurements and tracks the results are worse overall. Figure 6.10b and Figure 6.10c shows two situations where the sensors are not able to detect the object properly, leading to no decent tracks being established in Figure 6.11b and Figure 6.11c. Also in the case of Figure 6.10a the measurements have larger discontinuities between them, reflecting in the 5 tracks established in Figure 6.11a to track one

object. Finally Figure 6.10d shows that three objects are likely to have been tracked, albeit with some errors. This is evident in Figure 6.11d, where the lime-green and dark-pink is likely to originate from the container, while the red and pink/purple represent two different objects. The overlapping of the pink and yellow track could be due to a synchronization offset, where measurements that were supposed to come together were offset in time leading to two track initializations.

In total 10 out of the total 12 trials yielded tracks that could be correctly attributed to the introduced object. From the results obtained it is evident that all main components of the system have merit in the detection and tracking of floating trash. In optimal conditions the sensors have the ability to determine the presence of floating trash, and there are several scenarios where the multi-sensor fusion approach contributes to better tracking performance.

8

Conclusion and future work

8.1 Conclusion

The main goals of this thesis was to explore the trash detection and tracking problem, establish a framework that could relate measurements from a viable sensor package to a moving USV, and finally develop and test a trash detection and tracking system based on the findings. The literature review found that although several solutions for trash collecting USV have been proposed, few have the capabilities for fully autonomous operation. Furthermore, the combination of LiDAR and camera to detect floating trash have barely been explored, and the use of a tracking filter to relate measurements in time has not been tried to improve upon the proposed solutions. Using an established georeferencing method, a tailored approach to the filtering of LiDAR data and a JPDA filter, a system to track and detect trash was developed for an USV. Three real-life datasets were captured by the USV in question and used to establish the viability of both the individual system and the specific sensors. The overarching goal of detecting and tracking trash was achieved, with tracks of trash being properly attributed to actual objects on the water surface in a semi-controlled environment. A test in an uncontrolled environment at Brattørkaia proved that the system was capable of detecting objects on the water surface and placing them at a reasonable geographic location while the USV was moving. Due to a lack of ground-truth the precision of the system could not be determined quantitatively, and the datasets only represent a small subset of weather, light and sea conditions. Further experiments is therefore needed to validate the precision and performance in different settings. Several shortcomings of the sensor-package were identified, the most critical being the sensitivity of the georeferencing method to offsets in pose and orientation. The viability of a monoc-

ular camera for distance estimation in this setting can therefore be drawn into question. An alternative could be to use the camera for classification, tagging corresponding tracks using the calculated distance. The LiDAR sensor did not suffer the same sensitivity problems, but is not able to determine what the measurements stems from. Relying purely on LiDAR is therefore not viable with the developed system, as the USV might try to collect semi-submerged obstacles or animals on the surface.

8.2 Future work

This section presents some of the areas where one could build or improve upon the thesis. Firstly, some of the main shortcomings of the current data-collection system and what could be done to improve them are discussed. Some computational bottlenecks in the current python implementation are then identified and potential optimizations are suggested. Finally, a road-map for integrating the implemented system in a fully autonomous trash collecting USV is drawn up.

Full ROS integration

When looking at the datasets captured and used in this thesis two things are immediately apparent. Firstly, large amounts of effort had to be undertaken to format, synchronize and pipeline information from different sensors into a joint framework. Secondly, even after the measurements were unified problems with synchronization persisted. All of the issues can be significantly improved by changing a single system design decision, namely by running data collection through a roscore local to the USV. The roscore is the central node which all other nodes attaches to for ROS to function. This has three clear advantages. The first is that the camera stream can be integrated straight in to the system. The current solution has the camera publishing to a ip-address which then has to be transmitted and read before any processing can take place. This introduces significant, variable delay that is hard to measure without meta-data. The second and most important advantage extends on this in the way that all the measurements can be stamped as they arrive in the roscore, instead of having the measurements being stamped according to a remote desktop, satellites in orbit and the LiDAR itself. Some delays might still have to be accounted for due to internal transmission delay, but this should be more predictable than the wireless communication setup. The second improvement is that the system no longer will require a stable, high-speed radio connection to the computer during operation. As of now, if the trash detection system is to be integrated in a fully autonomous USV, the processing has

to be handled remotely and way-points sent back to the USV. This leaves it susceptible to drop outs in communication due to range limitations and environmental obstructions.

Runtime optimization

Real-time performance was an early ambition in this thesis. It was however deemed necessary to put the ambition aside and focus on proving and validating the concept of fusing LiDAR and camera measurements to detect and track trash. Throughout the work done there were however several options that were identified as potential improvements. One of the clearest candidates is optimizing the LiDAR intersection step, which runs an exhaustive search comparing line segments and measurements, yielding a O($m * n$) time complexity, where m is the amount of measurements and n is the amount of lines. Although some improvements were made, such as dynamically removing measurements and redundant lines, the process scales poorly. An improvement would be to implement the line-sweep algorithm first described in Bentley and Ottmann (1979), having a O(nlog(m)) runtime in this specific application.

The next improvement that has clear potential is parallelization. The largest (consistent) bottleneck identified is the YOLOv7 model running on every image, taking up to two seconds per frame. This processing could be done in parallel if a Graphics Processing Unit (GPU) was a added. This is likely to both reduce the processing time per frame and free up compute resources. The first is crucial as four images arrives every second. The latter is equally important as a delay might lead to loss of measurements or a backlog of measurement needing to be processed, both degrading performance. It is also possible that the LiDAR processing could benefit from being transitioned to a GPU.

Finally, rewriting the code in a faster language such as C++ (Alomari et al. (2015)) can also be considered. If implemented properly it could improve the run-time of the entire pipeline of processes. A caveat is that the packages used to offload tasks in python might not have an equivalent in C++, leading to development overhead to implement the necessary components. It might therefore be considered a last resort if the other optimization steps outlined fails to achieve real-time capabilities.

Fully autonomous USV

The logical next step for the system developed in this thesis is to integrate it in a overarching system capable of full autonomy in the detection, collection and deposit of floating trash. A draft of how such a system could be constructed is shown in Figure 8.1. The guidance system, identified by yellow boxes, are responsible for the overall positioning of the

USV. The user selects a geographic area that is to be cleaned and the global path planner creates a optimized path that covers the area, feeding a set of waypoints to the system. This is only necessary to do in the initial setup. The rest of the system runs continuously, with the local path planner deciding on an optimal path based on the current state of the USV, the amount of trash that can be collected, obstacles that are present and deviation from the global path. The heading and speed needed to follow the optimal path is passed on to a inner control loop which calculates and returns the needed control inputs to achieve the state using the available actuators. Sensor measurements from the USV is then propagated back into the sensor processors and the next iteration begins.

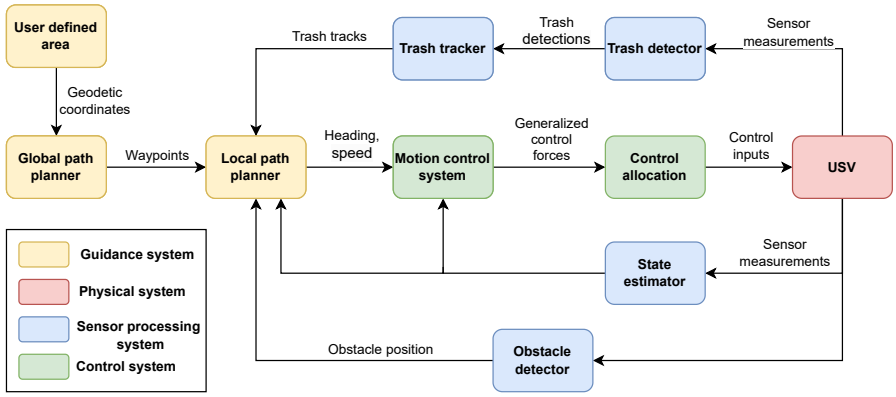


Figure 8.1: Draft of a system for a fully autonomous USV, only requiring a operator to select the area to be cleaned.

Bibliography

Akib, A., Tasnim, F., Biswas, D., Hashem, M.B., Rahman, K., Bhattacharjee, A., Fattah, S.A., 2019. Unmanned floating waste collecting robot, in: TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), pp. 2645–2650. doi:10.1109/TENCON.2019.8929537.

Alomari, Z., Halimi, O., Sivaprasad, K., Pandit, C., 2015. Comparative studies of six programming languages .

Antonio, F., 1992. Iv.6 - faster line segment intersection, in: KIRK, D. (Ed.), Graphics Gems III (IBM Version). Morgan Kaufmann, San Francisco, pp. 199–202. URL: https://www.sciencedirect.com/science/article/pii/B9780080507552500452, doi:https://doi.org/10.1016/B978-0-08-050755-2.50045-2.

Bentley, Ottmann, 1979. Algorithms for reporting and counting geometric intersections. IEEE Transactions on Computers C-28, 643–647. doi:10.1109/TC.1979.1675432.

Brekke, E., 2020. Fundementals of Sensor Fusion. Third ed., NTNU. URL: https://www.ntnu.edu/employees/edmund.brekke.

Briot, S., Khalil, W., 2015. Homogeneous Transformation Matrix. Springer International Publishing, Cham. pp. 19–32. URL: https://doi.org/10.1007/978-3-319-19788-3_2, doi:10.1007/978-3-319-19788-3_2.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G., 2013. API design for machine learning software: experiences from

the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122.

Chang, H.C., Hsu, Y.L., Hung, S.S., Ou, G.R., Wu, J.R., Hsu, C., 2021. Autonomous water quality monitoring and water surface cleaning for unmanned surface vehicle. Sensors 21, 1102. URL: http://dx.doi.org/10.3390/s21041102, doi:10.3390/s21041102.

Cheng, Y., Xu, H., Liu, Y., 2021. Robust small object detection on the water surface through fusion of camera and millimeter wave radar, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 15243–15252. doi:10.1109/ICCV48922.2021.01498.

Clark, A., 2015. Pillow (pil fork) documentation. URL: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf.

Clunie, T., DeFilippo, M., Sacarny, M., Robinette, P., 2021. Development of a perception system for an autonomous surface vehicle using monocular camera, lidar, and marine radar, in: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 14112–14119. doi:10.1109/ICRA48506.2021.9561275.

Comaniciu, D., Meer, P., 2002. Mean shift: a robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 603–619. doi:10.1109/34.1000236.

Doyle, A., 2018. Plastic waste in antarctica reveals scale of global pollution: Greenpeace. URL: https://www.reuters.com/article/us-antarctica-plastics-idUSKCN1J22YW.

Field, T., Leibs, J., Bowman, J., D., T., Perron, J., 2023. rosbag. https://github.com/ros/ros_comm/tree/noetic-devel/tools/rosbag.

Forsyth, D.A., Ponce, J., 2012. Computer Vision - A Modern Approach, Second Edition. Pitman.

Fortmann, T., Bar-Shalom, Y., Scheffe, M., 1983. Sonar tracking of multiple targets using joint probabilistic data association. IEEE Journal of Oceanic Engineering 8, 173–184. doi:10.1109/JOE.1983.1145560.

Fossen, T.I., 2011. Kinematics. John Wiley and Sons, Ltd. chapter 2. pp. 15–44. URL: https://onlinelibrary.wiley.com/doi/abs/

10.1002/9781119994138.ch2, doi:https://doi.org/10.1002/9781119994138.ch2.

Gall, S., Thompson, R., 2015. The impact of debris on marine life. Marine Pollution Bulletin 92, 170–179. URL: https://www.sciencedirect.com/science/article/pii/S0025326X14008571, doi:https://doi.org/10.1016/j.marpolbul.2014.12.041.

Gao, X., Fu, X., 2020. Miniature water surface garbage cleaning robot, in: 2020 International Conference on Computer Engineering and Application (ICCEA), pp. 806–810. doi:10.1109/ICCEA50009.2020.00176.

Gibbens, S., 2021. Another plastic bag found at the bottom of world's deepest ocean trench. URL: https://www.nationalgeographic.com/science/article/plastic-bag-mariana-trench-pollution-science-spd.

Gladstone, R., Moshe, Y., Barel, A., Shenhav, E., 2016. Distance estimation for marine vehicles using a monocular video camera, in: 2016 24th European Signal Processing Conference (EUSIPCO), pp. 2405–2409. doi:10.1109/EUSIPCO.2016.7760680.

Haghbayan, M.H., Farahnakian, F., Poikonen, J., Laurinen, M., Nevalainen, P., Plosila, J., Heikkonen, J., 2018. An efficient multi-sensor fusion approach for object detection in maritime environments, in: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2163–2170. doi:10.1109/ITSC.2018.8569890.

Halterman, R., Bruch, M., 2010. Velodyne HDL-64E lidar for unmanned surface vehicle obstacle detection, in: Gerhart, G.R., Gage, D.W., Shoemaker, C.M. (Eds.), Unmanned Systems Technology XII, p. 76920D. doi:10.1117/12.850611.

Hammer, M., Hebel, M., Laurenzis, M., Arens, M., 2018. Lidar-based detection and tracking of small UAVs, in: Buller, G.S., Hollins, R.C., Lamb, R.A., Mueller, M. (Eds.), Emerging Imaging and Sensing Technologies for Security and Defence III; and Unmanned Sensors, Systems, and Countermeasures, International Society for Optics and Photonics. SPIE. p. 107990S. URL: https://doi.org/10.1117/12.2325702, doi:10.1117/12.2325702.

Han, J., Cho, Y., Kim, J., Kim, J., Son, N., Kim, S., 2020. Autonomous collision detection and avoidance for aragon usv: Development and field tests. Journal of Field Robotics 37. doi:10.1002/rob.21935.

Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585, 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2, doi:10.1038/s41586-020-2649-2.

Helgesen, H.H., Leira, F.S., Bryne, T.H., Albrektsen, S.M., Johansen, T.A., 2019. Real-time georeferencing of thermal images using small fixed-wing uavs in maritime environments. ISPRS Journal of Photogrammetry and Remote Sensing 154, 84–97. URL: https://www.sciencedirect.com/science/article/pii/S0924271619301315, doi:https://doi.org/10.1016/j.isprsjprs.2019.05.009.

Hermann, D., Galeazzi, R., Andersen, J., Blanke, M., 2015. Smart sensor based obstacle detection for high-speed unmanned surface vehicle. IFAC-PapersOnLine 48, 190–197. URL: https://www.sciencedirect.com/science/article/pii/S2405896315021709, doi:https://doi.org/10.1016/j.ifacol.2015.10.279. 10th IFAC Conference on Manoeuvring and Control of Marine Craft MCMC 2015.

Hikvision, . Ds-2cd2045fwd-i. URL: https://www.hikvision.com/en/products/IP-Products/Network-Cameras/Pro-Series-EasyIP-/DS-2CD2045FWD-I/.

Hiles, J., 2023. Stonesoup. URL: https://stonesoup.readthedocs.io/en/v0.1b11/index.html.

Hunter, J.D., 2007. Matplotlib: A 2d graphics environment. Computing in Science & Engineering 9, 90–95. doi:10.1109/MCSE.2007.55.

Inc., P.T., 2015. Collaborative data science. URL: https://plot.ly.

Jambeck, J.R., Geyer, R., Wilcox, C., Siegler, T.R., Perryman, M., Andrady, A., Narayan, R., Law, K.L., 2015. Plastic waste inputs from land into the ocean. Science 347, 768–771. URL: https://www.science.org/doi/abs/10.1126/science.1260352, doi:10.1126/science.1260352, arXiv:https://www.science.org/doi/pdf/10.1126/science.1260352.

Jeong, M., Li, A.Q., 2021. Efficient lidar-based in-water obstacle detection and segmentation by autonomous surface vehicles in aquatic environments, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5387–5394. doi:10.1109/IROS51168.2021.9636028.

Jiang, K., 2017. Calibrate fisheye lens using opencv - part 1. URL: https://medium.com/@kennethjiang/calibrate-fisheye-lens-using-opencv-333b05afa0b0.

Øystein Kaarstad Helgesen, Brekke, E.F., Stahl, A., Øystein Engelhardtsen, 2020. Low altitude georeferencing for imaging sensors in maritime tracking∗∗this work was supported by the research council of norway (nfr) through the projects 223254 and 244116/o70. IFAC-PapersOnLine 53, 14476–14481. URL: https://www.sciencedirect.com/science/article/pii/S2405896320318607, doi:https://doi.org/10.1016/j.ifacol.2020.12.1449. 21st IFAC World Congress.

Kannala, J., Brandt, S., 2006. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. IEEE transactions on pattern analysis and machine intelligence 28, 1335–40. doi:10.1109/TPAMI.2006.153.

Kartverket, . Få veiledning om cpos. URL: https://www.kartverket.no/til-lands/posisjon/hva-er-cpos.

Khawaja, S., Amjad, M., Zafar, H., Rauf, F., 2020. Remotely operated water surface cleaner. Pakistan Journal of Engineering and Technology 3, 131–136. URL: https://hpej.net/journals/pakjet/article/view/435, doi:10.51846/vol3iss2pp131-136.

Li, X., Tian, M., Kong, S., Wu, L., Yu, J., 2020. A modified yolov3 detection method for vision-based water surface garbage capture robot. International Journal of Advanced Robotic Systems 17, 1729881420932715. URL: https://doi.org/10.1177/1729881420932715, doi:10.1177/1729881420932715, arXiv:https://doi.org/10.1177/1729881420932715.

Mihelich, P., Bowman, J., 2023. vision_opencv. https://github.com/ros-perception/vision_opencv.

Morel, J.M., Randall, G., Jakubowicz, J., Grompone Von Gioi, R., 2012. Lsd: A line segment detector - researchgate.net. https://www.researchgate.net

URL: https://www.researchgate.net/publication/279348491_LSD_A_line_segment_detector.

Ocean_Cleanup, 2023. URL: https://theoceancleanup.com/.

OECD, 2022. Section 3: Plastics use projections to 2060. ORGANIZATION FOR ECONOMIC Co-operation and Development. URL: https://www.oecd-ilibrary.org/content/publication/aa1edf33-en, doi:https://doi.org/https://doi.org/10.1787/aa1edf33-en.

OpenCV, 2014. The opencv reference manual. URL: https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html.

OpenCV, 2015. Cv::linesegmentdetector class reference. URL: https://docs.opencv.org/3.4/db/d73/classcv_1_1LineSegmentDetector.html.

OpenRobotics, a. Robot operating system. URL: https://www.ros.org/.

OpenRobotics, b. video_stream_opencv. URL: http://wiki.ros.org/video_stream_opencv.

O'Quin, J., . velodyne. URL: https://wiki.ros.org/velodyne.

Otter, . Otter. URL: https://www.maritimerobotics.com/otter.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830. URL: https://scikit-learn.org/stable/modules/clustering.html#mean-shift.

Prata, J.C., da Costa, J.P., Lopes, I., Duarte, A.C., Rocha-Santos, T., 2020. Environmental exposure to microplastics: An overview on possible human health effects. Science of

The Total Environment 702, 134455. URL: https://www.sciencedirect.com/science/article/pii/S0048969719344468, doi:https://doi.org/10.1016/j.scitotenv.2019.134455.

Satheesh, J., Nair, A.P., M., D., A., C., Mahesh, G., Jayasree, P.R., 2020. Wireless communication based water surface cleaning boat, in: 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), pp. 716–720. doi:10.1109/ICOEI48184.2020.9142960.

Scivision, 2023. Pymap3d. https://github.com/geospace-code/pymap3d.

bar shalom, Y., Daum, F., 2010. The probabilistic data association filter. Control Systems, IEEE 29, 82 – 100. doi:10.1109/MCS.2009.934469.

Stanislas, L., Dunbabin, M., 2019. Multimodal sensor fusion for robust obstacle detection and classification in the maritime robotx challenge. IEEE Journal of Oceanic Engineering 44, 343–351. doi:10.1109/JOE.2018.2868488.

Tomar, S., 2006. Converting video formats with ffmpeg. Linux Journal 2006, 10.

Ublox, 2023. Ann-mb series. URL: https://www.u-blox.com/en/product/ann-mb-series.

Valgur, M., 2023. velodyne_decoder. https://github.com/valgur/velodyne_decoder.

Velodyne, . 63-9229 rev-h puck datasheet web - mapix technologies. URL: https://www.mapix.com/wp-content/uploads/2018/07/63-9229_Rev-H_Puck-_Datasheet_Web-1.pdf.

Vormdal, M., 2022. Embedded system for maze-mapping.

Vormdal, M., 2023. Ttk4900. https://github.com/marcusvormdal/TTK4900.

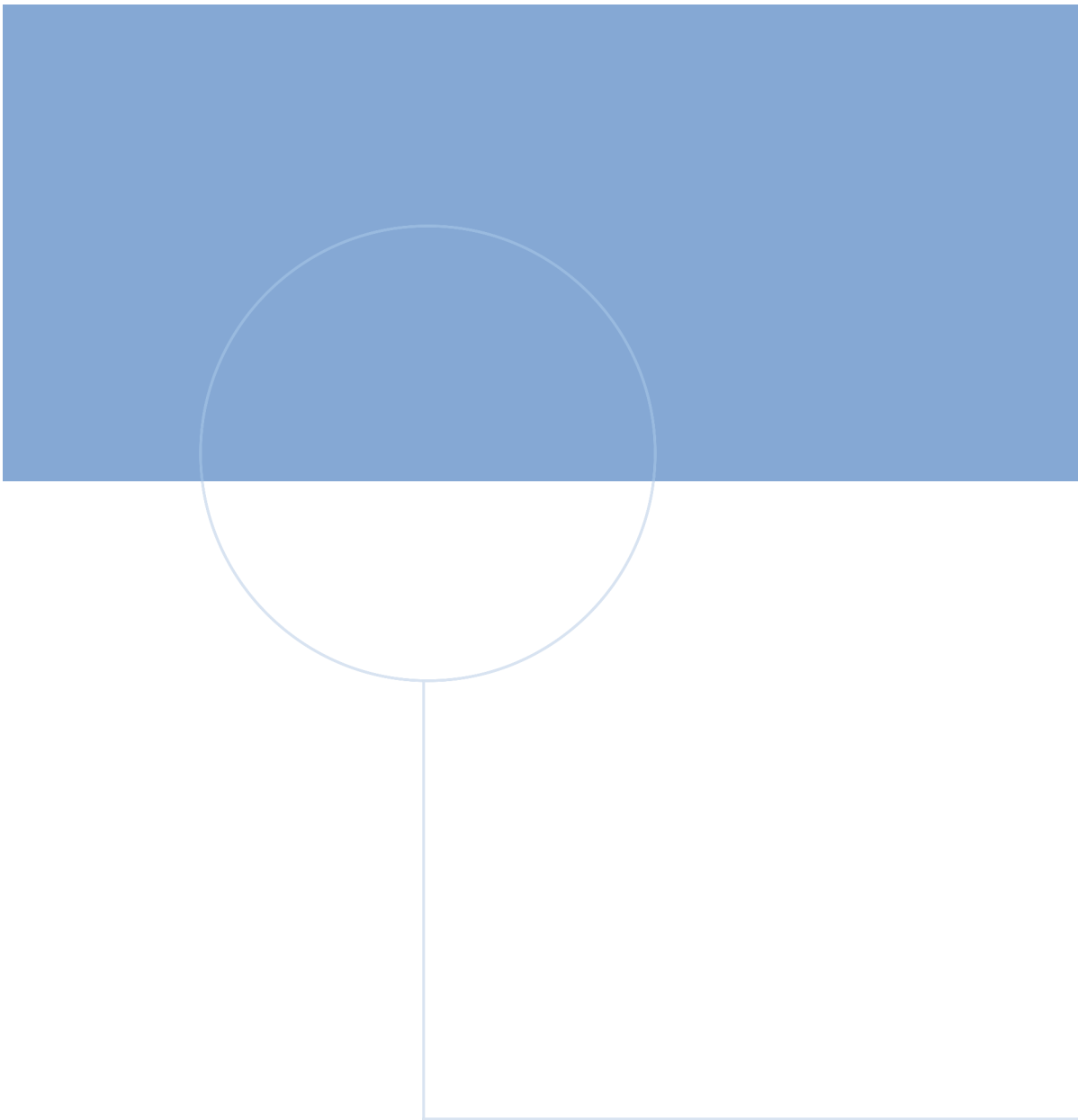
Woo, J., Kim, N., 2015. Vision-based target motion analysis and collision avoidance of unmanned surface vehicles. Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment 230. doi:10.1177/1475090215605136.

Zhang, M., Liu, Z., Cai, W., Yan, Q., 2021a. Design of low-cost unmanned surface vessel for water surface cleaning, in: 2021 China Automation Congress (CAC), pp. 2290–2293. doi:10.1109/CAC53003.2021.9728372.

Zhang, W., Jiang, F., Yang, C.F., Wang, Z.P., Zhao, T.J., 2021b. Research on unmanned surface vehicles environment perception based on the fusion of vision and lidar. IEEE Access 9, 63107–63121. doi:10.1109/ACCESS.2021.3057863.

Zhang, Z., 2000. A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 1330–1334. doi:10.1109/34.888718.

Zhu, J., Yang, Y., Cheng, Y., 2022. Smurf: A fully autonomous water surface cleaning robot with a novel coverage path planning method. Journal of Marine Science and Engineering 10. URL: https://www.mdpi.com/2077-1312/10/11/1620, doi:10.3390/jmse10111620.



NTNU

Norwegian University of
Science and Technology