FH VORARLBERG

UNIVERSITY OF APPLIED SCIENCES

Pose Estimation of Rope-guided Tool from Point Clouds

Algorithm Selection and Performance Analysis

Master's Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering, MSc

September, 2023

Submitted by
Thomas Rosenberger, BSc

Supervisors
Prof. (FH) Dipl.-Ing. Dr. techn. Franz GEIGER - FH Vorarlberg
Dr. Nicola Schlatter - Liebherr Werk Nenzing GmbH

Preface

Firstly, I want to thank the department "Vorentwicklung Assistenzsysteme" at Liebherr Werk Nenzing GmbH, especially Nicola Schlatter, for a welcoming experience in the workplace and for his continued support.

Secondly, thanks to all the staff of FHV, especially Johannes Steinschaden and the mechatronics course administration for always going the extra mile to show that they really care about their students. And to Franz Geiger, my thesis mentor, for his support his guidance, and his patience.

Another big thank you goes to my girlfriend, my friends and my family, who supported me in many ways during this time, not only by helping me reach the goal, but also by making the journey enjoyable.

Last but not least, everyone who helped me along the way who I have not mentioned here.

Thank you!

Abstract

Pose Estimation of Rope-guided Tool

The advent of autonomous and self-driving cranes represents a significant advancement in industrial automation. One critical prerequisites for achieving this long-term goal is the accurate and reliable detection of tools guided by ropes in real-world environments. Since the tool is suspended by ropes, the tool pose cannot be controlled directly. This master's thesis addresses the challenges of pose estimation for rope-guided tools using point cloud measurements.
The proposed algorithm utilizes constraints imposed by the crane kinematics and information extracted during the segmentation process to efficiently infer the pose of the hook, therefore enabling the use of the pose for decision making in real-time critical applications. RANSAC (Random Sample and Consensus) is deployed in the segmentation process to extract geometric primitives from the point cloud which represent the ropes and distinctive parts of the tool. Since the point cloud is often to sparse for feature matching a bounding box is used to estimate the initial position of the tool.
Two different methods are presented to improve the initial pose. A computationally expensive method with a high level of confidence, integrating the ICP (Iterative Closest Point) algorithm is used as a benchmark. A linear Kalman filter is used in the second method which is real-time capable. The benchmark is then used to evaluate the real-time capable approach.
The core contributions of this research lie in the innovative utilization of bounding boxes for pose estimation. The findings and methodologies presented herein constitute an advancement towards the realization of autonomous and self-driving cranes.

Keywords: *6DoF pose estimation, Point Cloud, Kalman Filter, RANSAC, ICP, Bounding Box*

Kurzreferat

Pose-Erfassung von seilgeführtem Werkzeug

Das Aufkommen von autonomen und selbstfahrenden Kränen stellt einen bedeutenden Fortschritt in der industriellen Automatisierung dar. Eine entscheidende Voraussetzung für das Erreichen dieses langfristigen Ziels ist die genaue und zuverlässige Erkennung von seilgeführten Werkzeugen in realen Umgebungen. Da das Werkzeug an Seilen aufgehängt ist, kann die Pose des Werkzeugs nicht direkt kontrolliert werden. Diese Masterarbeit befasst sich mit den Herausforderungen der Posenschätzung für seilgeführte Werkzeuge anhand von Punktwolkenmessungen.
Der vorgeschlagene Algorithmus nutzt die durch die Krankinematik auferlegten Einschränkungen und die während des Segmentierungsprozesses extrahierten Informationen, um die Pose des Hakens effizient abzuleiten und ermöglicht so die Verwendung der Pose zur Entscheidungsfindung in echtzeitkritischen Anwendungen. RANSAC (Random Sample and Consensus) wird im Segmentierungsprozess eingesetzt, um geometrische Primitive aus der Punktwolke zu extrahieren, die die Seile und markante Teile des Werkzeugs darstellen. Da die Punktwolke oft zu spärlich für Feature Matching ist, wird eine Bounding Box verwendet, um die initiale Pose des Werkzeugs zu schätzen.
Es werden zwei verschiedene Methoden zur Verbesserung der initialen Pose vorgestellt. Eine rechenintensive Methode mit hohem Vertrauensniveau, die den ICP-Algorithmus (Iterative Closest Point) integriert, wird als Benchmark verwendet. In der zweiten Methode wird ein linearer Kalman-Filter verwendet, der echtzeitfähig ist. Der Benchmark wird dann verwendet, um die Genauigkeit des echtzeitfähigen Ansatzes zu bewerten.
Der Hauptbeitrag dieser Forschung liegt in der innovativen Verwendung von Bounding Boxes für die Posenschätzung. Die hier vorgestellten Erkenntnisse und Methoden stellen einen Fortschritt auf dem Weg zur Realisierung von autonomen und selbstfahrenden Kränen dar.

Keywords: *6DoF-Pose-Erfassung, Point Cloud, Kalman-Filter, RANSAC, ICP, Bounding Box*

Contents

List of Figures

LHM	Liebherr Harbour Mobile Crane
ROS2	Robot Operating System 2
LiDAR	Light Detection and Ranging Sensor
AIT	Austrian Institute of Technology
ACU	Automation Control Unit
IPC	industrial PC
voxel	volumetric pixel
AABB	Axis-Aligned Bounding Box
OBB	Oriented Bounding Box
MVBB	Minimal Volume Bounding Box
PCA	Principal Component Analysis
RANSAC	Random Sample and Consensus
LMSE	Least Mean Squared Error
ICP	Iterative Closest Point

List of Algorithms

1 Introduction

Liebherr, a prominent manufacturer in the field of material handling machinery, builds and develops a diverse portfolio of equipment, including rope excavators, harbor cranes, and ship cranes. With advancements in automation technology on the horizon, it is believed that self driving cranes can be built and deployed in the near future, consequently automating material handling.

One major prerequisite for the automation of material handling is the accurate and reliable detection of the tool position and orientation. The tool and thus also the handled load are usually guided by rope(s). Due to the rope guidance, the exact position and orientation (pose) cannot be controlled directly. The pose is rather the result of the forces acting over the ropes, gravity and mass inertia.

Conventional sensor systems (e.g. IMU) can support the detection, but in many cases the placement of sensors on the tool is not possible. For this reason, optical sensors (Light Detection and Ranging Sensor (LiDAR) / stereo camera) will be used to capture the tool pose. The sensors provide a 3D point cloud and image data (grey-scale) of the measurement area, which includes or can include the tool, the possibly struck load and the environment. The aim is to harness the potential of optical sensors to accurately and reliably determine the tool pose and thus paving the way for the realization of autonomous material handling machinery.

1.1 Motivation

The material handling process is a highly dynamic process in which the machines perform a complete load cycle (loading and unloading) in a period of about 40-90 seconds. Driving of cranes is a very demanding task. The monotony and tedium of repetitive crane operations leads to fatigue and can induce human-based errors. This results in expensive damage, personnel injuries and fatalities. While human operators will always be required to make nuanced strategic decisions, rapid advances in robotics and AI-based technologies can assume many aspects of crane operations that are unattractive, risky and labour-intensive. This would enhance operational efficiency and relieve crane operators.

Right now, there are about 10 000 mobile cranes and 4 000 construction cranes deployed worldwide, which could be adapted with the sensors and controls to enable autonomous driving.

1.2 Prerequisites

This section gives an overview of the project prerequisites. It includes a short description of

1. **the crane**, for which the sensor data is acquired,
2. **the sensor setup**, which is a mechanical construction that holds the optical sensors, as well as other
3. **relevant electronic components**, both on the crane controller and on the workstation where
4. **the ROS2-Simulation**, is running and with which all the experiments in this thesis are made

1.2.1 The Liebherr Harbor Mobile Crane (LHM) 550

There are many different cranes and many different load suspension means for which the tool tracking could be used for. And while consideration is being taken to the possibility of different configurations of crane and load suspension means, this thesis focuses on one specific configuration, where a prototype is readily available as a test-setup. Namely, the Liebherr Harbor Mobile Crane (Liebherr Harbour Mobile Crane (LHM)) 550 with a hook attached as load suspension means. For understanding some of the requirements which come from the crane kinematics, the working principle of an LHM along with some technical terms are introduced here. As can be seen in Fig. 1.1, an LHM can be divided into five major parts:

1. Base
2. Tower
3. Boom
4. Ropes
5. Load Lifting Device (Hook)

and has four actuated joints:

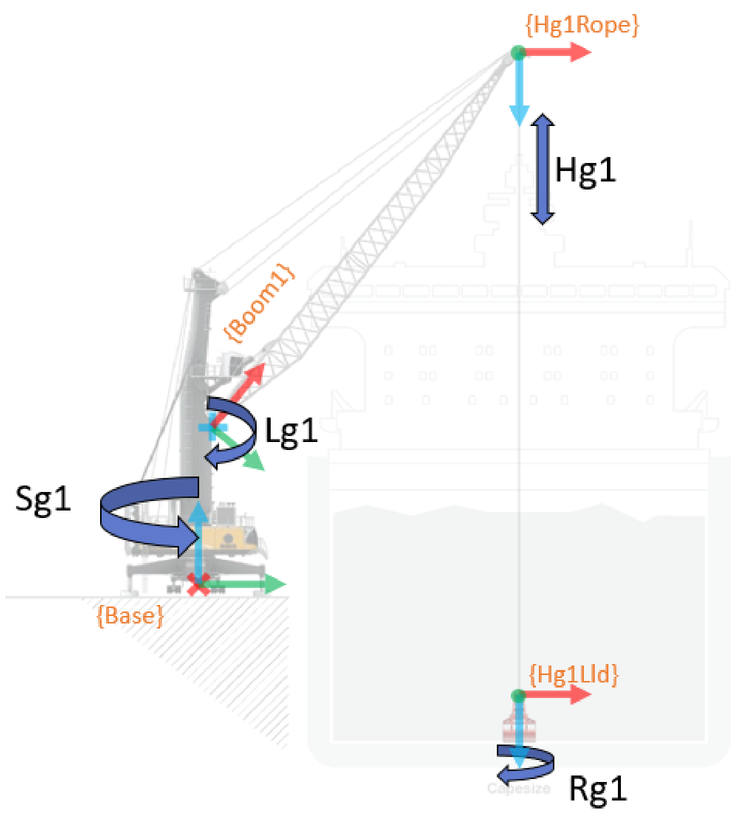


Figure 1.1: LHM 550 with the four actuated joints. Sg1: Slewing joint, Lg1: Luffing joint, Hg1: Hoisting joint, Rg1: Rotating joint
Source: own elaboration

1. slewing joint: rotation of the tower about the vertical axis
2. luffing joint: rotation of the boom about a horizontal axis
3. hoisting joint: length of the ropes on which the tool is suspended
4. rotating joint: rotation about a roughly vertical axis of the hook attachment point

Since the tool is suspended by wire from the boom tip the wire dispatch angle at the boom tip cannot be controlled. However, in the applications the wire dispatch angle is effectively limited to +/- 6° in the two horizontal directions. The following image 1.2 shows the boom with minimum and maximum angle with maximum tool height. The figure also depicts optimum working range (medium luffing angle) with minimum tool height.

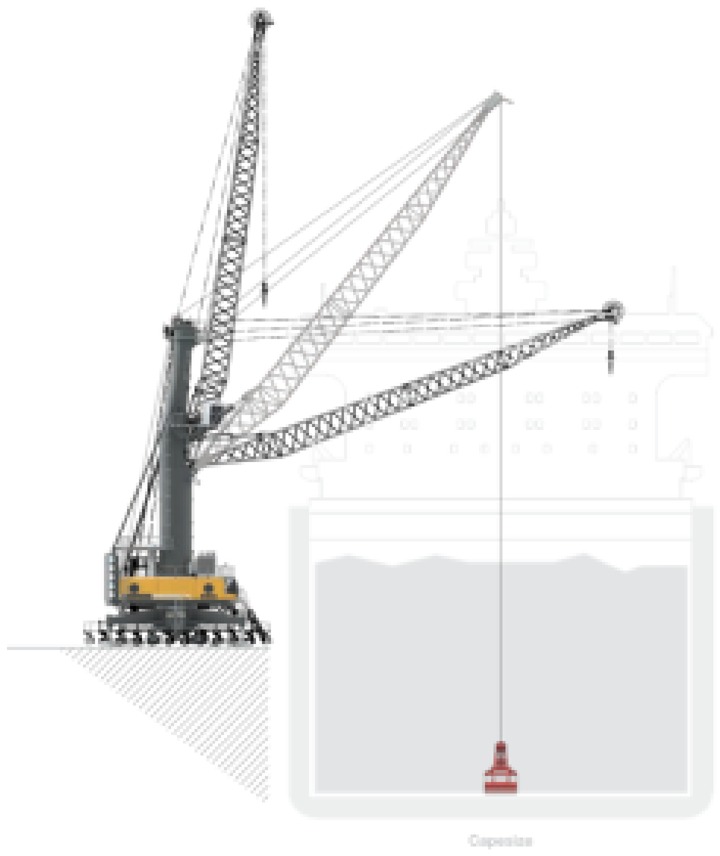
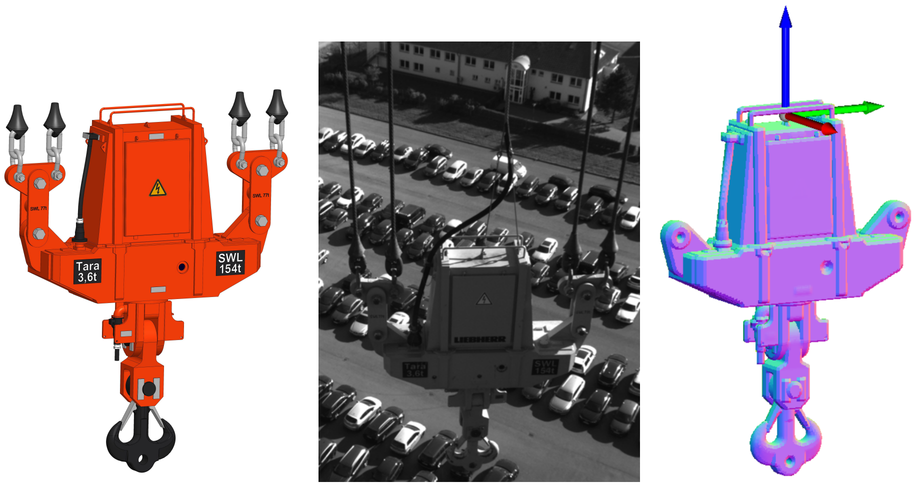


Figure 1.2: LHM 550 effective working range. boom with minimum and maximum luffing angle with maximum tool height. boom with medium luffing angle with minimum tool height
Source: own elaboration

1.2.2 The Hook

This section presents the tool which has to be tracked. The hook is displayed in figure 1.3. It can be seen, that the hook is supended by four ropes and has an additional cable to supply the rotating joint. The hook comprises several moving parts but movement is effectively limited to small angles during normal operation by gravity. Therefore, it is assumed that the hook is a rigid body. The hook is spanning 0.62 meters in depth (along the x axis), 2.75 meters in width (y axis), and 3.95 meters in height (z axis). For the purpose of accurate pose estimation a coordinate frame origin is defined for the hook. As can be seen in fig. 1.3c, the coordinate of the hook is located vertically at 5 centimeters above the top surface of the hook, and centered in the two horizontal directions. the



(a) (b) (c)

Figure 1.3: The tool (hook) of the LHM.
Source: own elaboration

pose of the hook is described as the pose of the origin of this coordinate frame.

1.2.3 The Sensor Setup

The sensor setup is a rigid construction that holds the optical sensors. It is mounted on the boom of the LHM near the boom tip as can be seen in fig. 1.4. One LiDAR and four cameras. The placement of the sensor setup on the crane was chosen in a manner, so that the tool is visible for the sensor at all times in nearly all machine poses. The optical sensors are described in detail in the following subsections.

1.2.3.1 The LiDAR Sensor

One of the most important pieces of equipment for this thesis is the LiDAR. LiDAR is an acronym for **Li**ght **D**etection and **R**anging. LiDAR sensors consist of a laser emitter and a receiver. The emitter projects a laser beam into the environment, and the receiver captures the reflected laser pulses. By measuring the time it takes for the laser pulses to return, the sensor can determine the distance to the objects in the scene. By incrementally changing direction of the laser between pulses, The LiDAR sensor generates a dense point cloud representation of the environment.

Different LiDARs can have very different properties depending on their requirements. Since the LiDAR in this case has already been selected, a short

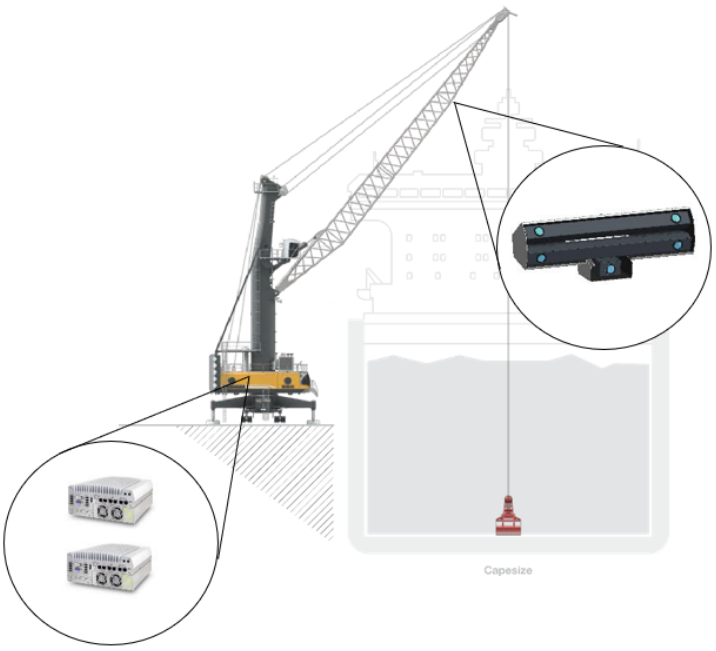


Figure 1.4: LHM 550 with sensor setup (right) and the two IPCs (left)
Source: own elaboration

description of the product is given and its most important properties are listed in table 1.5

Attribute	Value	Unit
Laser Wavelength	905	nm
Detection Range	0.5 - 90 (10% reflectivity)	m
FOV	0.5 - 81.7 x 25.1	°
Distance Error	< 0.02 per 20	m
Point Rate	< 240,000	points/s

Table 1.5: Most important properties of the LiDAR "livox Horizon"
Quelle: Tech, 2019

Most LiDARs have an acquisition pattern of straight lines with equal spacing. The "livox Horizon" was mainly chosen for this application because it has a pseudo-random aquisition pattern as depicted in fig 1.6. With straight lines, it would be possible for the ropes connecting the hook and the crane to slip through the spacing between the lines and not get represented in the resulting point cloud.

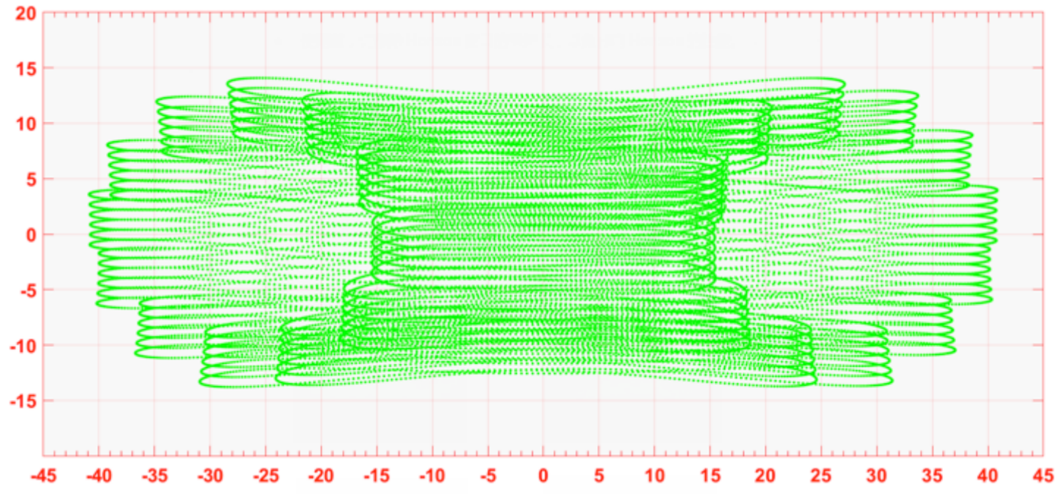


Figure 1.6: Field of View of the LiDAR "livox Horizon". axes are in degrees. points have been accumulated for 100 ms
Source: Tech, 2019

1.2.3.2 The camera system

Four grey-scale camera, of which two cameras each form a 3D stereo camera set covering different fields of view, are mounted on the sensor setup. From the camera images a map of the workspace surrounding the crane is built by other parties involved in the supervised autonomy project. For this thesis, the cameras do not play a major role.

1.2.4 The Automation Control Unit (ACU)

Sensor data from the sensor setup is sent via UDP to the Automation Control Unit (ACU). The ACU consists of two industrial PC (IPC)s. The IPCs operate on Linux Ubuntu 20.04 with Robot Operating System 2 (ROS2) galactic. During test drives with the LHM, the sensor data from the sensor setup, along with the crane kinematics and other machine parameters is converted to ROS2 standard message formats and saved in a rosbag.

1.2.5 The ROS2-Simulation

The ROS2-Simulation is the starting point for this master thesis. Because of obvious reasons (expensive, hazardous, ...), it would not be feasible to test the implemented algorithms directly on the real LHM. In the simulation, experiments can be conducted without the danger of causing damage or hurting people.

The Simulation contains a simple kinematic model of the LHM and allows for playback of the acquired sensor data, as if it was directly streamed from the sensors.

1.3 State-of-the-Art Pose Estimation with IMUs

State-of-the-art tool tracking for rope-guided tools estimates the tool position using a Luenberger observer (refer to [Luenberger, 1964]) and data from gyroscopes attached to the ropes at the boom tip (refer to [Schaper et al., 2011]). This so called "load position observer" system (LPO) is in use at Liebherr since 2012.

This method falls short due to the following reasons:
Accuracy not high enough:
The values in table 1.7 shows the accuracy of the LPO according to Liebherr.

Attribute	Value	Unit
Accuracy tool position	$\Delta x < 0.5$	m
	$\Delta y < 0.5$	m
	$\Delta z < 0.5$	m
Accuracy tool orientation	$\Delta\gamma < 2$	°
Time resolution tracking	>25	Hz
Computation time	$<< 1$	ms

Table 1.7: Accuracy of the state-of-the-art load position observer

cannot accommodate for errors in the kinematic chain of the crane:
Since the reference system is the crane origin (base frame), the accuracy of this method is subject to the errors in the kinematic chain of the crane from the base to the boom tip.
difficult installation: Another major downside of this method are the difficulties which arise during attachment and removal of the sensors. The sensors are dangling on the ropes. During assembly and disassembly of the crane, these sensors get damaged frequently.

1.4 Accommodating for Errors in the kinematic Chain of the Crane

The position and orientation of the base frame of the LHM is known with an accuracy of a few centimeters due to differential GPS measurements. However, due to errors in the kinematic chain from the base to the boom tip stemming mainly from bending of the boom due to load shifts, but also other influences, the actual position of the boom tip, and consequently the position of the sensor setup may vary up to 1 meter from the position received as machine parameter. To accommodate for these errors, the sensor setup is not referenced to the base frame of the crane directly, but rather to the world itself. For this purpose, other parties at Liebherr in corporation with the Austrian Institute of Technology (AIT) enable construction of a map of the crane workspace based on the optical sensors, and reference the sensor setup to the world with the help of landmarks in the map. This approach bypasses the errors in the kinematic chain of the cranes.

1.5 Goals of the Thesis

As the motivation for the thesis and the prerequisites have now been thoroughly explained, it is possible to define the goals of the thesis. At the core of this endeavor lies the pursuit of accurately determining the pose — comprising both the spatial location and orientation — of a tool, specifically the crane hook, in relation to the world. The ultimate aim is to achieve this pose estimation in real-time, responding promptly to dynamic scenarios and enabling efficient decision-making processes. However, the current articulation of this goal merits refinement to convey its essence more concisely.

1.5.1 Selection of suitable Algorithms for Pose Estimation of Rope-guided Tools

The scientific field of computer vision is well researched, but the research is far from being concluded. Apart from the fact, that there are different sensors for the acquisition of data, there is a manifold of algorithms for each sensor or combinations of them. Therefore, and since this is pioneer work done in the context of the Liebherr group, the first goal is to find suitable algorithms for pose estimation of rope-guided tools based on the given sensor data (grey-scale stereo images and point cloud).

1.5.2 Implementation and Testing of selected Algorithms

After the selection process, suitable algorithm candidates are implemented and evaluated. However, the evaluation of these algorithms is a challenging task in itself. The most imminent problem is the missing ground truth. Ground truth refers to the accurate and objective reference data that serves as a benchmark or standard against which other measurements, predictions, or estimations can be compared. Therefore it is difficult to formulate precise statements of how accurate an implemented algorithm is. The workaround for this is a "best effort" estimation of the ground truth as explained below.

1.5.2.1 Implementation of the Best Effort Ground Truth Estimation

The first goal is a best effort ground truth estimation. This means to implement an algorithm that estimates the pose of the hook as accurately as possible. The best effort is acquired discarding the real-time requirements and taking as much time as needed to estimate the pose of the hook. The result from this method can then be used to evaluate the real-time application. In this way, although the true pose is still unknown, a deviation from the best solution possible can be retrieved. For this purpose, the requirements listed in table 1.8 have been defined in accordance with Liebherr.

Attribute	Value	Unit
Accuracy tool position	$\Delta x < 0.1$	m
	$\Delta y < 0.1$	m
	$\Delta z < 0.1$	m
Accuracy tool orientation	$\Delta \alpha < 5$	$^\circ$
	$\Delta \beta < 5$	$^\circ$
	$\Delta \gamma < 5$	$^\circ$
Time resolution tracking	>10	Hz
Computation time	< 2	s

Table 1.8: Best effort ground truth estimation requirements

1.5.2.2 Implementation of a Real-Time capable Approach

The second goal is to implement an algorithm capable of tracking the tool pose in real-time and to evaluate the algorithm based on the deviation from the best effort ground truth. The values listed in table 1.9 describe the requirements for the real-time approach.

Attribute	Value	Unit
Accuracy tool position	$\Delta x < 0.2$	m
	$\Delta y < 0.2$	m
	$\Delta z < 0.2$	m
Accuracy tool orientation	$\Delta \alpha < 11$	°
	$\Delta \beta < 11$	°
	$\Delta \gamma < 11$	°
Time resolution tracking	>10	Hz
Computation time	< 50	ms

Table 1.9: Real-time pose estimation requirements

2 Overview/Selection of Pose Estimation Algorithms

This chapter gives an overview of approaches which have been considered to estimate the pose of the hook and an explanation of the problems and difficulties that arise when applied to the problem at hand It is noted, that this is not a complete study on all possibilities, but rather a short notion of approaches that were considered.

2.1 Sensor Selection for Pose Estimation

In the context of pose estimation, the choice of sensor plays a pivotal role in determining the accuracy, reliability, and computational feasibility of the system. This section explains the decision-making process behind selecting a LiDAR sensor over stereo cameras for the acquisition of point cloud data and subsequent pose estimation.

2.1.1 Stereo Cameras and Depth Estimation

Stereo cameras have long been employed in computer vision applications for depth estimation based on the principles of triangulation. By capturing images from two slightly offset viewpoints, stereo vision systems determine the disparity between corresponding image points to compute depth information. However, upon evaluation of stereo camera-based depth maps, certain limitations came to light.

2.1.1.1 Limited Depth Resolution

One of the primary concerns with stereo camera-based depth estimation lies in the limitation of depth resolution. While stereo cameras can provide accurate depth information for objects within a certain range, the accuracy and precision diminish with increasing distance from the cameras. This is particularly pronounced when dealing with scenes that encompass objects at varying distances. In contrast, the LiDAR sensor excels in providing consistent and high-resolution depth measurements across a wide range, rendering it more suitable for scenarios where accurate localization of distant objects is imperative.

2.1.1.2 Computational Complexity

Another significant drawback of relying on stereo cameras is the computational complexity associated with depth map generation and subsequent 3D reconstruction. The rectification, feature matching, and disparity calculation processes demand substantial computational resources, making real-time or near-real-time performance challenging to achieve. This is particularly critical in dynamic environments where the pose estimation system must respond swiftly to changing conditions. The computational efficiency of the system directly influences its practicality in real-world applications.

2.1.1.3 Environmental Robustness

Furthermore, it is crucial to address the impact of environmental conditions on sensor performance. LiDAR sensors often outperform stereo cameras in challenging scenarios, such as low light, rain, and fog. Stereo cameras heavily rely on image quality and feature extraction, which can be compromised in adverse weather. The active sensing principle of LiDAR sensors ensures reliable data acquisition even in conditions where stereo cameras might struggle, enhancing the robustness of the overall pose estimation system.

2.1.2 Lidar Sensor and Point Cloud Acquisition

LiDAR sensors operate on the principle of emitting laser pulses and measuring the time-of-flight for the reflected signal to calculate distances. This technique offers several advantages that align well with the requirements of our pose estimation system.

2.1.2.1 Consistent Depth Resolution

LiDAR sensors provide consistent depth resolution across a wide field of view, ensuring reliable distance measurements regardless of the object's position within the scene. This consistency is invaluable for accurate pose estimation, especially in scenarios where objects of interest span various distances. The uniformity of depth information facilitates robust localization and recognition of objects, contributing to the overall accuracy of the system.

2.1.2.2 Data Density and Sparsity

An additional advantage of LiDAR-based point cloud acquisition is the high data density captured by the sensor. Point cloud data generated by LiDAR sensors contain detailed spatial information, even for complex scenes with textureless surfaces or challenging lighting conditions.

2.1.2.3 Computational Efficiency

Compared to stereo camera-based methods, LiDAR-based point cloud acquisition requires significantly less computational effort. The direct measurement of distances through laser pulses eliminates the need for complex feature matching and disparity calculations. This reduction in computational complexity leads to faster and more responsive pose estimation, enabling real-time or near-real-time performance in dynamic environments.

2.1.3 Conclusion

In conclusion, the decision to employ a LiDAR sensor for point cloud acquisition and subsequent pose estimation is grounded in the superior depth resolution, consistency, environmental robustness, data density, and computational efficiency it offers over stereo cameras. The accurate and reliable distance measurements provided by the LiDAR sensor address the limitations posed by varying depth resolution in stereo cameras. Moreover, the reduced computational demands of the LiDAR-based approach enable timely and efficient pose estimation, enhancing the practicality of the system in real-world applications.

2.2 Selection of the Pose Estimation Algorithm

2.2.1 Model Based Feature Matching

The most common approach for 6 DoF pose estimation from point clouds is model based feature matching. A feature in this context is a distinctive point or region in an image or point cloud that has unique geometric properties. A descriptor is the numerical representation of geometric properties in a feature space. In general descriptors can be divided into local, regional and global descriptors. With local descriptors each point in the point cloud is represented by a descriptor in the feature space and with global descriptors the whole cloud is represented by one descriptor (refer to Aldoma et al., 2012). Regional descriptors are a mixture of these two where points are clustered together and one descriptor represents a cluster. Local descriptors are typically used for pose estimation.

Several descriptors including, but not limited to, the Point Pair Feature (PPF) (refer to Drost et al., 2010) and the Fast Point Feature Histogramm (FPFH) (refer to Rusu et al., 2009) were implemented and tested, along with different Random Sample and Consensus (RANSAC) based matching algorithms like the initial alignment algorithm described in Rusu et al., 2009.

2.2.1.1 Computational Expenses in Feature Space

Model-based feature matching involves the comparison of detected image features with the corresponding features of the 3D model. These comparisons are typically performed within a high-dimensional feature space, demanding extensive computational resources. As it is aimed to achieve real-time or near-real-time performance in dynamic environments, the computational complexity of this process is a paramount concern.

Despite various efforts, computation time for model based feature matching did not reach values smaller than 300 milliseconds, which is nowhere near the specified requirements of 50 milliseconds.

2.2.1.2 Quality loss with increasing Distance

While initial results seemed promising, particularly at very close ranges, a significant decline in matching quality was encountered as the distance between the object and the sensor increased. The degradation in matching quality with increasing distance is rooted in the decreasing resolution of the point cloud. Fig. 2.1 shows point clouds of the hook captured by the LiDAR from different distances. The point cloud at close distance has approx. 800 points on the hook and is suited for feature matching. The point cloud at a far distance of about 40 meters only contains 67 points belonging to the hook, which is not enough to reliably represent the underlying geometry.

2.2.2 Encompassing the Point Cloud with a Bounding Box

In light of the shortcomings encountered with model-based feature matching, the approach of encompassing the points belonging to the hook with a bounding box emerged as a compelling alternative. Bounding boxes have a long history in algorithmic geometry and can efficiently encapsulate complex geometric structures, thereby saving computation time. This method involves fitting a cuboid around the points representing the object of interest, leveraging the position and orientation of the bounding box to estimate its pose. In comparison to the model based feature matching, which is designed to to be robust against cluttered scenes, the bounding box is very sensitive to outliers. Since the bounding

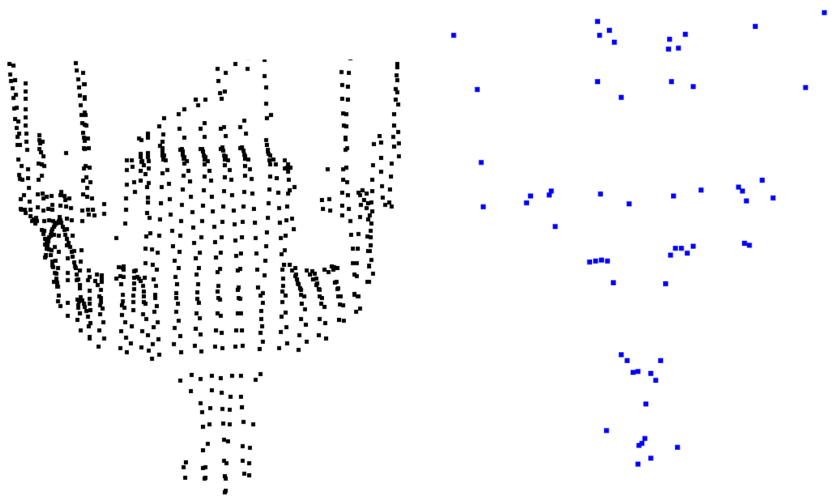

(a) close distance: ca. $10m$ (b) far distance: ca. $40m$

Figure 2.1: Point clouds of the hook captured by the LiDAR from different distances.
Source: own elaboration

box simply encompasses all points in the cloud a single outlier could significally change the result, so careful consideration has to be taken to the segmentation of the point cloud. Despite the need for accurate segmentation, the bounding box approach has shown promise in providing accurate and reliable pose estimation for the hook object.

2.2.2.1 Conclusion

In summary, the exploration of model-based feature matching for the task of pose estimation highlighted its limitations stemming from both computational expenses and the distance-dependent degradation of matching quality. The high computational demands within the feature space rendered the approach unsuitable for achieving the real-time performance required in dynamic scenarios. Additionally, the sensitivity of the approach to variations in distance compromised its robustness and accuracy.

To fulfill the real-time requirements, it is imperative to either avoid transformation into a high-dimensional feature space or to utilize a more efficient matching algorithm. Given the sparse nature of the data points and the observed limitations, it became evident that finding a pose estimation algorithm that operates solely within the three-dimensional Euclidean space of the point cloud is a more promising direction.

2.3 Other Approaches considered for Pose Estimation

2.3.1 Neural Networks

Neural networks find increasing popularity in computer vision and many other applications. Going into detail on this topic could be a master thesis of it's own and is beyond the scope this thesis. The main problem for this approach is the acquisition of training data. For the LHM 500 this training data is not readily available and therefore these methods have very limited usability for the problem at hand.

2.3.2 Mono 3D Image Pose Estimation

In most cases, images are used to detect objects (finding out whether an object is present in an image) or also to locate objects, but not to infer their pose (which includes the orientation). Even though there are algorithms that tackle the problem of 3D-Pose Detection of objects based on image data, they usually fail, when the rotation exceeds a certain threshold (ca. $\pm 30\,\mathrm{deg}$) from a predefined viewpoint or diverge over time.

2.4 Conclusion

Having examined the strengths and limitations of various pose estimation approaches, the subsequent chapter delves into the theoretical background of algorithms and concepts that play a pivotal role in shaping our solution. This chapter equips us with the foundational knowledge required to comprehend the intricacies of the bounding box-based pose estimation technique. Subsequently, we transition into Chapter 1.5.2.1, where the detailed explanation and evaluation of the bounding box pose estimation method begins.

3 Theoretical Background

In this chapter, the most important algorithms and concepts which contribute to the pose estimation of the hook are described in detail. Section 3.1 explains the RANSAC algorithm, which is used in chapter 4.3 for the segmentation of the point cloud. Section 3.2 describes the Iterative Closest Point (ICP) and derivatives of the ICP which are used in chapter 1.5.2.1 and in section 3.4, the Kalman filter is thouroughly discussed. The application of the Kalman filter is described in chapter 5.3.

3.1 Robust Model Parameter Estimation in Point Clouds using Random Sample Consensus (RANSAC) Algorithm

The accurate estimation of model parameters for geometric primitives in point clouds is a fundamental task in computer vision and 3D reconstruction applications. However, real-world point cloud data is often corrupted with noise and outliers, making traditional parameter estimation methods susceptible to errors. In this section, the RANSAC algorithm (refer to [Fischler and Bolles, 1981]) is introduced: a robust and widely adopted technique for estimating model parameters in the presence of outliers. RANSAC's application in point cloud analysis facilitates the reliable identification of geometric primitives, such as planes, lines, or spheres. In pose estimation, RANSAC can be employed to determine the transformation between two coordinate systems

RANSAC is an iterative algorithm designed to robustly estimate model parameters from a data set containing outliers. The primary idea behind RANSAC is to randomly sample a minimal subset of data points to form a candidate model and then evaluate its fitness to the entire data set based on a user-defined threshold. By iteratively repeating this process, RANSAC efficiently identifies inliers, which are data points that conform to the candidate model, while disregarding outliers.

3.1.1 Outline of the RANSAC Algorithm for Model Paramater Estimation

The pseudo code of algorithm 1 represents the RANSAC algorithm for model parameter estimation in point clouds, as it is mainly used in this thesis. The RANSAC algorithm can be described through the following steps:

Step 0: Initialization:

The model type and a minimum sample size n required to estimate the model parameters is defined. For instance, three points are needed to estimate a plane or the pose of a model, four points for a sphere, and two points for a line. A desired confidence level C, a threshold value ϵ and a maximum number of iterations k is set. An input point cloud P has to be provided.

Step 1: Iterative Sampling:

Repeat for a fixed number of iterations or until a desired confidence level is reached: Randomly select a minimal subset of data points (minimum sample size) from the point cloud to form a candidate model.

Step 2: Model Fitting:

Use the selected data points to compute the model parameters for the geometric primitive (e.g., plane coefficients, sphere center and sphere radius, or line equation).

Step 3: Inlier Selection:

Evaluate the fitting error for each data point in the entire point cloud based on the candidate model. The user-defined threshold ϵ is used to distinguish inliers from outliers. Data points with errors below the threshold are considered inliers, while those with errors above the threshold are treated as outliers.

Step 4: Model Evaluation:

Count the number of inliers obtained from Step 4. If the number of inliers improves compared to previous iterations, consider the current candidate model as a potential solution.

Step 5: Termination:

If the maximum number of iterations is reached, the algorithm is considered to have converged and returns the best model parameters M^* and the inlier set I^*.

These are the essential steps for the RANSAC algorithm. Naturally, RANSAC can be adapted to fit the problem at hand. In algorithm 1, lines 6, 18, and 22 are optional steps which are not necessary for the algorithm but have been added to improve the performance of the algorithm for the problem at hand. These additions are described in the next subsection (3.1.2)

Algorithm 1 RANSAC Algorithm for Model Parameter Estimation

```
Require: Point cloud data $P$,
    Minimum sample size $n$,
    Confidence level $C$,
    Maximum iterations $k$,
    Inlier threshold $\epsilon$
Ensure: Best model parameters $M^*$ and corresponding inliers $I^*$
 1: Initialize best model parameters $M^*$ and inliers $I^*$ to empty
 2: Set maximum number of inliers $N_{\text{inliers}}$ to 0
 3: for $t = 1$ to $k$ do
 4:   Randomly select $k$ points from $P$ to form a candidate model
 5:   Estimate model parameters $M_k$ using the selected points
 6:   if $M_k$ is a valid model /* optional */ then
 7:     Initialize current inliers $I_k$ to empty
 8:     for each data point in $P$ do
 9:       Compute the error $e$ between point and the candidate model $M_k$
10:       if $e < \epsilon$ then
11:         Add point to $I_k$
12:       end if
13:     end for
14:     if Number of inliers in $I_k$ is greater than $N_{\text{inliers}}$ then
15:       Update best model parameters $M^*$ to $M_k$
16:       Update inliers $I^*$ to $I_k$
17:       Set $N_{\text{inliers}}$ to the number of inliers in $I_k$
18:       Calculate the required number of iterations $k$ based on $N_{\text{inliers}}$ and
          the confidence level $C$ /* optional */
19:     end if
20:   end if
21: end for
22: optimize model parameters $M^*$ /* optional */
23: return Best model parameters $M^*$ and inliers $I^*$
```

3.1.2 Additions to the RANSAC-Algorithm

Algorithm 1 contains three lines which are optional and not necessary for the algorithm to work.
Step 1a: preemptive model rejection:
For the time savvy user, in this optional step models can be rejected based on arbitrary conditions, before the evaluation, which is the most computationally expensive procedure, is executed. A typical example would be, to preemptively reject a model, if the same data points have already been selected.
As it will be described later in chapter 3.1, RANSAC is used to segment the ropes and parts of the hook from the point cloud. Since it is known, that the ropes are roughly vertical, a model constraint which only counts a model as valid if their axis is roughly aligned vertically, is implemented.
Step 5a: Ensure the confidence level
Fischler and Bolles, 1981 have shown, that the number of iterations n needed to find the best solution (a model incorporating all true inliers) can be determined as a function of the probability of success (= confidence level C). Let p be the desired probability of success and w the the ratio of number of inliers N_I to number of points in the cloud N_P. That is

$$w = N_I / N_P$$

. Then, w^n is the probability that all samples drawn in one iteration are inliers and $1 - w^n$ the probability that at least one sample is an outlier and therefore wrong model parameters are calculated. Consequently, $(1 - w^n)^k$ is the probability that the sample sets from all iterations contain at least one outlier each, which is the same as $1 - p$, the probability of an unsuccessful termination of RANSAC. This means that, starting from

$$1 - p = (1 - w^n)^k$$

and taking the logarithm of both sides of the equation yields

$$k = \frac{\log(1 - p)}{\log(1 - w^n)}$$

, which is the required number of iterations as a function of p. In many cases, the number of true inliers is initially unknown, but this problem can be compensated by initially choosing a large k and then updating k iteratively based on the number of inliers in the current best Model N_{I^*}.
Step 6: model optimization

The model with most inliers was found based on the sample of a minimum size consensus set. The parameters can be further improved by fitting the model to all inliers. To fit the model parameters to the inliers, an Least Mean Squared Error (LMSE)-optimization is deployed in this step.

3.1.3 Advantages and Limitations of RANSAC

The RANSAC algorithm offers several advantages for geometric primitive estimation in point clouds:

Robustness to outliers: RANSAC effectively deals with outliers, ensuring accurate model parameter estimation even in the presence of noisy data.

Flexibility: The algorithm can be adapted to fit various geometric primitives, making it versatile for different 3D reconstruction tasks.

Simplicity: The algorithm is relatively straightforward to implement and computationally efficient. However, it also has some limitations:

Sensitivity to parameters: The choice of parameters, such as the minimum sample size and inlier threshold, can impact the performance of RANSAC and necessitates careful parameter tuning.

Computational cost: The iterative nature of RANSAC may lead to increased computational overhead, especially for large point clouds.

Non-deterministic output and computation time: Since RANSAC is based on random resampling, convergence of the algorithm is a statistical process, which can be problematic for real-time applications.

3.2 Point Cloud Registration with the Iterative Closest Point (ICP) Algorithm

In the field of computer vision and robotics, point cloud registration plays a vital role in aligning and fusing multiple 3D point clouds obtained from different sensors or viewpoints. Registration in this context refers to the process of finding the transformation that aligns a source point cloud to a target point cloud. In Scan-Matching, Registration is used to align consecutive sensor readings taken from different viewpoints. In 6 DoF pose estimation registration is used to align a point cloud modelled from CAD-Data of an object (source cloud) to a sensor reading (target cloud) to recover the pose of the object. The ICP algorithm (refer to Holz et al., 2015) has emerged as a prominent method for achieving accurate and efficient point cloud registration.

This section presents a basic exploration of the ICP algorithm, highlighting its fundamental concepts, mathematical formulation, and potential applications.

3.2.1 Outline of the ICP Algorithm

The ICP algorithm operates in an iterative manner to refine the transformation between two point clouds. At each iteration, the algorithm performs the following steps:
Step 1: Correspondence Search:
The algorithm establishes correspondences between points in the source point cloud and the target point cloud based on the euclidean distance. This step pairs each point in the source cloud with its closest counterpart in the target cloud.
Step 2: Transformation Estimation:
Given the established correspondences, the ICP algorithm estimates the transformation that minimizes the distance between corresponding points.

$$T = \arg\min_T \sum_i w_i \|m_i - T \cdot b_i\|^2 \tag{3.1}$$

Step : Convergence Check:
The algorithm checks for convergence by comparing the difference between the current and previous transformations. If the difference is below a predefined threshold, the algorithm terminates as it has achieved alignment. $\arg\max_x$

3.2.2 Point-To-Plane ICP

Several Variations to the ICP algorithm have been introduced since the publication of ICP. Based on the assumption, that the point cloud is actually representing the surface of an object, the Point-To-Plane ICP algorithm improves the performance of ICP by incorporating surface normal information of the target cloud. The transformation estimation (refer to eq. 3.1) is adapted so that it minimizes the error along the surface normal, not incorporating other directions. Eq. 3.2 shows the updated transformation estimation:

$$T = \arg\min_T \sum_i w_i \|n_i \cdot (m_i - T \cdot b_i)\|^2 \tag{3.2}$$

By calculating the dot product of the error (distance) and the surface normal, the error is projected onto the surface normal of the target cloud. This way, a tighter alignment and a faster convergence is achieved.

3.2.3 Generalized ICP

The generalized ICP takes the idea behind the point-to-plane ICP one step further and combines the point-to-point metric of the standard ICP with the

Algorithm 2 Iterative Closest Point (ICP) Algorithm

```
Require: Source Point Cloud $A = \{a_i\}$,
    Target Point Cloud $B = \{b_i\}$
    Initial Transformation $T_0$
    Correspondence Threshold $d_{\max}$ Maximum Iterations $K$
Ensure: TransformationMatrix $T$
 1: Initialize TransformationMatrix $T = T_0$
 2: repeat
 3:     for $i = 1$ to $N$ (Number of points in $B$) do
 4:         $m_i$ = Find closest Point to $a_i$ in $T \cdot b_i$
 5:         if $\|m_i - T \cdot b_i\| \leq d_{\max}$ then
 6:             $w_i = 1$
 7:         else
 8:             $w_i = 0$
 9:         end if
10:     end for
11:     Estimate the transformation that best aligns SourcePointCloud to Tar-
        getPointCloud by minimizing distance of all correspondences
12:     Check for convergence (e.g., the change in transformation below a thresh-
        old)
13: until Converged or maximum iterations $K$ reached
14: return $T$
```

point-to-plane metric and adds also a plnae-to-plane metric. the generalized ICP basically represents points on surfaces locally by covariance matrices and performs a data association which takes the shape of these covariances into account in the transformation estimation. the interested reader is referred to Segal et al., 2009. For the application of the generalized ICP it is important to know, that the point cloud normals, i. e. vectors representing the direction of the surface of the underlying geometry of the point cloud are needed. How the point cloud normals are obtained is explained in the next section.

3.3 Estimating Surface Normals of a Point Cloud

Estimating surface normals of point clouds is a fundamental task in computer graphics, computer vision, and 3D geometry processing. Surface normals provide crucial information about the orientation of a point on the surface of an object, which is essential for tasks like shape analysis, object recognition, and rendering. There are many methods for estimating surface normals of point clouds. A comprehensive study on surface normals is done by Klasing et al., 2009. The approach taken in this thesis for estimating normals involves using Principal Component Analysis (PCA) on the covariance matrix of points in the local neighborhood as described in the following.

3.3.1 Local Neighborhood Selection

For each point in the point cloud, a local neighborhood of nearby points needs to be selected. The size of this neighborhood is a parameter that can be adjusted based on the density of the point cloud and the desired level of detail. Common methods for selecting neighbors include fixed-radius search or a fixed number of nearest neighbors.

3.3.2 Covariance Matrix Computation

Once the local neighborhood is selected for a point, the covariance matrix of the neighborhood points is computed. To account for the fact that the local point neighborhood is not centered around the origin, we calculate the centroid of the local neighborhood points and use it in the covariance matrix calculation. The covariance matrix is a symmetric 3×3 matrix and is calculated according to:

$$C = \frac{1}{N} \sum_{i=1}^{N} ((p_i - \bar{p}) \cdot (p_i - \bar{p})^T) \tag{3.3}$$

Where:

C is the covariance matrix.
N is the number of points in the local neighborhood.
p_i represents a 3D point in the local neighborhood.
$\bar{p}$ is the centroid of the local neighborhood points.

$$\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p_i$$

$(p_i - \bar{p})^T$ is the transpose of the vector difference $(p_i - \bar{p})$.

3.3.3 Eigenvalue Decomposition

After computing the covariance matrix, an eigenvalue decomposition is performed. Eigenvalues and eigenvectors of the covariance matrix are calculated. These eigenvalues represent the variances of the point distribution along the principal axes of the local neighborhood, and the corresponding eigenvectors represent the directions of those principal axes.

$$C = V\lambda V^{-1}$$

Where:

C is the covariance matrix.
V is a matrix of eigenvectors.
λ is a diagonal matrix of eigenvalues.

3.3.4 Normal Estimation

The eigenvector corresponding to the smallest eigenvalue (i.e., the direction with the least variance) is chosen as the estimated normal direction for the point. This is because the eigenvector with the smallest eigenvalue corresponds to the axis along which the local neighborhood is most spread out. In other words, it indicates the direction of the surface normal.

3.3.5 Orientation and Consistency

The computed normal might need to be flipped to ensure consistency across the point cloud. This can be done by comparing the computed normal with the viewing direction or by considering the orientation of neighboring normals.

3.4 Tracking the Pose of an Object with a Kalman Filter

The Kalman filter is an efficient recursive algorithm that estimates the state of a dynamic system from a series of noisy measurements. It was first published in [Kalman, 1960] and has since been widely used in various applications, such as tracking, navigation, control systems, computer vision, and robotics, where it is crucial to estimate the state of a system in the presence of noise and uncertainty. The filter output is an optimal estimate of the true state by combining predictions from a mathematical model (system dynamics) and measurements from various sensors. The filter minimizes the mean square error between the estimated state and the true state, assuming that the system and measurement noises are Gaussian and the model is linear.

3.4.1 Outline of the Kalman Filter

The Kalman filter consists of two main steps: the prediction step and the update step. In each time step, the filter predicts the next state based on the system dynamics and the previous state estimate, and then updates the state estimate using new measurements.
Step 0: Initialization:
Define the state vector $\mathbf{x}$ representing the variables to estimate (position, velocity, etc.). Initialize the state estimate $\mathbf{x}$ and the state covariance matrix $\mathbf{P}$ representing the uncertainty in the state estimate.
Step 1: Prediction:
The next state $\mathbf{x}$ is predicted based on the system dynamics and the control inputs (if any). The state covariance matrix $\mathbf{P}$ is updated based on the process noise covariance matrix $\mathbf{Q}$ that accounts for uncertainty in the prediction.
Step 2: Update:
Measurements $\mathbf{z}$ are obtained from the sensors. The measurement residual $\mathbf{y}$ is calculated by comparing the predicted measurements with the actual measurements. The Kalman gain $\mathbf{K}$ is calculated, which determines the trade-off between the prediction and measurement update. The state estimate $\mathbf{x}$ is updated based on the measurement residual and the Kalman gain. The state covariance matrix $\mathbf{P}$ is updated based on the measurement noise covariance matrix $\mathbf{R}$ that accounts for uncertainty in the measurements. The Kalman filter iteratively repeats the prediction and update steps for each time step to continuously estimate the current state of the system.

Algorithm 3 Kalman Filter

```
 1: **Initialization:**
 2: Initialize state vector $\mathbf{x}$ and state covariance matrix $\mathbf{P}$
 3: **repeat**
 4:     **Prediction Step:**
 5:     Predict the state $\hat{\mathbf{x}}_t$ using system dynamics and control input $\mathbf{u}_t$
 6:     Update the state covariance matrix $\mathbf{P}_t$ based on the process noise covari-
        ance matrix $\mathbf{Q}$
 7:     **Update Step:**
 8:     Get measurement vector $\mathbf{z}_t$ from sensors
 9:     Calculate the measurement residual $\mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_t$
10:     Calculate the Kalman gain $\mathbf{K}_t = \mathbf{P}_t\mathbf{H}^T(\mathbf{H}\mathbf{P}_t\mathbf{H}^T + \mathbf{R})^{-1}$
11:     Update the state estimate $\mathbf{x}_t = \hat{\mathbf{x}}_t + \mathbf{K}_t\mathbf{y}_t$
12:     Update the state covariance matrix $\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{H})\mathbf{P}_t$
13: **until** you die
```

3.4.2 Advantages and Disadvantages of the Kalman Filter

It's important to note that the Kalman filter assumes linearity of the system dynamics and Gaussian noise for both the process and measurement models. If the system is nonlinear, an Extended Kalman Filter (EKF) or a more advanced technique like the Unscented Kalman Filter (UKF) could be used to handle non-linearity. Additionally, in practice, careful tuning of the process noise covariance matrix **Q** and the measurement noise covariance matrix **R** is essential to achieve accurate and robust state estimation.

4 Implementation of the best Effort Ground Truth Estimation

This chapter describes how the pose of the hook is recovered when the real-time requirements can be disregarded. It is called the best effort ground truth estimation, because it is expected yield more accurate results than the real-time capable approach and is used as a benchmark to evaluate other methods. Firstly, an overview of the approach is given, and then the steps are described in detail. Finally, the evaluation of the approach is presented.

4.1 Pose Estimation Pipeline of the best Effort Ground Truth Estimation

This section gives an overview of the Pipeline of how to extract the pose of the hook from the raw LiDAR-sensor data. Figure 4.1 shows the pipeline. After some preprocessing, the point cloud is downsampled and segmented into smaller clusters of points to extract information from the cloud. "Pose Estimation" refers to the process of generating a pose hypothesis based on the information extracted from the segmented point cloud. The information from the segmentation process is used to estimate the orientation of the hook and a bounding box is fitted around the points of the hook to estimate its position. The so retrieved pose is then used as an initial transformation for the registration process, where the result is refined by aligning a model cloud to the sensor data using generalized ICP. The registration requires point cloud normals, so these are estimated in the step before.

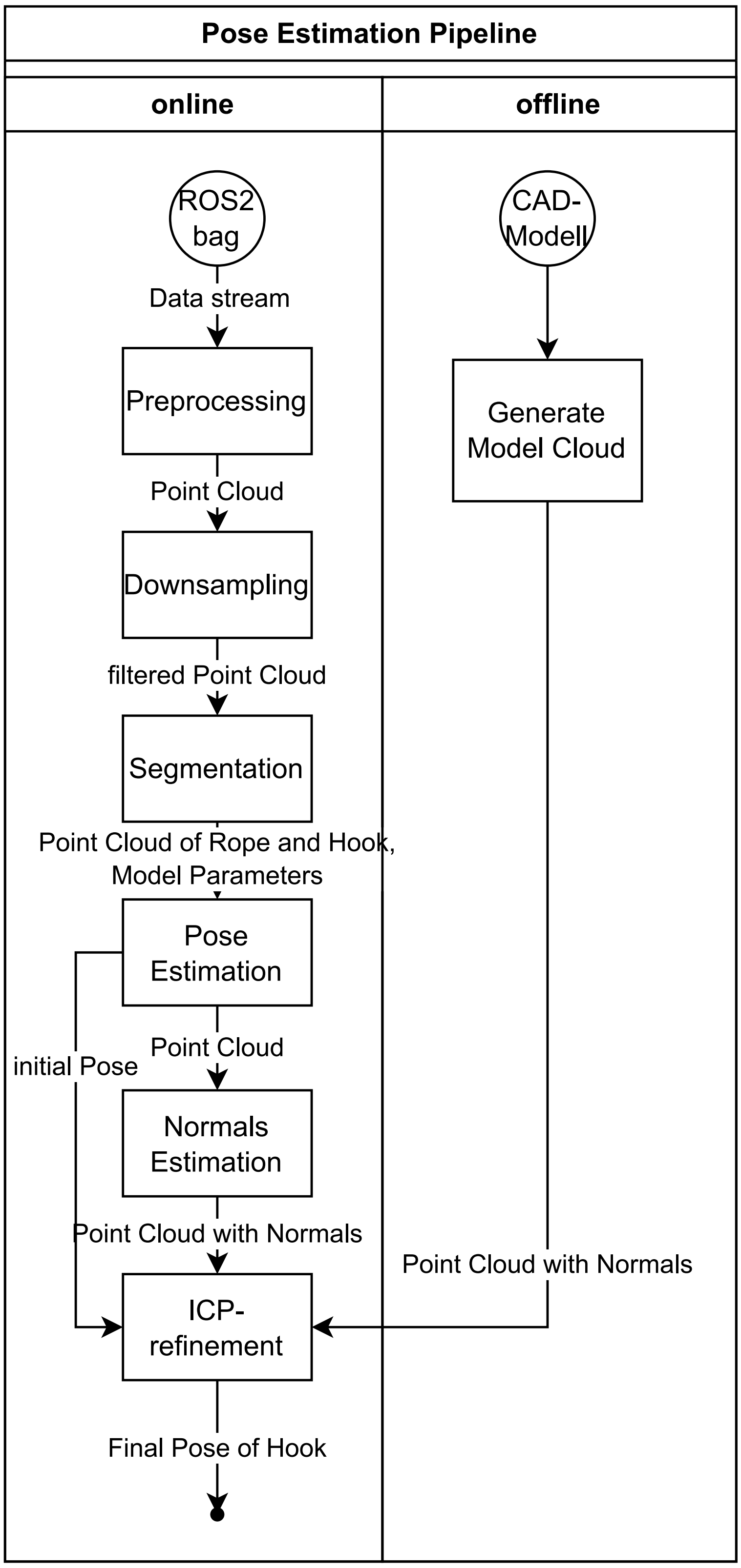


Figure 4.1: The steps of the Pose Estimation Pipeline for best effort ground truth estimation
Source: own elaboration

4.2 Preprocessing Step

The preprocessing step transforms the input stream of sensor data, received in a ROS2 standard message format, into sensible point clouds in the .pcd point cloud data format.

4.2.1 Formatting the Input Stream of Sensor Data

The LiDAR sensor requires around 100 milliseconds to cover the entire field of view with one layer of points, as mentioned in chapter 1.2.3.1. However, sensor measurements are sent every 50 milliseconds, resulting in each measurement containing only half of the area covered by the field of view of the sensor.

As previously mentioned, the sensor data is transmitted to the ACU PLC via UDP. Consequently, some measurements are lost. It is believed, that most of the measurements go missing, because of high data traffic on the test machine (LHM 500) and there not being enough time to save all the data to disk. However, this is outside the scope of this thesis.

Considering the constant movement of both the LiDAR sensor and the hook, waiting indefinitely for the second half of the point cloud isn't feasible. The hook's potential speed of approximately 1.5 meters per second sets a practical limit of 100 milliseconds for a single measurement's duration before it becomes ineffective. Thus, two consecutive sensor readings are necessary for a meaningful point cloud.

The arrival rate of new measurements may vary in the future, therefore, the preprocessing step stores the previous sensor readings with a timestamp and adds them to the current sensor reading if the time difference is less than a specified threshold (95 milliseconds). If this condition isn't met, the algorithm still attempts to find the hook in the half-point cloud. However, the success chances of pose estimation are considerably reduced even before the process begins. The implications of failed pose estimation are discussed in detail in the chapter on Filtering (5.3).

4.2.2 Downsampling of the Point Cloud using Voxel Grid Filter

The downsampling step in the pose estimation pipeline reduces the point cloud's point count. A voxelized grid approach, dividing the point cloud into volumetric pixel (voxel)s, is employed for downsampling. This not only saves computation time but is also a preparatory step for segmentation and normals estimation in Chapter 4.5.2. This procedure necessitates even point distribution across the underlying geometry's surface for optimal outcomes.

The point density of measurements on the hook acquired by the LiDAR sensor depends heavily on various factors due to the sensor's acquisition process. The distance between the object and the sensor, the surface's tilt angle towards the sensor, and the reflective properties of the surface all contribute to local point density variations. To address these factors, a voxel grid filter is applied to achieve a more uniform local point density distribution. This filter calculates the centroid of points within each voxel, limiting the filtered point cloud's maximum density to one point per voxel.

The voxel size selection lacks a definitive procedure. The lower bound is set by point cloud resolution, while the upper bound is defined by underlying geometry details. A heuristic approach is adopted, with a voxel edge length of 10 cm proving effective.

4.2.3 Conclusion

The preprocessing step effectively transforms the data stream into coherent point clouds, while even point density distribution is achieved to the best extent possible. The resulting point cloud is now ready for passing to the pose estimation algorithm. The next chapter will delve into the segmentation process.

4.3 Segmentation of the Ropes and the Hook using RANSAC

Segmentation involves partitioning a point cloud into distinct clusters, representing regions of interest, while disregarding irrelevant points. Various point cloud segmentation algorithms exist, such as euclidean cluster extraction, region growing, and min-cut based segmentation. This section focuses on using the RANSAC algorithm as described in chapter 3.1 for the segmentation task.

4.3.1 Coarse Segmentation: Filtering Uninteresting Points

The initial step of segmentation employs a straightforward approach to differentiate points belonging to the rope and hook from those outside the workspace. A cylinder encompassing the potential hook location is used to filter points within its volume (inliers, likely rope or hook points) from those outside (outliers, unrelated points). Eliminating irrelevant points at this stage improves the efficiency of subsequent point cloud operations.

However, this basic method falls short due to the crane's kinematic errors and uncertainties regarding the rope attachment points. As described in chapter 1.4, machine parameters for the rope dispatch point can vary from the true position about 1 meter. Therefore, a more sophisticated approach is required to distinguish ropes from the hook.

4.3.2 Fine Segmentation: Extracting Model Parameters

To differentiate ropes from the hook, a strategy based on detecting expected geometric primitives within the point cloud is adopted. The RANSAC algorithm is used to identify model parameters of these primitives. As it can bee seen in figure 4.5b, five lines represent the ropes, while two planes characterize the top surfaces of the hook components.

4.3.2.1 Rope Detection

The ropes can be approximated as lines, benefiting from their relatively straight nature. Under normal conditions, rope bending is minimal, external forces are insignificant and self-collision does not occur, because otherwise the crane would not be operable. The RANSAC algorithm is employed with the parameters from table 4.2 to find rope line models, and a threshold of 10 centimeters is set for inlier identification. Additionally, the models have to be aligned vertically with withing a threshold of $\pm 10\,\mathrm{deg}$

Attribute	Value	Unit
n	2	-
k	500	-
t	0.1	m
p	98	%

Table 4.2: Parameters used by the RANSAC-algorithm for line fitting

4.3.2.2 Hook Detection

The perspective from which the LiDAR Sensor views the hook varies from an almost front view (when the hook is close), to an almost top view (when the hook is far away). considering the varying viewpoint and the geometric constraints from the crane kinematics limiting the orientation of the hook, two planes emerge quickly as geometric primitives which can always be expected to be visible in the point cloud. Namely, the two top surfaces of the hook as depicted in fig: The RANSAC parameters used for extracting these planar surfaces are listed in table 4.4. Again, a model constraint, which assures that the plane normal is parallel to the vertical axis with a tolerance of ±10 degrees is used to verify the model.

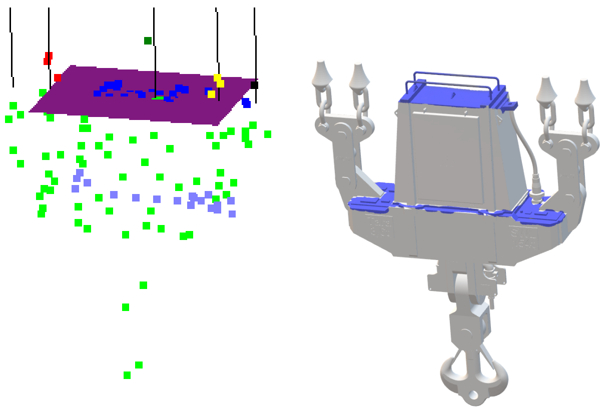


Figure 4.3: (a) Point cloud of the hook with the two planes painted in different colour. (b) CAD model with the respective areas coloured.
Source: own elaboration

Attribute	Value	Unit
n	3	-
k	500	-
κ	0.05	m
p	98	%

Table 4.4: Parameters used by the RANSAC-algorithm for plane fitting

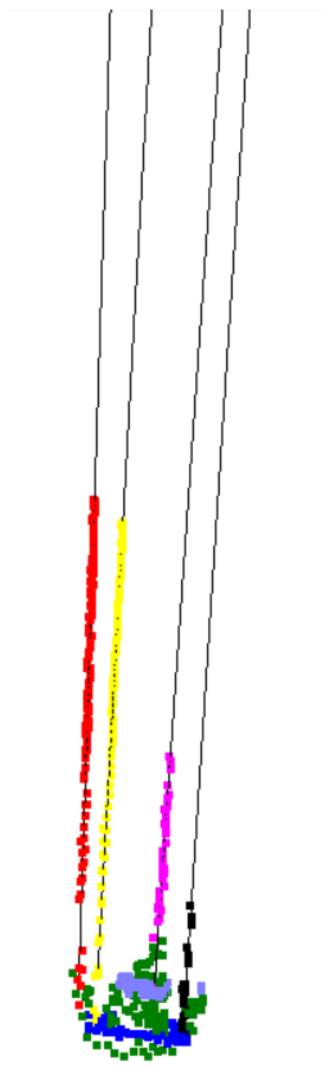
4.3.2.3 Hook and Rope Separation

Geometric primitives in the form of lines and planes are used to separate the hook from the ropes. After the rope inliers are removed from the point cloud, the two planes representing the top surfaces of the hook are located, of which the higher plane is used as separation plane. By calculating the signed point-to-plane distance, points are classified as hook or rope points based on their positions relative to the separation plane. This approach accounts for the uncertainties in the location of the rope dispatch point.

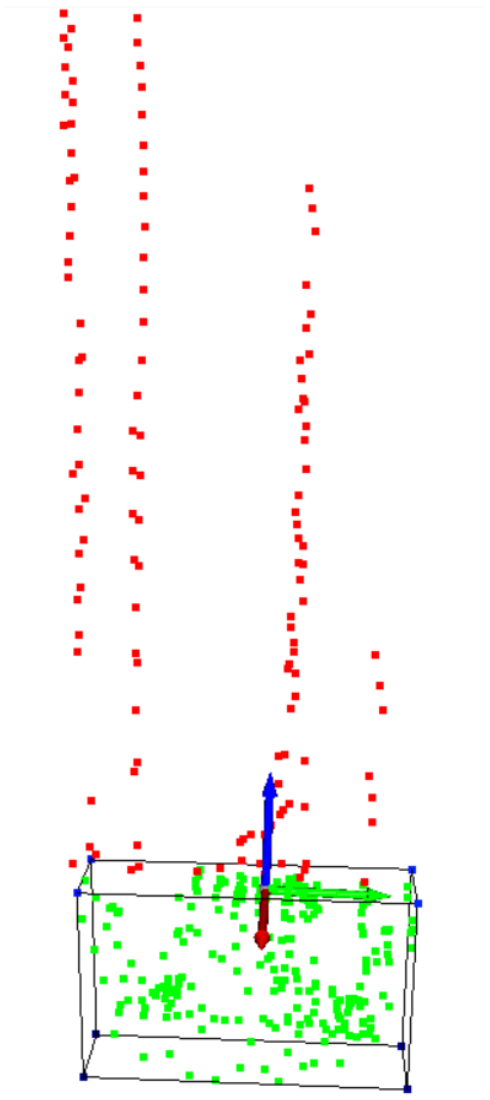
4.3.3 Outline of the RANSAC-Segmentation Algorithm

The segmentation process involves the following steps:

1. Create a copy of the original point cloud for later use.



(a)



(b)

Figure 4.5: left: The geomoetric primitives extracted from the coarsely segmented point cloud; lines are in accordance with the model parameters. right: axis-aligned bounding box encompassing the hook with coordinate frame.
Source: own elaboration

2. Extract rope points from the original cloud based on an approximate segmentation height.
3. Apply RANSAC to find rope line models and remove inliers.
4. Repeat the process for multiple ropes, removing inliers each time.
5. Locate and extract the two planes representing hook surfaces using RANSAC.
6. Choose the plane with higher inliers as the separation plane.
7. Based on signed point-to-plane distance, separate hook and rope points.
8. Return the hook points, separation plane, and rope line models.

A more detailed description of is given in algorithm 4

4.3.4 Conclusion

One could argue, that finding the ropes is actually unnecessary and only the plane which represents the lid of the rotator is needed for the separation. While that is true, the separation process is closely related to the bounding box calculation and the direction of the lines representing the ropes are essential to the bounding box which is described in the following chapter 4.4. And as mentioned before, since the ropes are a large part of the point cloud, removing them first speeds up the process of finding smaller parts later on. One drawback of this "algorithm" is, that this process is very specific to this tool, but it can be adapted to other tools by selecting suitable geometric primitives. As the points representing the hook have now been separated from the rest of the point cloud, it is possible to estimate the pose of the hook, by drawing a bounding box around the points.

4.4 Pose Estimation with the Bounding Box

The basic types of bounding boxes are described in the following sections, with comments on how suitable they are for this application.

4.4.1 Types of Bounding Boxes

There are three types of bounding boxes. Listed in order of ascending computational complexity these are:

1. the Axis-Aligned Bounding Box

Algorithm 4 RANSAC-Segmentation

```
Require: coarsely segmented point cloud $P_{\mathrm{S}}$,
    approx. segmentation height of rope and hook $h_{\mathrm{seg}}$
 1: $P_{\mathrm{S2}}$ = copy of original cloud $P_S$
 2: $P_{\mathrm{Ropes}}$ = Points of $P_S$ where $height \geq h_{\mathrm{seg}}$
 3: for $i = 1$ to 5 do
 4:   $M_{\mathrm{line},i}$ = estimate line model parameters in $P_{\mathrm{Ropes}}$ using RANSAC
 5:   remove line inliers from $P_{\mathrm{Ropes}}$
 6:   if no/too few points left in cloud then
 7:     break
 8:   end if
 9: end for
10: for all $M_{\mathrm{line}}$ do
11:   find line inliers in $P_{\mathrm{S}}$ based on model parameters $M_{\mathrm{line},i}$
12:   remove line inliers from $P_{\mathrm{S}}$
13: end for
14: for $i = 0$ to 1 do
15:   $M_{\mathrm{plane},i}$ = estimate plane model parameters in $P_{\mathrm{S}}$ using RANSAC
16:   remove plane inliers from $P_{\mathrm{S}}$
17:   if no/too few points left in cloud then
18:     this should not happen
19:     return "LiDAR measurement bad" error message
20:   end if
21: end for
22: separation plane $M_{\mathrm{plane}}$ = plane out of $\{M_{\mathrm{plane},0}, M_{\mathrm{plane},1}\}$ with higher inliers

23: for all points $p$ in $P_{\mathrm{S2}}$ do
24:   $d_p$ = signed point to plane distance
25:   if $d_p \leq 0.05$ then
26:     append $p$ to cloud containing hook points $P_{\mathrm{hook}}$
27:   else
28:     append $p$ to cloud containing rope points $P_{\mathrm{ropes}}$
29:   end if
30: end for
31: return $P_{\mathrm{hook}}, M_{\mathrm{plane}}, M_{\mathrm{lines}}$
```

2. the Oriented Bounding Box
3. the Minimal Volume Bounding Box

4.4.1.1 Axis-aligned Bounding Box

The simplest type of bounding box is the Axis-Aligned Bounding Box (AABB). The faces of the AABB are each perpendicular to one of the basis vectors. The bounds for the AABB are calculated by simply taking the maximum and minimum value of the points in the point cloud for each of the basis vector directions (x, y, z). For the purpose of encompassing the hook in this thesis, this is the bounding box that works best. In the next section, it is explained how the axes for the axis-aligned bounding box are obtained, but before that, the other methods for calculating the bounding box are described and it is explained, why they are not feasible for this application.

4.4.1.2 Oriented Bounding Box

The Oriented Bounding Box (OBB) or object oriented bounding box can be used, when the object which is to be encompassed needs its own coordinate frame. In practice, most libraries (e. g. pcl, compas, Open3d, ...) which have a method for calculating the OBB apply a Principal Component Analysis (PCA) on the point cloud for this purpose.

Fig. 4.6 shows some example snippets of the OBB where the axes have been calculated using PCA. As it can be seen, the orientation of the bounding box is not consistent with the orientation of the hook. Since the hook is almost symmetrical along two vertical planes and the segmentation is done correctly, a tight alignment of the orientation of the bounding box and the hook would be expected. though there is a visible correlation between the orientation of the hook and of the bounding box, some differences in the local point density across the surface persist despite the downsampling and shift the axes acquired by PCA into the wrong direction.

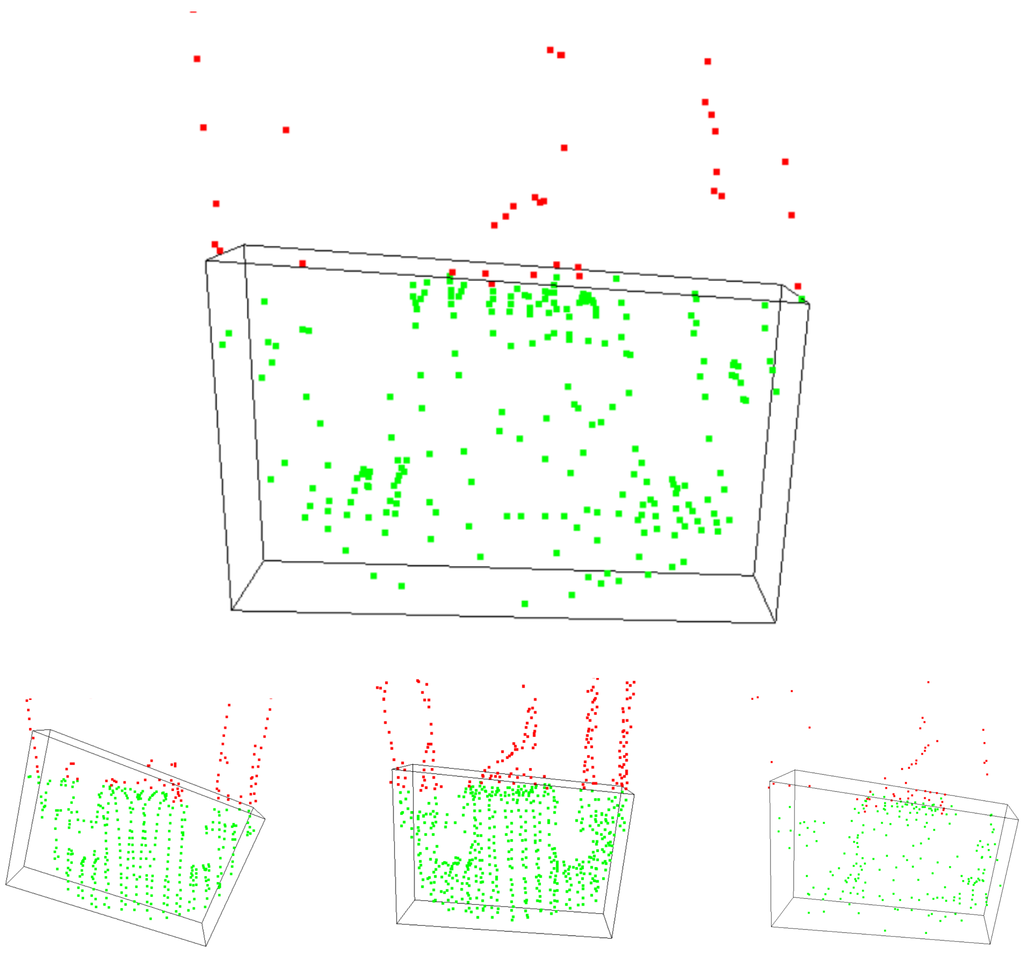


Figure 4.6: Examples of the oriented bounding box encompassing the hook. the hook points are displayed in green and the rope points are displayed in red. the lines are the edges of the bounding box. Source: own elaboration

4.4.1.3 Minimal Volume Bounding Box

The Minimal Volume Bounding Box (MVBB) is the box with the smallest possible volume that encompasses all points of a point cloud. It is calculated by first calculating the convex hull of the point cloud. According to O'Rourke, 1985, at least two edges of the convex hull are aligned with the faces of the MVBB. The MVBB is found, by comparing the bounding boxes based on all edge pairs of the convex hull and selecting the one with the smallest volume. The OBB is considered a good approximation of the MVBB in most cases, but as it was seen in the subsection before, the OBB is influenced by differences in local point densities. The MVBB is not affected by this and results are much better as can be seen in fig. 4.7. However, calculation of the MVBB takes about 100 milliseconds (in comparison AABB and OBB take less than 1 millisecond) and is therefor not feasible for the real-time approach of the pose estimation.

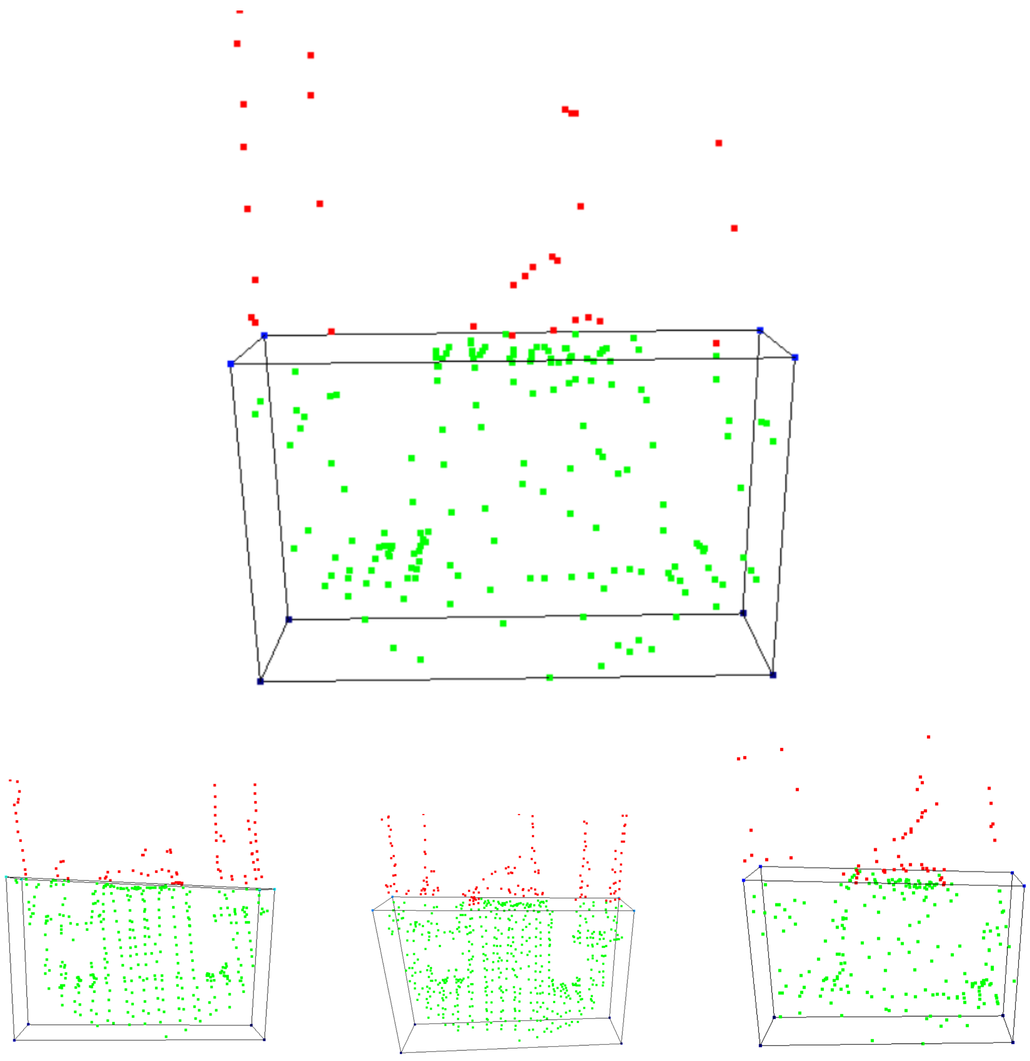


Figure 4.7: Examples of the minimal volume bounding box encompassing the hook. the hook points are displayed in green and the rope points are displayed in red. the lines are the edges of the bounding box. Source: own elaboration

4.4.2 Calculation of the Axes for the Axis-aligned Bounding Box

Since the MVBB is infeasible because computation time is too long and the OBB is infeasible because results are not accurate enough, the AABB, with pre-aligned axis, is used to estimate the hook position. This section describes how the axes for the AABB, which also directly correspond to the rotation of the hook can be calculated based on information extracted from the segmentation process in the previous chapter. Recall, that the RANSAC segmentation (algorithm 4) returns not only the point cloud P_{hook} containing the segmented hook, but also the parameters of the separation plane and the lines representing the ropes. The z-axis is simply the normal vector of the separation plane. Since the y-axis ought be perpendicular to the z-axis, it must be parallel to the separation plane. By calculating the intersection points of the lines with the separation plane and then fitting a line to the intersection points with LMSE, the y-axis is obtained by taking the direction of this line. The remaining third axis (x-axis) is calculated as the cross product of the first two axes.

4.4.3 Recovery of the Hook Pose from the Bounding Box

The axes of the AABB correspond directly to the estimated orientation of the hook. The estimation of the position of the coordinate frame of the hook as described in chapter (TBD: ref to chapter hook; write chapter hook, and mention a frame of origin for the hook) is based on parameters from the bounding box. As the coordinate frame origin of the hook was chosen to align with the separation plane from chapter 4.3.3, the position in the vertical direction (z-axis) corresponds to the upper bound of the bounding box. For the two horizontal directions (x-axis and y-axis) the center of the bounding was chosen. Since the hook is not a rigid object and has several movable parts, this method unavoidably introduces some error. However, it is still the best approach that was found to guess the initial pose. In the best effort ground truth estimation, the initial pose is refined using the ICP algorithm as described in the following two sections.

4.5 Refinement of the Pose with the generalized ICP Algorithm

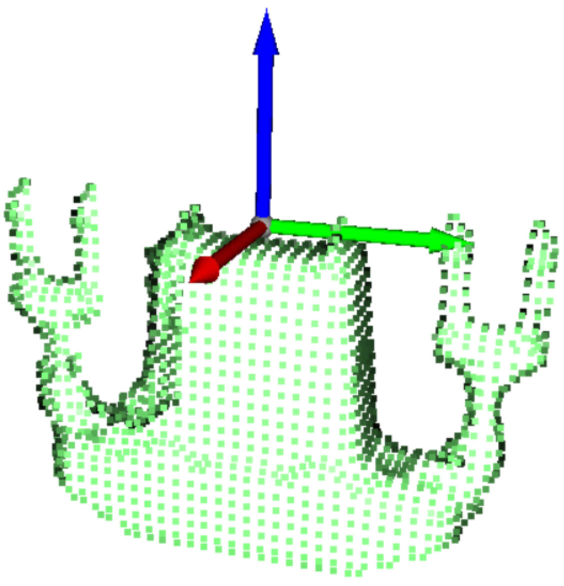
This section describes the details of the implementation of the refinement process with point cloud registration. An ICP variant, namely the generalized ICP was used for this purpose. As it was described in chapter 3.2, the generalized ICP needs a source point cloud and a target point cloud (both with point normals) and an initial transformation which transforms the source cloud in the proximity of the target. Therefore, a target point cloud is generated from a **CAD!** (**CAD!**) model of the hook and the point normals of the segmented point cloud (which serves as the source) are calculated.

4.5.1 Generating the Model Point Cloud from CAD Data

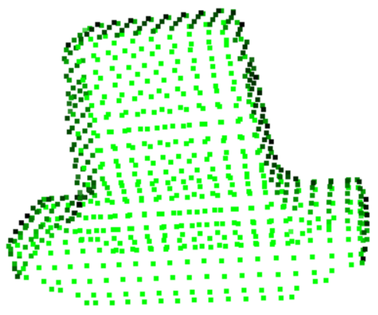
The generation of the target was done using the software "blender". Blender provides tools with which points can be sampled from the geometry with a specified density easily. It is noted, that the number of points correlates strongly with the computational effort of the ICP, so the point density is chosen low enough to fulfill the previously specified requirements for the computation time. Furthermore, the level of detail and which parts of the hook get represented in the source cloud are important decisions. Fig. 4.8 below shows the chosen model and some other source candidates which are not suitable. The model yielding the best results while still fulfilling the computation time requirements is the one depicted in 4.8a. For this model the load hook was removed and points were only sampled from the top, front and side surfaces of the CAD model. The back and bottom are not represented in the point cloud to reduce the number of points. If the resolution is too low as depicted in 4.8b, then smaller parts of the geometry of the hook cannot be represented. To represented all the details of the CAD model in the point cloud as depicted in 4.8c, roughly 22'000 points are necessary. However, this is not sensible, because the target cloud is never that detailed and the refinement takes from 10 seconds up to 1 minute.

4.5.2 Estimating the Point Cloud Normals based on the local Covariance

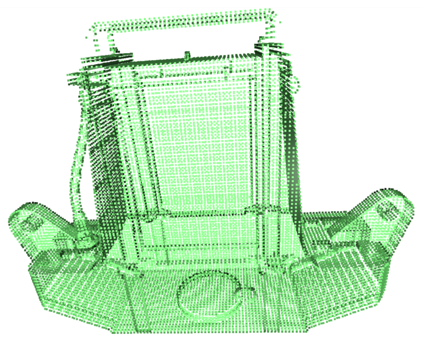
As it was described in chapter 3.2, the generalized ICP algorithm needs the point cloud normals of both the source and the target. The source normals are already provided, because the point cloud is generated by sampling the points from the surface of a CAD model. The normals of the target are estimated from the LiDAR measurement in each timestep as described in chapter 3.3. Searching for the $k = 13$ nearest neighbours has proven to yield good results. Figure 4.9 shows the point normals calculated for a target with 128 points.



(a) suitable model for source



(b) too simple model



(c) too detailed model

Figure 4.8: Source point clouds generated from CAD data. (a): best model with 1'441 points, also depicted is the origin coordinate frame; (b): a light model with very little detail and only 478 points; (c): a very detailed model of the hook with ca. 22'000 points.
Source: own elaboration

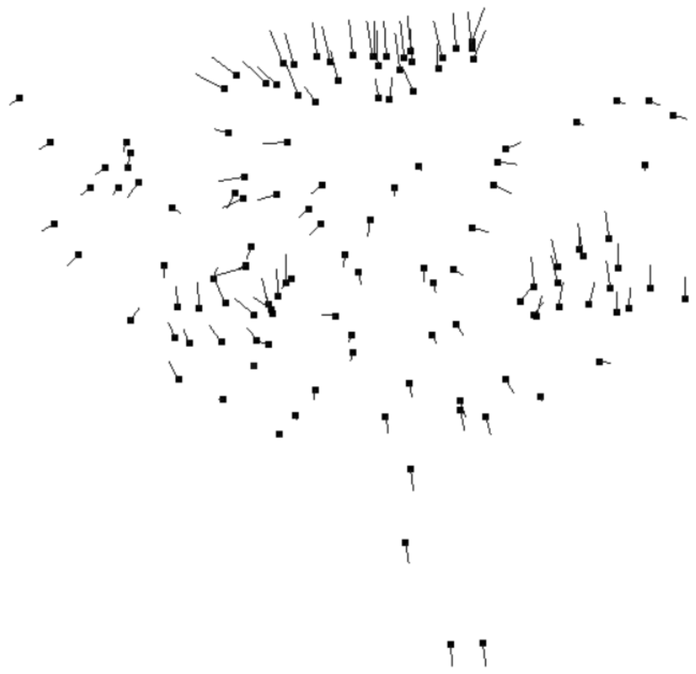


Figure 4.9: Target cloud with direction of normals depicted as lines. The cloud has 128 points.

4.6 Results of the best Effort Ground Truth Estimation

4.6.1 Evaluation of the Accuracy

The evaluation of the accuracy of the best effort ground truth estimation is based on the fitness returned from the generalized ICP algorithm. The fitness is the average residual/error of point correspondences between the source and the target cloud. As can be seen in fig. 4.10, from 861 measurements, only one measurement has an error slightly bigger than allowed to fulfill the requirements defined in section 1.5.2.1. Apart from a visual inspection, the error of the individual elements (x, y, z, roll, pitch, yaw) can not be determined. However, with the average error of 2.6 *cm* being much lower than specified in the requirements (10 *cm*), it is assumed that this result suffices as ground truth and can be used to evaluate the real-time approach.

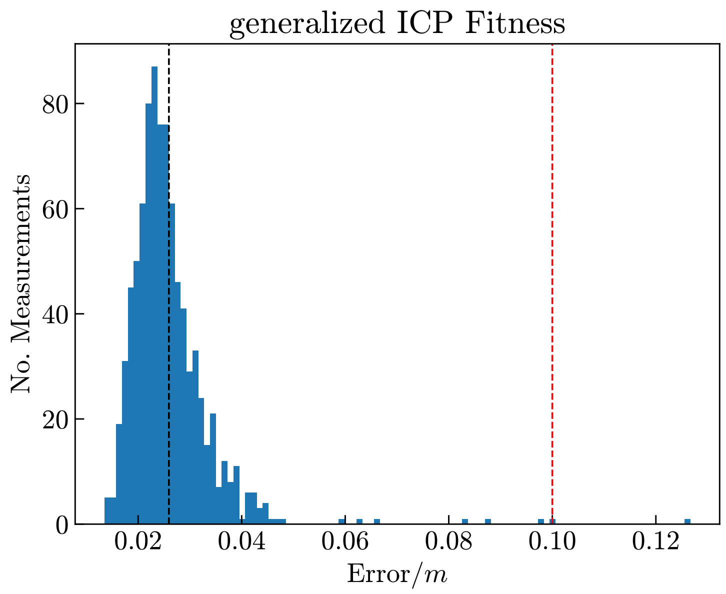


Figure 4.10: Histogram showing the fitness (avg. residual between point correspondences of source and target) of the generalized ICP algorithm. black dotted line: mean error (2.6 *cm*); red dotted line: max. error specified by requirements (10 *cm*).

4.6.2 Evaluation of the Performance

Regarding the computation time of the best effort ground truth estimation, the algorithm also fulfills the requirements specified in 1.5.2.1. Figure 4.11 shows a histogram of the computation time needed for the same 874 samples as before. The average computation time is about 700 milliseconds and all measurements except for one take less than the allowed two seconds.

4.7 Conclusion

The best effort ground truth estimation of the hook pose is the method that yields the most accurate result within the specified computation time, that was found during this endeavour. Additionally, this solution has an intrinsic evaluation of its fitness. Since this fitness suggests, that the accuracy requirements were indeed met, this result can confidently be used as benchmark to evaluate the real-time approach which is presented and evaluated in the next chapter.

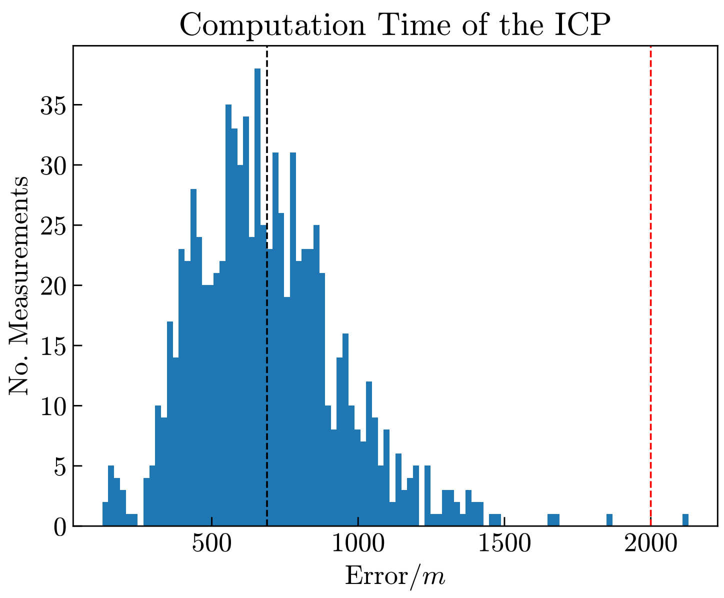


Figure 4.11: Histogram showing the computation Time of the generalized ICP Algorithm
Source: own elaboration

5 Implementation of the Pose Estimation Algorithm fulfilling the real-Time Requirements

This chapter gives a comprehensive description of the implementation of the real-time approach. In general, the real-time approach is similar to the ground truth estimation, but omitting the ICP refinement, which would not be applicable in the real-time approach due to its computational performance.

5.1 Pose Estimation Pipeline for the real-Time Approach

The pose estimation pipeline for the real-time approach is depicted in figure 5.1. As it can be seen, the steps "Preprocessing", "Downsampling", "Segmentation" and "Pose Estimation" are equivalent to the best effort ground truth estimation (refer to chapter 4.1). Even though, the impact of the downsampling is much greater on the outcome of the normals estimation than it is on the segmentation, this step is taken in the real-time approach as well because it is not computationally expensive and it helps to improve the results. Instead of the ICP refinement, a filtering step is added, which involves a verification, if the pose hypothesis is plausible and improves the initial estimation of the pose by incorporating time series data. The implemented Filter is described in the following sections.

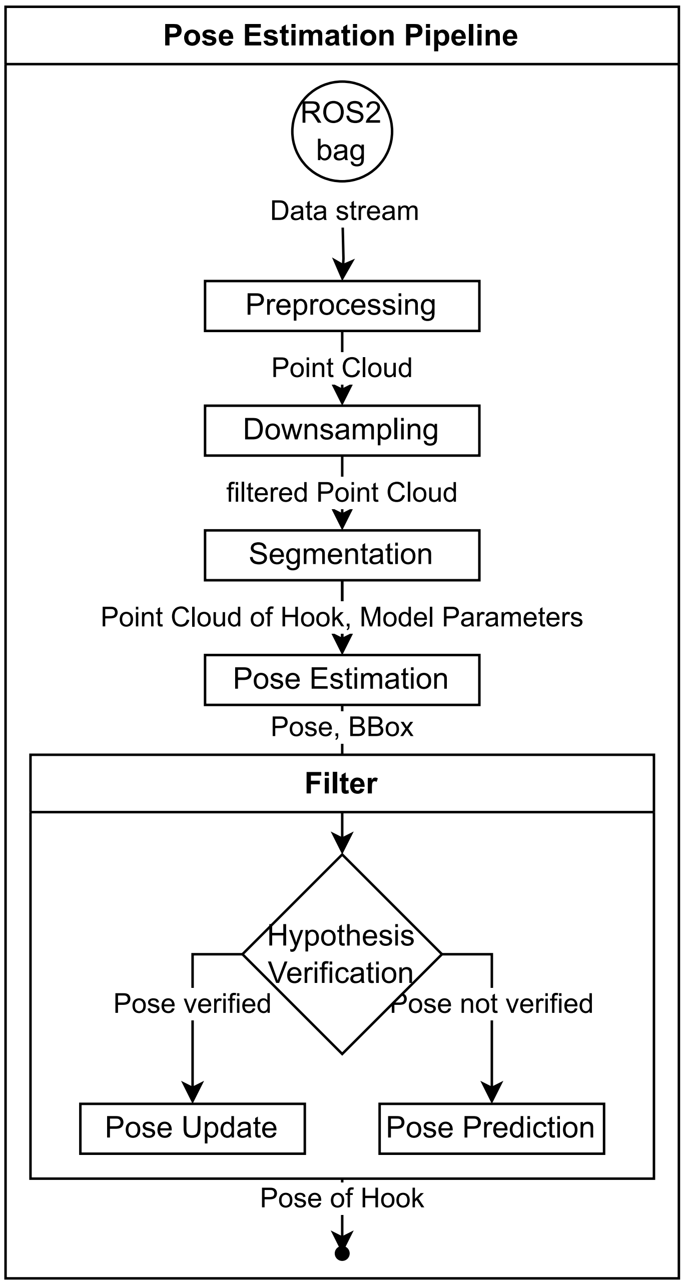


Figure 5.1: The steps of the Pose Estimation Pipeline for the real-Time Approach
Source: own elaboration

5.2 Verification of the Measurement based on the Dimensions of the Bounding Box

In the context of pose estimation, hypothesis verification refers to the process of evaluating and validating potential pose hypotheses to determine the most accurate and reliable estimation of the pose of an object. Based on additional information obtained from the pose estimation it is verified against some predefined constraints to see if the pose is valid. For this purpose, the size of the bounding box has to match with the size of the hook in order for the pose to be considered as a valid measurement. Fig. 5.2 shows the extent of the bounding box for each basis direction in the coordinate frame of the hook. The extent of each directions has to be inside the respective colored region.

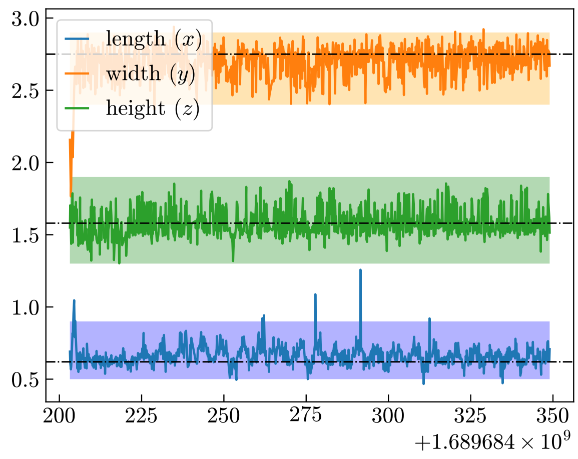


Figure 5.2: Size of the measured bounding box for each time step. The black lines represent the actual size of the hook. The colored regions mark areas for which the hypothesis is considered valid.
Source: own elaboration

If the pose hypothesis is valid, then the measurement will be passed on to the Kalman filter to update the states of the filter. If the hypothesis fails, then the pose will be estimated by the prediction of the Kalman filter for the current time.

5.3 Designing a linear Kalman Filter to track the Pose

A linear Kalman filter is used to keep track of the pose of the hook. As described in chapter 3.4, the filter finds the optimal solution for a linear system with Gaussian process and measurement noise. This chapter describes the model used to generate the prediction and how the noise variance is estimated.

5.3.1 Choosing the Reference Frame in which the States are tracked

Due to the procedure of acquiring a measurement and the crane kinematics (i. e. the tracked object is suspended by ropes), the individual elements comprising the pose all have their own characteristics. Therefore, choosing a sensible frame of reference is a crucial part of designing the Kalman filter. The coordinate frame, in which the pose of hook the is tracked shares it's origin with the base of the crane. The orientation of the tracking frame follows the slewing angle of the crane, so that the x axis is always facing towards the hook. The z axis is more or less aligned with the z axis of the hook. This way, the expected values of the orientation are constant: $\alpha = 0\,\text{deg}$, $\beta = 0\,\text{deg}$ and $\gamma = 180\,\text{deg}$.

5.3.2 Designing the States and the State Transition Matrix

A very simple mathematical model is chosen to simulate the system. given the fact, that the hooks movement is changing relatively slow compared to the sampling rate of the measurements, the velocity and angular velocity of the hook does not change significantly between two consecutive measurements. A constant velocity model for the hook is therefore assumed. Furthermore, it is assumed, that the individual elements of the hook pose $(x, y, z, \alpha, \beta, \gamma)$ are independent of each other. Based on these assumptions, the state vector $\mathbf{x}$ is defined as

$$\mathbf{x} = (x, \dot{x}, y, \dot{y}, z, \dot{z}, \alpha, \dot{\alpha}, \beta, \dot{\beta}\gamma, \dot{\gamma}) \tag{5.1}$$

, where the states x, y, z, α, β, γ are tracking the pose of the hook and $\dot{x}$, $\dot{y}$, $\dot{z}$, $\dot{\alpha}$, $\dot{\beta}$, $\dot{\gamma}$ are hidden states tracking the velocity and angular velocity of the hook. Since the elements of the hook pose are independent of each other, the state transition matrix can be derived for one of the elements, and the other follow accordingly. Consider the state space representation of a one dimensional constant velocity model with no input:

$$\dot{\mathbf{x}} = A\mathbf{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{5.2}$$

Using discrete forward Euler integration gives the state transition matrix **F**:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}_t \Delta t = \mathbf{x}_t + A\mathbf{x}_t \Delta t = (I + A\Delta t)\mathbf{x}_t \tag{5.3}$$

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} \tag{5.4}$$

Since the other elements of the pose follow accordingly, the state transition matrix for the system is

$$\mathbf{F} = \begin{bmatrix}
1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix} \tag{5.5}$$

The measurement matrix **H** transforms the predicted state into the measurement space to obtain the measurement prediction. Since the states of the filter correspond directly to the pose of the hook, designing the measurement matrix is straightforward.

$$\mathbf{H} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix} \tag{5.6}$$

5.3.3 Estimating the Noise Covariance Matrices

5.3.3.1 Measurement Noise Covariance

The measurement noise covariance **R** represents the uncertainty of the measurement. A measurement in this context is the estimated pose of the hook. Based on the assumption, that the error from the best effort ground truth estimation is much smaller than the error from the bounding box pose estimation, the real noise can be approximated by the difference of these two. For this purpose figure 5.3 shows the error of the individual elements of the pose calculated this way. It can be seen, that the measurement noise is sufficiently Gaussian distributed to expect good results from the Kalman filter.

As it can be seen from the results in the figures above, the bounding box pose estimation already satisfy the requirements specified for the real-time approach in chapter 1.5.2.2.

Table 5.4 shows the expected values and standard deviations for the individual coordinates.

From these errors, the measurement noise covariance matrix **R** is calculated according to equation

$$\mathbf{R} = \frac{1}{N}\sum_{i=1}^{N}((\mathbf{e_i} - \bar{\mathbf{e}}) \cdot (\mathbf{e_i} - \bar{\mathbf{e}})^T) \tag{5.7}$$

Where:

N is the number of measurements/pose estimations

$\mathbf{e_i}$ is a vector containing the errors for each coordinate of measurement i

$\bar{\mathbf{e}}$ is the mean error/expected error of all measurements

$$\bar{\mathbf{e}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{e_i}$$

which yields

$$\mathbf{R} = \begin{bmatrix} 4.4\text{e-}04 & 7.8\text{e-}06 & -9.0\text{e-}05 & 9.9\text{e-}06 & -1.2\text{e-}05 & 1.8\text{e-}05 \\ 7.8\text{e-}06 & 2.8\text{e-}03 & -8.7\text{e-}05 & 1.9\text{e-}06 & -8.0\text{e-}06 & -3.4\text{e-}04 \\ -9.0\text{e-}05 & -8.7\text{e-}05 & 9.9\text{e-}04 & 1.1\text{e-}05 & 2.0\text{e-}05 & 3.9\text{e-}05 \\ 9.9\text{e-}06 & 1.9\text{e-}06 & 1.1\text{e-}05 & 1.0\text{e-}04 & -1.9\text{e-}06 & 2.7\text{e-}04 \\ -1.2\text{e-}05 & -8.0\text{e-}06 & 2.0\text{e-}05 & -1.9\text{e-}06 & 7.2\text{e-}06 & -6.3\text{e-}06 \\ 1.8\text{e-}05 & -3.4\text{e-}04 & 3.9\text{e-}05 & 2.7\text{e-}04 & -6.3\text{e-}06 & 8.0\text{e-}04 \end{bmatrix} \tag{5.8}$$

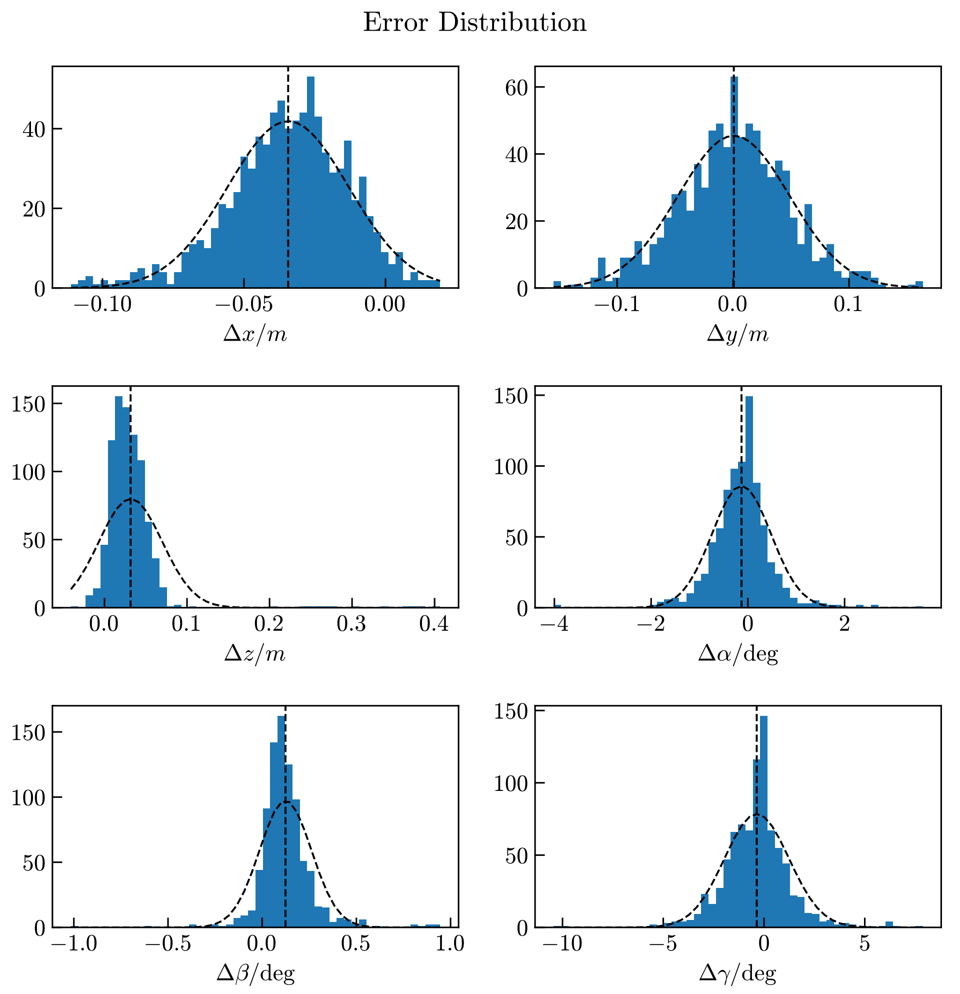


Figure 5.3: Measurement Noise of individual coordinates
Source: own elaboration

Coordinate	Expected Value	Standard Deviation	Unit
x	-0.032	0.021	m
y	0.001	0.053	m
z	0.028	0.031	m
α	-0.113	0.585	deg
β	0.124	0.154	deg
γ	-0.313	1.622	deg

Table 5.4: Standard deviations and expected values for error of bounding box

5.3.3.2 Process Noise Covariance

The process noise covariance $\mathbf{Q}$ represents the uncertainty of the model used by the Kalman filter. It can not be calculated as easily as the measurement noise covariance. Though there are methods for deriving $\mathbf{Q}$, going into detail on this topic is beyond the scope of this thesis. For practical purposes, discrete white noise is assumed for the process noise. As described in [Labbe Jr, 2023, p. 241], discrete white noise for a first order system where $\mathbf{x} = (x, \dot{x})$ can be approximated by

$$\mathbf{Q} = \mathbf{G}\mathbf{G}^T v = \begin{bmatrix} \frac{1}{4}dt^4 & \frac{1}{2}dt^3 \\ \frac{1}{2}dt^3 & dt^2 \end{bmatrix} \tag{5.9}$$

Where:

$$\mathbf{G} = \begin{bmatrix} \frac{1}{2}dt^2 & dt \end{bmatrix}^T \text{ is the process noise per time step and}$$
$$v \text{ is the variance of the noise}$$

The variance v is obtained empirically by trying different values until the results looks as expected. As it can be seen in equation 5.8, variance of the individual coordinates ranges from $var_\beta = 7.2e-06$ to $var_y = 2.8e-03$. Therefore, the measurement noise variance is used as a reference value and scaled with a constant factor to estimate the process noise variance. Since a constant velocity model is used for the prediction, it is expected that the more trust (more trust corresponds to lesser uncertainty) is put into the model, the smoother the result will get, but if too much trust is put into the model, the filter result will diverge from the measurement when the velocity changes. The figures below show the filter output for different process noise covariance matrices compared to the unfiltered bounding box.

Fig. 5.5 shows the x-coordinate of the hooks position. In this example, the hook has a relatively high velocity and swings with an amplitude of $10m$ back and forth. It can be seen, that with low process noise variance, the filtered position overshoots approx. $50cm$ at the peaks.

figure 5.6 depicts the tool height over time. This time series contains two bad measurements at time $t = 54s$ and $t = 63.5s$. The Kalman filter reduces the influence of these outliers with decreasing scale for $\mathbf{Q}$.

Fig. 5.7 shows the orientation γ about the z-axis.

A value of 40 is chosen for the scale of the process noise covariance. It is noted, that there is a linear correlation between the variance of the process noise and the measurement noise, but a scale of 1 does not mean that the variance of the

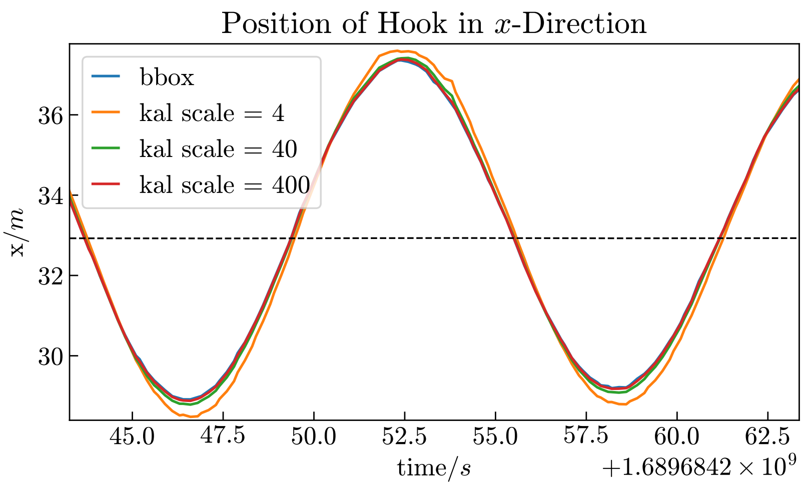


Figure 5.5: x-coordinate for different process noise covariance matrices $\mathbf{Q}$. The dotted line represents the rope dispatch point at the boom tip of the crane received as machine parameter.
Source: own elaboration

process noise and the measurement noise are equal. Since the process noise is also time dependent as described in 5.9, $\mathbf{Q}$ may vary depending on the step size. The covariance for the individual elements of the state vector of the process noise for a step size of 100 milliseconds is therefore:

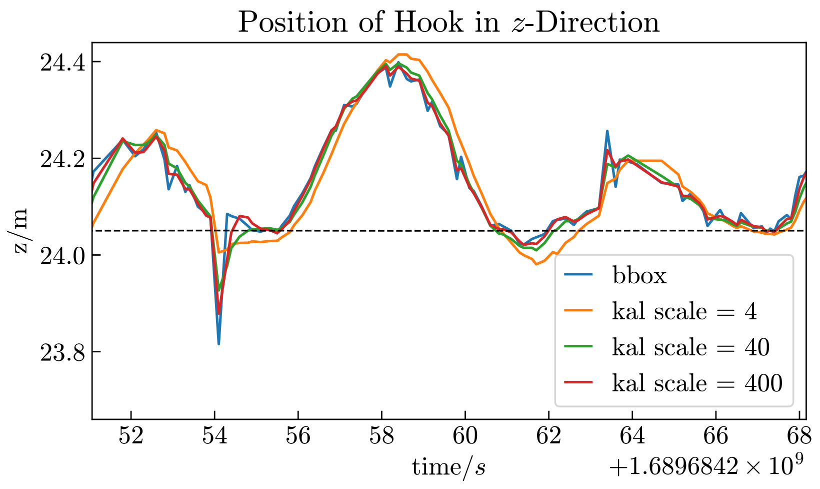


Figure 5.6: z-coordiante for different process noise covariance matrices $\mathbf{Q}$. The dotted line represents the hook height received as machine parameter
Source: own elaboration

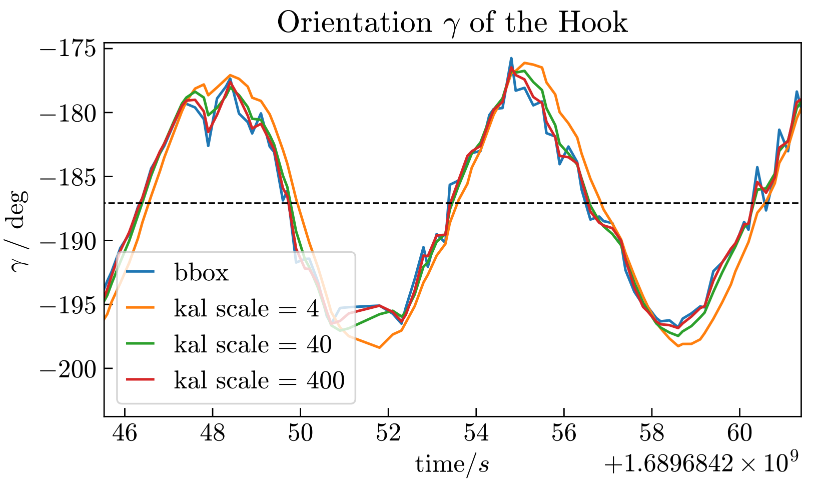


Figure 5.7: Orientation γ for different process noise covariance matrices $\mathbf{Q}$. The dotted line represents the hooks orientation received as machine parameter
Source: own elaboration

$$\mathbf{Q}_x = \begin{bmatrix} 4.47\text{e-}07 & 8.95\text{e-}06 \\ 8.95\text{e-}06 & 1.79\text{e-}04 \end{bmatrix}, \mathbf{Q}_y = \begin{bmatrix} 2.81\text{e-}06 & 5.62\text{e-}05 \\ 5.62\text{e-}05 & 1.12\text{e-}03 \end{bmatrix}, \mathbf{Q}_z = \begin{bmatrix} 9.90\text{e-}07 & 1.98\text{e-}05 \\ 1.98\text{e-}05 & 3.96\text{e-}04 \end{bmatrix},$$
$$\mathbf{Q}_\alpha = \begin{bmatrix} 1.04\text{e-}07 & 2.08\text{e-}06 \\ 2.08\text{e-}06 & 4.17\text{e-}05 \end{bmatrix}, \mathbf{Q}_\beta = \begin{bmatrix} 7.25\text{e-}09 & 1.45\text{e-}07 \\ 1.45\text{e-}07 & 2.90\text{e-}06 \end{bmatrix}, \mathbf{Q}_\gamma = \begin{bmatrix} 8.00\text{e-}07 & 1.60\text{e-}05 \\ 1.60\text{e-}05 & 3.20\text{e-}04 \end{bmatrix}$$

$\mathbf{Q}$ is simply the block diagonal matrix of $\{\mathbf{Q}_x, ..., \mathbf{Q}_\gamma\}$.

5.4 Results of the real-Time Approach

The results of the real-time approach regarding the Accuracy and performance in terms of computation time are presented in this chapter.

5.4.1 Evaluation of the Accuracy

Figure 5.8 shows histograms of the errors of the individual coordinates for the bounding box pose estimation and the filtered poses. The errors are calculated as the transformation between the best effort ground truth estimation and the respective pose from the real-time approach. The angles are extrinsic xyz-Euler angles. It can be seen, that the calculated errors have not improved at all. Especially at high velocity (as can be seen in fig 5.8 top left), the filter with the constant velocity model does not perform well.

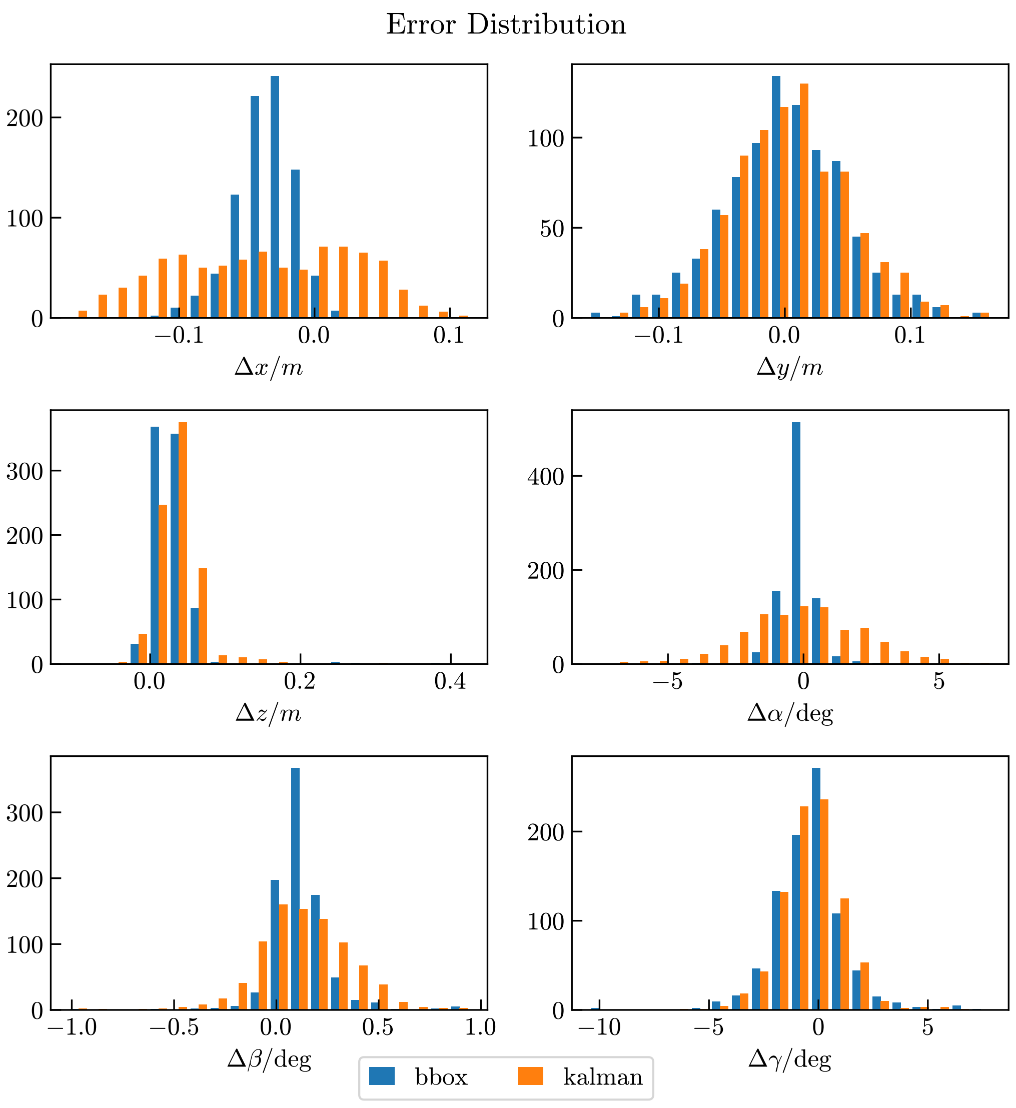


Figure 5.8: Histograms of errors from Kalman filter of individual Coordinates. The y-axis counts the number of measurements in the respective bin.
Source: own elaboration

5.4.2 Evaluation of the Performance

The whole pipeline from the arrival of the measurement to the filtered pose currently takes about 15 to 20 milliseconds, which meets the specified requirements for the real-time approach. Special attention has to be given to the steps, where RANSAC is deployed because the computation time of this algorithm is subject to strong fluctuations. Figure 5.9 shows a box-plot diagram of the computation times for the individual geometric primitives which are extracted from the point cloud. The boxes were drawn based on the results from 902 sensor readings. It can be seen, that all calls to the RANSAC algorithm return within less than 0.5 milliseconds. Also, computation takes the longest for the first line, and subsequently gets shorter as more points are extracted from the point cloud. It is concluded, that the computation time of the RANSAC algorithm, even though subject to strong fluctuations, does not contribute significantly to the overall computation time and it is therefore safe to deploy it in the real-time approach.

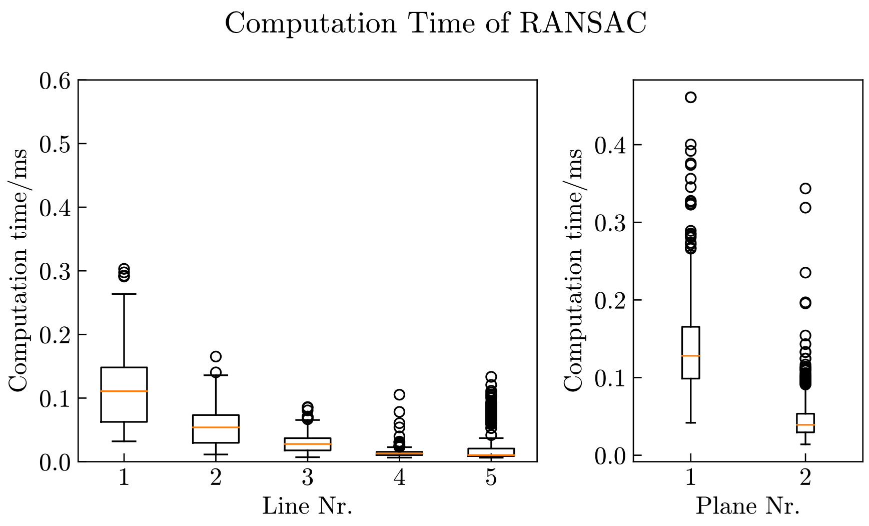


Figure 5.9: Computation times of the individual geometric primitives of the RANSAC segmentation for 902 measurements. left: Computation times needed for the lines; right: Computation times needed for the planes
Source: own elaboration

5.5 Conclusion

It is remarkable, that the RANSAC algorithm is magnitudes faster when deployed to search for geometric primitives rather than the pose of an object. As mentioned in chapter 2.2.1, feature matching takes about 300 milliseconds on average, whereas the detection of a line or plane is achieved in less than 0.5 milliseconds. This is mainly due to the efficieny of operations on geometric primitives, which can be described in closed form with methods of linear algebra.

For the evaluated test data, the real-time approach fulfills the specified requirements. The accuracy of the initial pose estimated utilizing the AABB is higher than the filtered pose, which leads to the conclusion, that the constant velocity model of the Kalman filter is poorly chosen. Nevertheless, the filter has advantages justifying its use. Firstly, the influence of outliers is greatly reduced as can be seen in fig. 6.5. Secondly, the velocity of the hook is tracked, consequently enabling the prediction of poses at arbitrary times, even when the measurement is missing or the pose estimation fails.

6 Disscusion of the Results

Results of the most important parts were already presented as necessary across the thesis. This chapter presents the results of a test drive where data has been recorded for 146 seconds. The outreach of the crane in this example is roughly 40 meters and the tool height is about 20 meters, which corresponds to a medium outreach and medium tool height. The crane performs some movement in the beginning of the sample, and then stops, so that a free pendulum motion can be observed in the sample.

6.1 Results of the Preprocessing

In the recorded test drive, point cloud data arrives at intervals of 100 milliseconds. Fig. 6.1 shows the missing measurements of the recorded test drive exemplary for a duration of 10 seconds. It can be seen, that only 59 of the expected 100 measurements were managed to be recorded. The test drive recorded data for 146 seconds which corresponds to 1 460 point clouds. However only 902 point clouds were saved to disk and can be played back in the simulation.

6.2 Success Rate of the Bounding Box Pose Estimation

Out of the 902 Measurements that were fed to the simulation, 861 successful pose estimations remain after the verification step of the filter, which corresponds to a success rate of 95 percent.

6.3 Comparison of the best Effort Ground Truth Estimation and the real-Time Approach

The following figures (fig. 6.2 to 6.9) show the results of the pose estimation for a recorded test data set. The figures depict the estimated ground truth (ICP), the bounding box pose estimation and the filtered pose over time for the individual

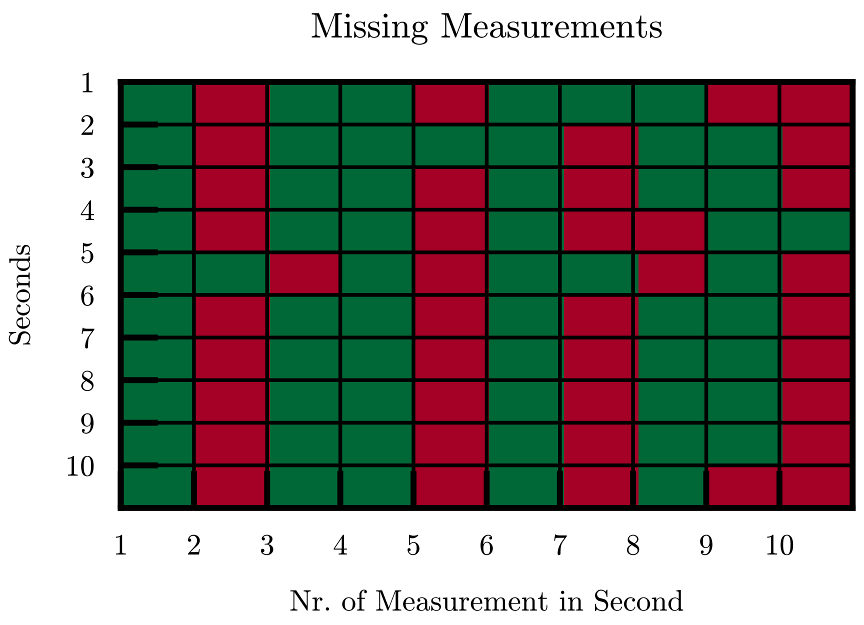


Figure 6.1: Missing measurements for a duration of 10 seconds which corresponds to 100 time steps. Of the 100 expected measurements, 41 are missing.
Source: own elaboration

coordinates $x, y, z, \alpha, \beta,$ andγ in the base coordinate frame. The crane is moving in the beginning and then stops at time $t = ...240s$.

Figure 6.2 shows the x coordinate of the hook. Due to the scale of the of the y-axis and the high quality of the results, no distinction between the different methods can be made. For more detail, fig. 6.3 shows the same graphs zoomed in on one of the peaks. It can be concluded, that these results are very good.

Figure 6.4 shows the y-coordinate of the hook. Since the pendulum motion is clearly visible with an amplitude of about two to three meters and all three methods overlap, it is concluded, that these results are also very good.

Figure 6.5 shows the z-coordinate of the hook (corresponding to the tool height). The dotted line has been adjusted manually with an offset to fit the data. A few wrong measurements can be seen from the real-time approach, however, the filter drastically reduces the impact of these errors.

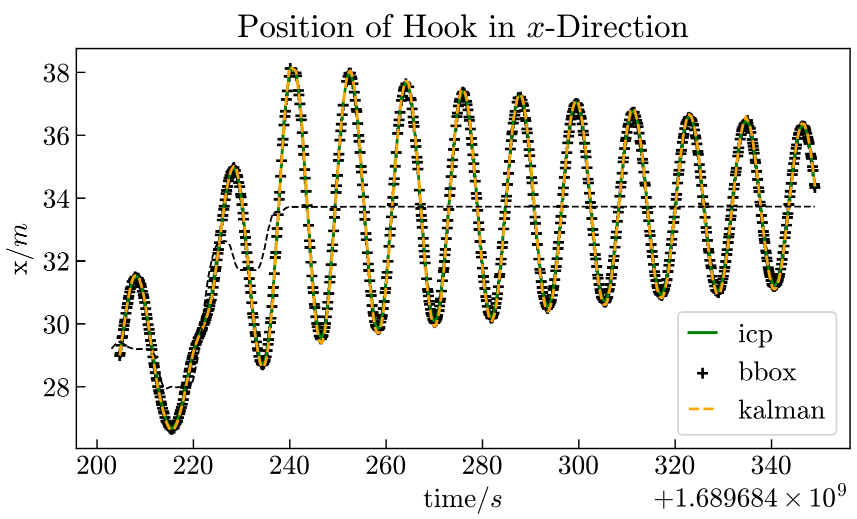


Figure 6.2: Position of Hooks x-coordinate with different methods. Source: own elaboration

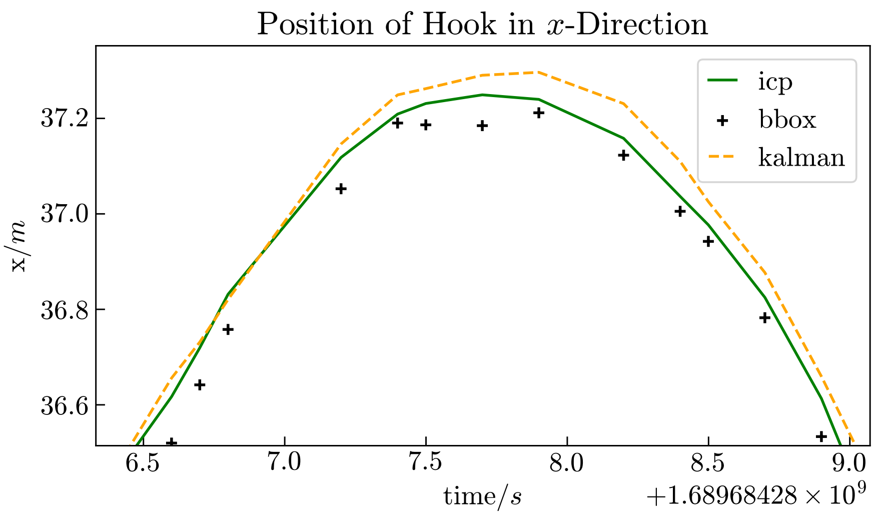


Figure 6.3: Position of hooks x-coordinate with different methods (zoomed in). Source: own elaboration

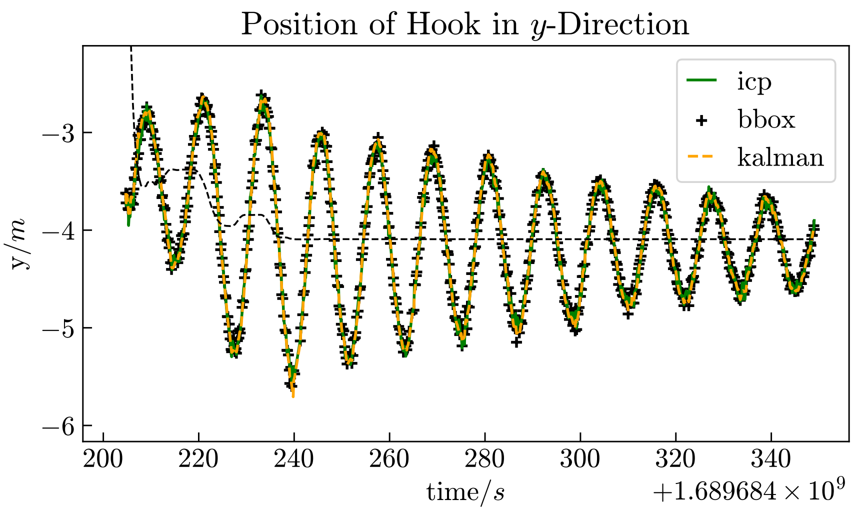


Figure 6.4: Position of hooks y-coordinate with different methods
Source: own elaboration

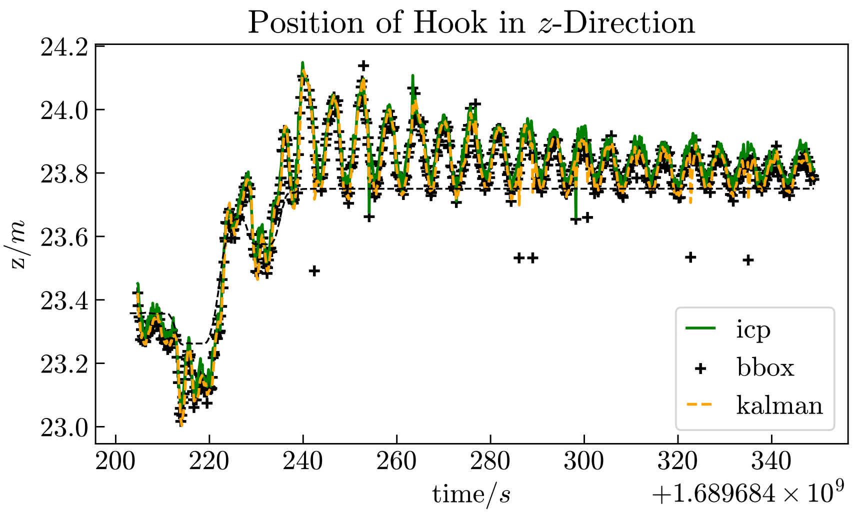


Figure 6.5: Position of hooks z-coordinate with different methods
Source: own elaboration

Figure 6.6 shows the α coordinate of the hook corresponding to the rotation about the x-axis of the base coordinate frame. A rotation about this axis is effectively prevented by the way the hook is suspended, so it would be expected that this angle is always very close to zero and it is safe to assume, that all deviations from zero are actually errors. Fig. 6.7 zooms in on the orientation α and reveals a tight alignment of the bounding box and ICP results. This error is likely due to the geometry of the hook. The hooks extent in the z and y directions is roughly 2 meters, whereas the extent in the x direction is only about 0.5 meters. When the points are spread further apart along an axis a more accurate result for the angle can be retrieved than with narrowly spread points with the same amount of noise. Nevertheless, the error is still smaller than the threshold of 10 deg specified by the requirements, especially after the filter is applied.

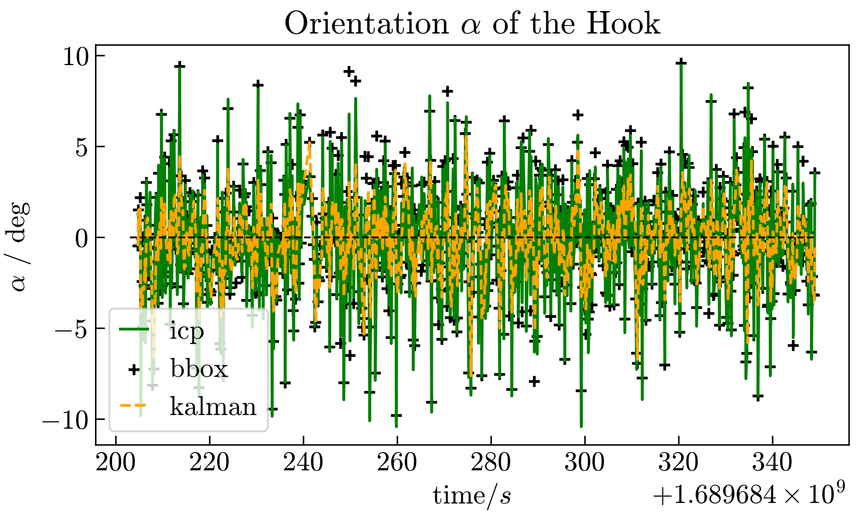


Figure 6.6: Position of Hooks α Coordinate with different methods
Source: own elaboration

Figures 6.8 and 6.9 shows the β and γ coordinate of the hook corresponding to the rotation about the y- and the z-axis of the base coordinate frame respectively. These results look plausible.

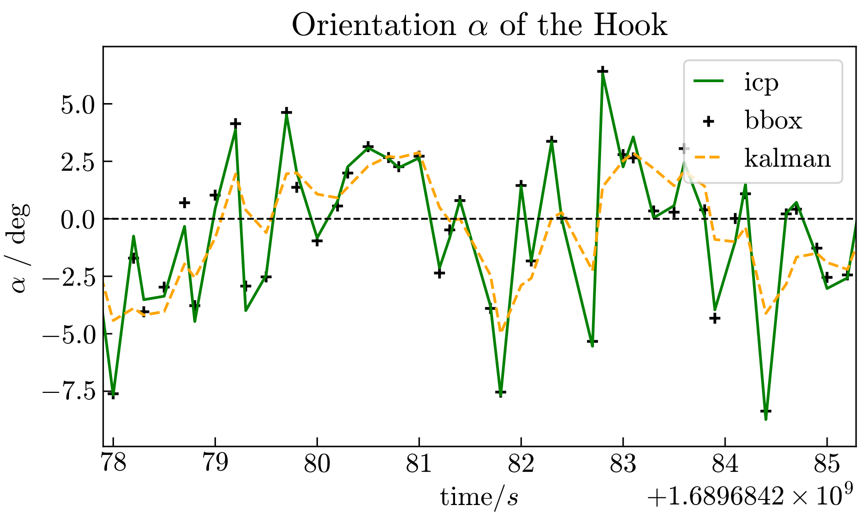


Figure 6.7: Position of Hooks α Coordinate with different methods
Source: own elaboration

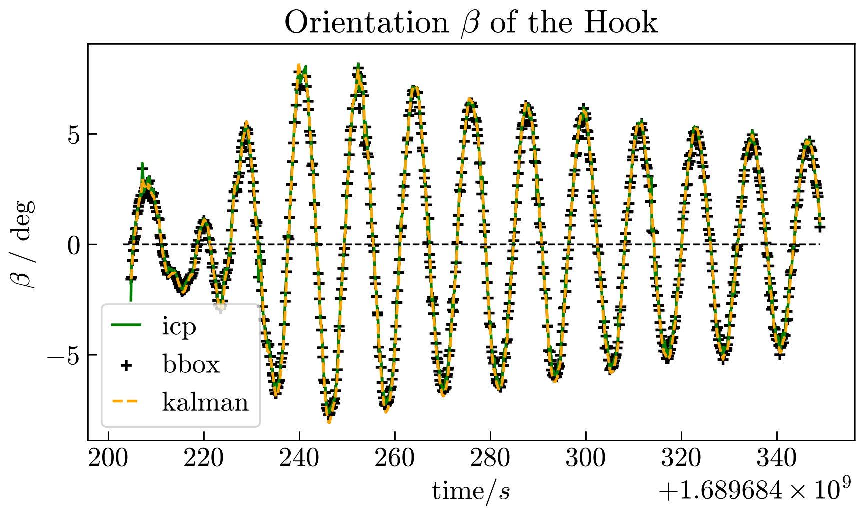


Figure 6.8: Position of Hooks β Coordinate with different methods
Source: own elaboration

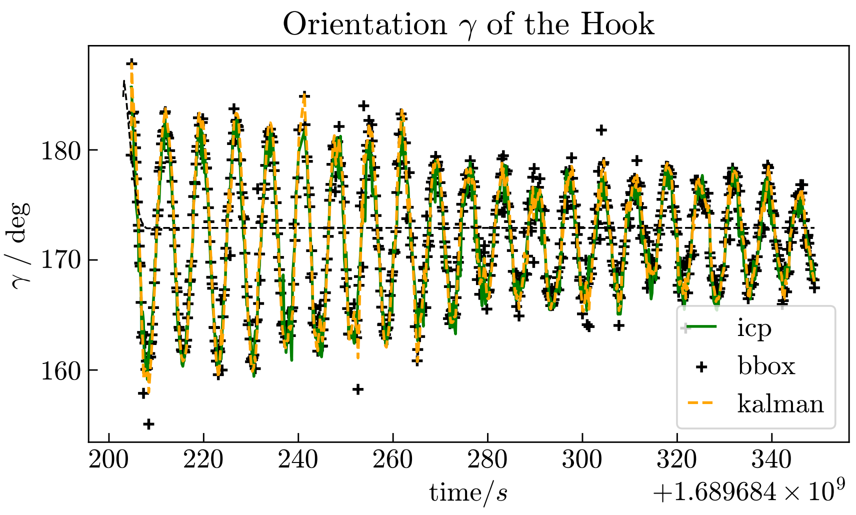


Figure 6.9: Position of Hooks γ Coordinate with different methods
Source: own elaboration

6.4 Comparison of the real-Time Approach with the State-of-the-Art

This section compares the results of the real-Time capable approach with the state-of-the-art "Load Position Observer" as described in chapter 1.3. While the trajectories drawn by the different methods are similar in shape, a considerable offset off about 40 centimeters along with a slight phase shift can be observed between the LPO and the LiDAR based measurement.

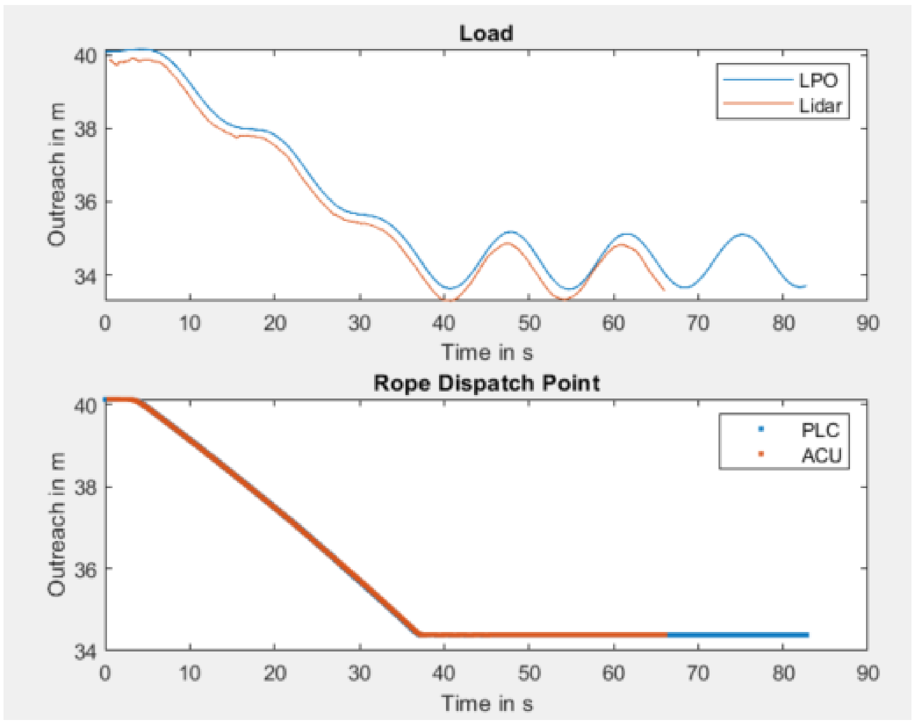


Figure 6.10: Comparison of the real-time approach (Lidar; orange line) with the state-of-the-art (LPO, blue line).
Source: with the kind permission of Liebherr

6.5 Visualization of the Results

Figure 6.11 shows a model of the hook projected to the image of one of the cameras. Figure 6.12 shows the hook model and the point cloud in the ROS2 simulation visualized with rviz. The pose for these visualizations was estimated by the real-time approach.

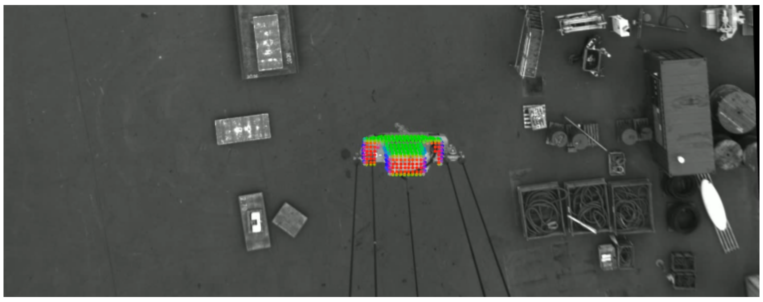


Figure 6.11: The hook model projected onto the camera image
Source: own elaboration

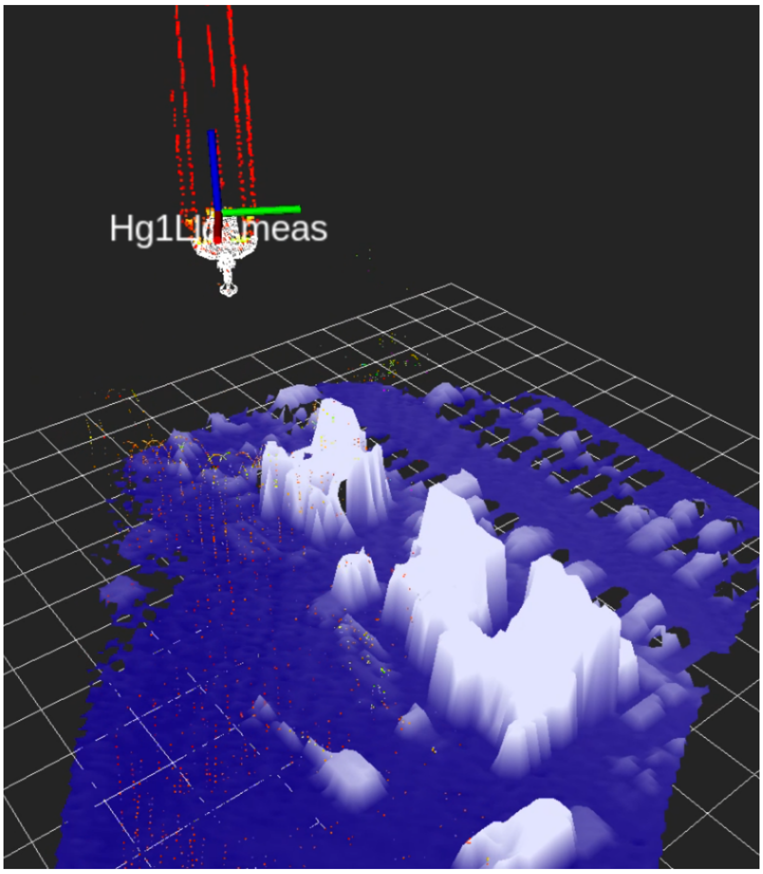


Figure 6.12: [The Hook Model and the Point Cloud visualized in rviz.
Source: own elaboration

7 Conclusion

A method for the pose estimation of a rope-guided tool from optical sensors was presented in this thesis. The main difficulties of this endeavour were the sparsity of data points in the cloud, when the hook is relatively far away from the sensor and the requirements regarding the computation time of the pose estimation. These difficulties were tackled by omitting the calculation of descriptors. The real-time requirements were met by efficiently deploying RANSAC in the segmentation process to extract geometric primitives, rather than building computationally expensive correspondences in the feature space for pose estimation. The pose of the hook based on point cloud data, that is too sparse for feature matching was estimated by encompassing the hook points with a bounding box and leveraging constraints from the crane kinematics and information extracted during the segmentation process. While this proposed approach still has its weaknesses, and there might be other approaches that were not considered yet, it was shown, that the point clouds acquired with a LiDAR sensor can be used to estimate the pose of the hook fast and accurate enough to be used in real-time critical applications, therefore enabling its use in the development of autonomous or semi-autonomous cranes. Limitations and parts, where there is still room for improvement are mentioned in the next chapter.

8 Future Work and Limitations

The method for pose estimation presented in this thesis, while performing well on the evaluated test data, is highly specific and will probably not find widespread usage in the field of pose estimation/3D object recognition from point cloud data. Its capability is limited to objects, that are suspended by multiple ropes and have distinctive surfaces that can be approximated with geometric primitives.

Improve the System Dynamics of the Kalman Filter:
The mathematical model with constant velocity used in the Kalman filter has proven to be a poor choice for the dynamics of the system. A more sophisticated model should replace the current model in the filter. The new model should incorporate acceleration and allow for the simulation of the pendulum motion of the hook. If the improved model is more suitable to simulate the system dynamics, more confidence can be put into the prediction of the states by the Kalman filter, therefore significantly reducing the noise imposed by the measurements in the filtered pose.

Evaluate more realistic tests:
The tests were all done with no load attached to the hook and with an unobstructed view on the tool. More data from test drives with attached load and/or obstacles in the vicinity of the workspace should be recorded and evaluated. The coarse segmentation of the workspace and the ropes/hook may have to be adapted to consider these scenarios.

Refine Result of Ground Truth Estimation with Post Processing:
The best effort ground truth estimation has considerable high frequency noise in the result, especially the *alpha* coordinate (rotation about the x-axis) as depicted in fig. 6.6. Since it is known, that due to the high inertia of the system, the hook cannot perform considerable movements at high frequencies, it was concluded, that this noise is a measurement error. Since the results of the best effort ground estimation do not need to be readily available for decision making, the data could be further processed. A post processing refinement of this data should be applied to smooth the result and therefore reduce the error. This would in turn allow for a more accurate evaluation of the real-time capable approach and other methods. If the improvements mentioned above are implemented and the evaluation of more realistic tests has been successful, then this method could be used to detect the tool pose for autonomous material handling machinery.

Bibliography

Aldoma, A., Marton, Z.-C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R. B., Gedikli, S., & Vincze, M. (2012). Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation. *IEEE Robotics & Automation Magazine*, *19*(3), 80–91. https://doi.org/10.1109/MRA.2012.2206675

Drost, B., Ulrich, M., Navab, N., & Ilic, S. (2010). Model globally, match locally: Efficient and robust 3D object recognition. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 998–1005. https://doi.org/10.1109/CVPR.2010.5540108

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395. https://doi.org/10.1145/358669.358692

Holz, D., Ichim, A. E., Tombari, F., Rusu, R. B., & Behnke, S. (2015). Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D. *IEEE Robotics & Automation Magazine*, *22*(4), 110–124. https://doi.org/10.1109/MRA.2015.2432331

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35–45. https://doi.org/10.1115/1.3662552

Klasing, K., Althoff, D., Wollherr, D., & Buss, M. (2009). Comparison of surface normal estimation methods for range sensing applications. *2009 IEEE International Conference on Robotics and Automation*, 3206–3211. https://doi.org/10.1109/ROBOT.2009.5152493

Labbe Jr, R. R. (2023). *Kalman and Bayesian Filters in Python*. https://freecomputerbooks.com/Kalman-and-Bayesian-Filters-in-Python.html

Luenberger, D. G. (1964). Observing the State of a Linear System. *IEEE Transactions on Military Electronics*, *8*(2), 74–80. https://doi.org/10.1109/TME.1964.4323124

O'Rourke, J. (1985). Finding minimal enclosing boxes. *International Journal of Computer & Information Sciences*, *14*(3), 183–199. https://doi.org/10.1007/BF00991005

Rusu, R. B., Blodow, N., & Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. *2009 IEEE International Conference on*

Robotics and Automation, 3212–3217. https://doi.org/10.1109/ROBOT.2009.5152473

Schaper, U., Sagert, C., Sawodny, O., & Schneider, K. (2011). A load position observer for cranes with gyroscope measurements. *IFAC Proceedings Volumes*, *44*(1), 3563–3568. https://doi.org/10.3182/20110828-6-IT-1002.01456

Segal, A., Haehnel, D., & Thrun, S. (2009). Generalized-ICP. *Robotics: Science and Systems V*. https://doi.org/10.15607/RSS.2009.V.021

Tech, L. (2019). Livox Horizon User Manual. Retrieved August 23, 2023, from https://www.livoxtech.com/3296f540ecf5458a8829e01cf429798e/assets/horizon/Livox%20Horizon%20user%20manual%20v1.0.pdf

Statement of Affirmation

I hereby declare that all parts of this thesis were exclusively prepared by me, without using resources other than those stated above. The thoughts taken directly or indirectly from external sources are appropriately annotated. This thesis or parts of it were not previously submitted to any other academic institution and have not yet been published.

Signed in Dornbirn on September 4, 2023

[illegible]

Thomas Rosenberger