UNIVERSITY OF CALIFORNIA
RIVERSIDE

Frame Definition, Pose Initialization, and Real-Time Localization in a Non-Stationary Reference Frame With LiDAR and IMU: Application in Cooperative Transloading Tasks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Zeyi Jiang

December 2022

Dissertation Committee:

Dr. Jay A. Farrell, Chairperson
Dr. Matthew J. Barth
Dr. Guoyuan Wu

The Dissertation of Zeyi Jiang is approved:

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgments

As I approach the completion of my dissertation, I would like to express my sincere gratitude to all those who have helped and supported me.

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Jay A. Farrell. I would like to thank him for his guidance, advice, and encouragement on my journey toward the Ph.D. He showed and taught me to think, communicate, and work with the professional rigor of an engineer. The lessons he taught me will be a lifelong treasure for me. I would also like to thank the other members of my dissertation committee, Prof. Matthew Barth and Prof. Guoyuan Wu, for their valuable advice and support on the dissertation.

I would also like to thank my colleagues, Dr. Farzana Rahman, Dr. Mohammad Billah, Dr. Felipe O. Silva, Xuqing Liu, Mike Ma, Dr. Elahe Aghapour, Wang Hu, Jean-Bernard Uwineza and Ashim Neupane, who have provided me with invaluable support and inspiration in my work. Without the collaboration and discussions with them, it would have been much more difficult to complete this journey. Also, I would like to thank my friends, Ruzhuo Wang, Yongqing Zhu, Xuezixiang Li, Xinping Shi, Xuechen Zhang, Hang Yang, Kaiqing Chen, Xiaojun Dong, Dr. Luting Yang, Dr. Xinyue Kan, Dr. Hongsheng Yu, Dr. Shan Sun, and Dr. Pengxiang Zhu, for making the days in Riverside one of the most cherished times of my life.

I would especially like to thank my parents, Hongbin Jiang and Qian Zhang, for their unconditional support, understanding, and encouragement, and my girlfriend Jiaojiao Cheng and our cat Mojito, for experiencing all the joyful and tough moments together with me. Their strong and warm companionship encouraged me to face any challenge at any time.

To my beloved parents and girlfriend.

Your love is the best thing I have.

ABSTRACT OF THE DISSERTATION

Frame Definition, Pose Initialization, and Real-Time Localization in a Non-Stationary Reference Frame With LiDAR and IMU: Application in Cooperative Transloading Tasks

by

Zeyi Jiang

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2022
Dr. Jay A. Farrell, Chairperson

Cargo transloading is an important part of transportation, and the increasing need for transportation is a challenge to the transloading capacity. Therefore, automating the process to reduce manpower consumption and improve the efficiency of transloading is of interest. Cooperative transloading is a common method for bulk cargo transloading, which refers to a robot inside the container that transfers the bulk cargo to a convenient position that the crane can reach. Cooperation improves the crane grabbing efficiency and avoids the bulk cargo being in the container areas that the crane cannot reach. In this dissertation, a method is proposed to recognize and locate the container hatch from point cloud, to unify the reference frame of the crane and robot, to determine the initial robot pose in that shared frame, and to localize the robot as it maneuvers in the container in real-time during the transloading. The main contributions of this research fit into two categories.

- Frame Definition and Initial Pose Determination: To establish a reference frame recognizable by both the crane and the robot, a portion of this dissertation focus on extracting the hatch from the robot point cloud. One hatch corner and the hatch edges define the origin and axis of the shared working frame for the robot and crane. To find the hatch, the 3D point cloud

scanned by the robot is rasterized into 2D data, preserving the relative position information of the hatch. A method based on the Hough-Transform (HT) is used to determine the initial point cloud translation and rotation with respect to the hatch using the 2D data. With the determined translation and rotation, the common reference frame is defined, the point cloud is re-coordinatized into this frame and the initial robot pose can be determined and expressed in this frame.

- Real-Time Localization: The transloading starts after determining the robot initial pose. The robot moves in the container. To avoid collisions, the robot real-time position needs to be reported to the crane. In this dissertation, a basemap is created initially using the point cloud scanned for determining the initial pose. Later, when the robot moves, its Light Detection and Ranging (LiDAR) scans are matched with the basemap using an Iterative Closest Point (ICP) algorithm to determine the robot pose in real-time. To achieve more reliable matching results with the ICP algorithm, several approaches are compared for roughly aligning the real-time scans to the basemap before using the ICP algorithm, including the use of LiDAR-estimated poses and velocities, and the use of Inertial Measurement Unit (IMU) measurements to calculate the pose change. The experimental comparisons of those methods are assessed to determine the most suitable one for cooperative transloading.

In addition to the analysis and development of new methods for hatch recognition, cooperative frame definition, and real-time localization in a non-stationary frame, this research has developed a fully functional real-time prototype implementation.

Contents

List of Figures

Chapter 1

Introduction

Transloading is an important part of international shipping, where products are transferred from one mode of shipping to another for the next shipping step. Cooperative transloading is a typical process in bulk cargo shipping in which several pieces of equipment work together to complete the transloading process. A common scenario is the cooperative transloading of a robot and a crane, where a robot in the original container moves the cargo to a convenient location for the crane to pick up, and the crane cyclically picks the cargo stacked by the robot and places it at the target discharge location. This cooperative transloading process can greatly improve the transload speed, and the robot can also help to complete the transloading of cargo that are unreachable by the crane. This cooperative transloading process is still mainly done manually. Engineers and researchers are interested in automating the transloading process to reduce shipping costs. Furthermore, the automation of transloading process can also reduce the potential for injury to personnel. One of the key points of automated transloading is to communicate the robot's position to the crane in real time to avoid collisions, which is also the main problem of this dissertation. In this dissertation, this

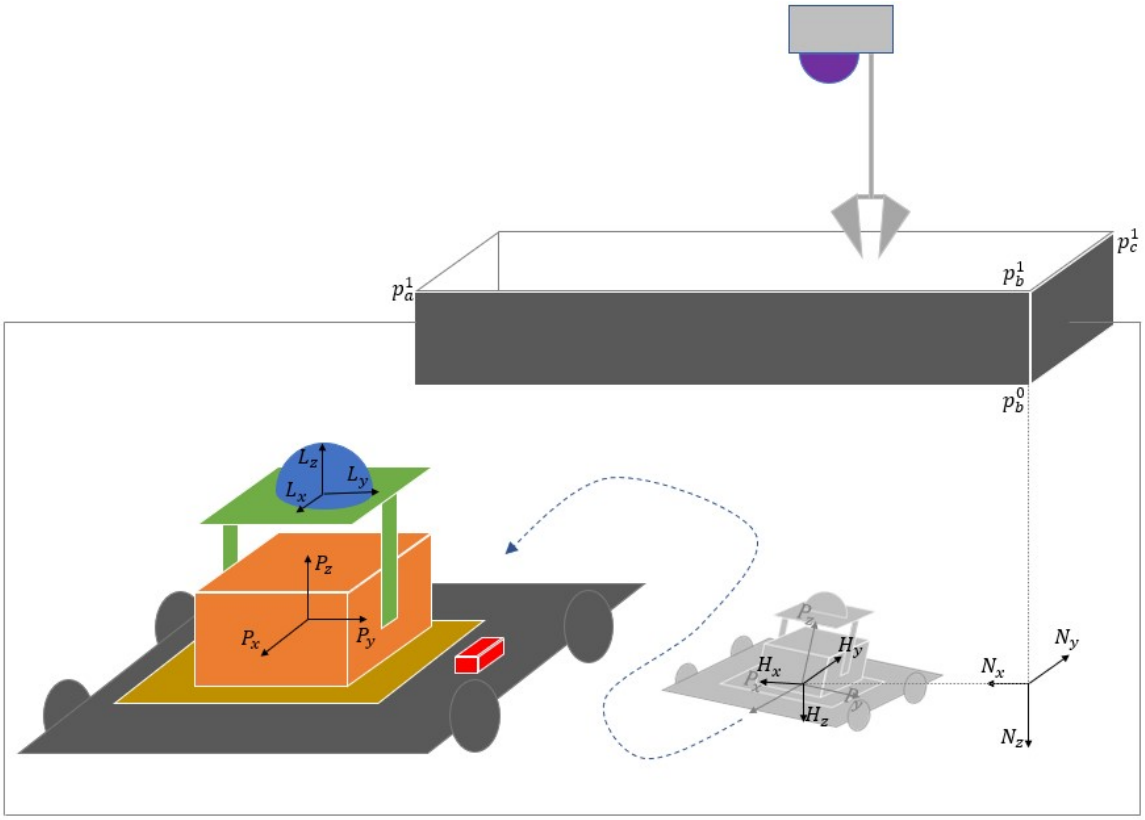

Figure 1.1: Schematic showing the frames and the problem. The sizes of the objects do not match their actual sizes. The gray (color) flatbed indicates the robot initial (current) pose. The purple semicircle represent the crane LiDAR position, where it may not be able to observe the robot directly. The rectangular hatch is a common object detected separately by the robot and the crane such that they can define a common-reference-frame for representing the robot pose.

problem is divided into two parts. First, establishing a common reference frame for the robot and the crane to represent the robot's position. Second, obtaining the robot's position in real time during transloading work.

1.1 Common-Reference-Frame

A common-reference-frame refers to a frame that is used to represent both robot and crane positions during the whole cooperative transloading period. Because the robot works in a

close-space container, it may not have direct visibility to the crane due to occlusion. Therefore, the common-reference-frame is essential for sharing the robot position to the crane during the occlusion such that the vector from the crane to the robot can be calculated in real-time by differencing the latest crane and robot positions.

Figure 1.1 shows a scenario of the cooperative transloading. A crane is hovering above a container with a rectangular hatch. A LiDAR is mounted at the crane base to scan the point cloud below, marked in purple in the figure. The robot is placed inside the hatch and a LiDAR marked in blue is mounted to scan the hatch environment. The location where the robot was first placed inside the container is marked in the figure with a gray flatbed, from where more of the structure of the hatch can be observed. To represent the robot's position in the same frame as the crane during the whole transloading, it is necessary to define a common reference frame with the crane at the initial position. To define a common reference system during the initialization phase, it is necessary to recognize and locate the hatch from the point cloud scanned by the LiDAR of the crane and the robot. The common reference frame is established based on the extracted hatch, which is the most common object that the crane and the robot can both observe and the hatch is prevalent on different types of containers. Meanwhile, the robot's initial pose can also be determined in this common reference frame.

1.2 Real-Time Robot Positioning

After establishing the common-reference-frame and determining the initial robot pose, the robot can move and start the cooperative transloading work. The LiDAR on the robot generates points during its motion, and the coordinates of these points are represented in the robot's body

frame. Therefore, as long as the rotation and translation parameters can be confirmed in real time to transform these points to match the point cloud scanned at the initial robot position (where the common reference frame is defined), the robot position can be determined in real-time and represented in the common reference frame to avoid collisions between the crane and the robot during cooperative transloading.

1.3 Sensor Choice

The sensors of the application are selected to succeed for solving the problem. Commonly used sensors for positioning includes: Global Positioning System (GPS), camera, odometer, IMU, and LiDAR. GPS [43, 55] can reach centimeter-level accuracy with good signals and provide a common reference frame when sharing the receiver position. However, the GPS localization is unreliable when the signal is weak. In this dissertation, a vehicle is in the container (see Fig. 1.1) with limited vision of the outer space. Therefore, the GPS signal cannot be guaranteed for localization. Camera [50, 52, 59] provides color information for a dense array of pixels, which is convenient for determining the boundaries of objects. However, camera performance may degrade under poor lighting, shading, or at night. The transloading tasks commonly last for a few days so camera is inappropriate for work at night. Odometer [69] can provide the distance that a vehicle travels. However, the application of this dissertation aims at modifying existing vehicles while the odometer reading is unavailable. IMU [12, 25, 34] generates acceleration and angular-rate measurements of its body-frame relative to the inertial-frame at a high frequency. Moreover, it can be easily installed on existing devices. LiDAR [21, 46, 53, 63] is active, emitting energy and detecting reflections, allowing it to determine time-of-flight at specific directions around the clock. The LiDAR time-of-flight

can be used to calculate the distance between the sensor and the reflected point at an known angle, allowing computation of the vector from the sensor to that point along with the reflection intensity. LiDAR's are inexpensive, easy to work with, and have accuracy and resolution specifications high enough at a long distances to succeed in this application. Therefore, LiDAR and IMU are selected for the problem.

1.4 Proposed Approach

In the approach of this dissertation, the whole process is divided into two stages: the hatch extraction stage and the real-time localization stage.

A rectangular hatch is selected as the target in the hatch extraction stage because it is commonly observable from both the robot and the crane. For the hatch extraction for the crane, it is discussed in [24]. A sensor suite of LiDAR and Pan-Tilt Unit (PT) is used to accumulate point cloud cyclically. The point cloud is voxelized and converted into point pillars [26], which can be rasterized into 2D data preserving information of the hatch position. A HT method is used to extract lines from the rasterized data and determine the hatch position. The crane position in the common-reference-frame is updated per cycle from the accumulated points of each cycle. For the hatch extraction for the robot, because the hatch edges are not aligned as in [24], the point cloud is first rasterized such that the points on vertical walls in the point cloud can be converted into 2D points on lines. With the knowledge that the inner walls of the container are mostly parallel to the hatch edges, those extracted lines are used to calculate a rotation matrix which can align the initial point cloud and determine the hatch position. The common reference frame is defined, and the robot

initial position in the common reference frame is determined with the extracted hatch. The detailed process will be discussed in Chapter 3.

After determining the common reference frame and initial robot pose, the initial point cloud used for extracting the hatch can be converted into the common reference frame. The transformed point cloud will be used as the basemap, which is used for the following localization of the robot. When the robot starts moving, each scan from the LiDAR can be registered to the basemap and the transformation parameters for registration can be used to obtain the robot pose at collection of the scan. To make the registration more reliable, the scan might be roughly aligned to the basemap before registration algorithms are applied. The prior pose can be calculated in multiple approaches: the pose estimated in previous epochs can be used to calculate estimated velocity and angular rate solely based on LiDAR data, and the angular rate measured by IMU might be used instead of the estimated angular rate, and the IMU acceleration measurements might be used in addition to the LiDAR estimated velocity. The discussion on calculating prior pose and the real-time localization will be discussed and compared in Chapter 5.

Chapter 2

Preliminary Knowledge and Information

2.1 Hardware Setup

The hardware setup includes the setup of the robot and the setup of the crane. The hardware setup of the crane should be referred to [24]. The hardware setup of the robot is shown in Figure 1.1. The hardware of the robot includes a flatbed, a PT, a LiDAR, and an IMU. The installation of these sensors are shown as the schematic in Figure 1.1.

The PT consists of three parts as shown in Figure 1.1: base (brown piece), body (orange cube), and platform (green piece). The base is rigidly attached on the flatbed. The body can rotate (i.e., pan) around the *P*-frame z-axis relative to the base. The platform can rotate (i.e., tilt) relative to the body with the two green arms as shown in Figure 1.1. The vector formed by the two connection points between the arms and the body is the rotation axis (i.e., *P*-frame y-axis) of the tilt-rotation. There are two Dimension-of-Freedom (DOF) for the PT, which are the pan and tilt rotation. The *P*-frame will be defined in Section 2.2 for more details. The PT is installed on the

flatbed such that the robot heading is defined along the same direction of P-frame x-axis as shown in Figure 1.1.

The LiDAR (drawn as a blue hemisphere) is rigidly attached to the PT platform. Therefore, the LiDAR can be rotated by the PT and scan difference direction. The LiDAR is a mechanical LiDAR which has a laser-array distributed and fixed on a rotation-axis (i.e., the L-frame z-axis) within a 0-to-90 degree angular range. The laser-array can spin around the L-frame z-axis at a consistent speed of 10 revolutions per second. Therefore, the Field of View (FOV) of the LiDAR is 360 degrees horizontally and 90 degrees vertically. For an example, when the PT is at zero position as shown in Figure 1.1, the LiDAR should scan half of the space above it. The points generated by all lasers during one such revolution is called on scan.[1] The scan-rate of the LiDAR is denoted as f_l and a scan is generated every $1/f_l$ second in L-frame.

The IMU is rigidly mounted on the flatbed near the PT base as shown in Figure 1.1. The IMU is capable of measuring acceleration and angular velocity relative to the inertial frame at a frequency of $f_m = 200$ Hz. The axes for the IMU measurements are marked on the red cube in Figure 1.1. In this dissertation, the IMU is used to maintain the robot position and orientation between LiDAR scans.

[1] A 'scan' is also known as a 'frame' in the literature. In this dissertation, scan-rate (scan) is used in place of frame-rate (frame) to avoid confusion with the definitions of coordinate systems in Section 2.2.

2.2 Frame Definitions

The frames used in this dissertation are defined as:

1. *L*-frame: This frame is the fixed frame on the LiDAR determined by the LiDAR manufacturer. The LiDAR generates points in this frame. The points are transformed into other interested frame for further processing.

2. *P*-frame: This frame is fixed relative to the PT base (i.e., brown piece in Figure 1.1) with its origin at the pan and tilt rotation center. The x-axis is to the front of the robot and y-axis points left. z-axis points up to form an right-hand coordinate system (RHS). The PT base is rigidly attached to the robot so that the *P*-frame moves together with the robot.

3. *H*-frame: this frame is defined when the robot is initially placed in the room. The origin coincides with the *P*-frame origin before the robot moves. The x and y axes of *H*-frame are nominally aligned to the consecutive edges of a corner point of a rectangular hatch, which is determined during the point cloud alignment and will be discussed in Section 3.6. *H*-frame is fixed at a point in the room as defined with the robot initial position. Therefore, *H*-frame will stay when the robot moves. *H*-frame is only used in the initialization process. After that it is ignored.

4. *N*-frame: Navigation frame is defined with the origin at a fixed reference point in the container, which has the same x and y coordinates in *H*-frame with the corner point used for defining *H*-frame. The origin z coordinate is zero in *H*-frame. The x-axis of *N*-frame is parallel to the *H*-frame y-axis. The y-axis of *N*-frame is parallel to the *H*-frame x-axis. The z-axis is defined to form a RHS. In this dissertation, the navigation map and robot pose are

both represented in N-frame. This frame is shared with the crane outside of the container to provide the estimated robot pose in an unique frame. The establishment of N-frame is based on the point cloud alignement and hatch edge extraction results in H-frame, which will be discussed in Section 3.8.

5. A-frame: This frame is nominally coincident with N-frame after the transformation with the current pose of the robot. The A-frame of one scan lives only for expressing this scan at a position roughly aligned to the N-frame basemap. Each scan will define its own A-frame which is determined by the prior knowledge of the robot pose. In Chapter 5, each scan needs to be transformed into A-frame for better performance when using ICP algorithm to register it to the basemap.

6. g-frame: This frame is defined as the inertial frame, which will be used for discussing problems related to IMU raw measurements.

2.3 Notation

This section describes the the general format of the notation. Note that only P-frame is specifically used and the other subscripts and superscripts in this section are non-specific.

1. ${}^{X}\mathbf{T}^{k}_{YZ} \in \mathscr{R}^3$: This symbol represents the translation vector from Y to Z at time epoch t_k represented in X-frame. When the symbol k is absent, it means the symbols in the equation are at the same time epoch.

2. ${}^{W}_{X}\mathbf{R}^{k}_{j} \in \mathscr{R}^{3\times3}$: This symbol represents a general form of a rotation matrix. One of the subscripts will be absent (e.g., ${}^{W}_{X}\mathbf{R}^{k}$ or ${}^{W}\mathbf{R}^{k}_{j}$) when this symbol is used. The form used in the dissertation are as below:

   - ${}^{W}_{X}\mathbf{R}^{k}$: The rotation matrix from X-frame to W-frame at time t_k.
   - ${}^{W}\mathbf{R}^{k}_{j}$: The rotation of W-frame from time t_j to t_k.

3. ${}^{X}\mathbf{P}^{k}_{k-1} \in \mathscr{R}^{3}$: This symbol is used in this dissertation with or without the subscript. The definition of each is as below:

   - ${}^{X}\mathbf{P}^{k}$: The vector from the X-frame origin to the P-frame origin at time t_k represented in X-frame. Therefore, the symbol can be represented as a vector: ${}^{X}\mathbf{P}^{k} = {}^{X}\mathbf{T}^{k}_{XP}$. It is defined as a simpler expression of the position of an object when P-frame is attached to this object.
   - ${}^{X}\mathbf{P}^{k}_{k-1}$: The change of P-frame origin position from t_{k-1} to t_k represented in X-frame.

4. ${}^{X}\mathbf{p}_{i} \in \mathscr{R}^{3}$: This symbol represents the i-th point ${}^{X}\mathbf{p}_{i} = [{}^{X}x,\ {}^{X}y,\ {}^{X}z]^{\top}$ in an X-frame point cloud. ${}^{X}x,\ {}^{X}y,\ {}^{X}z$ are the $x,\ y,\ z$ coordinates of the point in X-frame. The subscript i will be absent if the point is not in a specific point cloud. The symbol is equivalent to the vector from X-frame origin to the point: ${}^{X}\mathbf{p}_{i} = {}^{X}\mathbf{T}_{Xp}$.

5. ${}^{X}\mathbf{S}^{k} \in \mathscr{R}^{3\times m}$: The symbol ${}^{X}\mathbf{S}^{k}$ represents the scan collected at time t_k with m points in X-frame.

6. ${}^{X}\mathbf{C} \in \mathscr{R}^{3\times n}$: The symbol ${}^{X}\mathbf{C}$ represents a point cloud with n points in X-frame.

7. The symbol $\lceil \mathbf{X} \rceil$ is used as the ceiling function of a vector $\mathbf{X}$, which outputs a vector with same dimension. Each element of the output vector is the smallest integer larger than or equal to the corresponding element in $\mathbf{X}$.

8. The symbol $\lfloor \mathbf{X} \rfloor$ is used as the floor function of a vector $\mathbf{X}$, which outputs a vector with same dimension. Each element of the output vector is the largest integer smaller than or equal to the corresponding element in $\mathbf{X}$.

9. The symbol $\mathbb{B}$ is used to represent a set of binary numbers.

2.4 Georectification

The points are initially generated in L-frame. This section introduces the frame conversion applied to each LiDAR points. The different reference frames and conversions between them are well discussed in Section 2 of [15].

2.4.1 L-frame to P-frame

Every L-frame point ${}^{L}\mathbf{p} \in \mathscr{R}^3$ is recorded together with the PT pan-angle α_i and tilt-angle β_i at the time of collecting the point. Therefore, each point ${}^{L}\mathbf{p}_i$ can be transformed into P-frame at its generation:

$$ {}^{P}\mathbf{p}_i = {}^{P}_{L}\mathbf{R}_i \left({}^{L}\mathbf{p}_i + {}^{L}\mathbf{T}_{PL} \right), \tag{2.1} $$

$$ {}^{W}\mathbf{p}_i = {}^{W}_{X}\mathbf{R}^k \left({}^{X}\mathbf{p}_i + {}^{X}\mathbf{T}_{WX} \right), \tag{2.2} $$

where ${}^{L}\mathbf{T}_{PL}$ is the vector from P-frame origin to L-frame origin in L-frame. This vector ${}^{L}\mathbf{T}_{PL}$ is a constant and measured in advance. The rotation matrix ${}^{P}_{L}\mathbf{R}_i$ is calculated with the pan-angle α_i and tilt-angle β_i at the same time with the point generation time of ${}^{L}\mathbf{p}_i$ using eqns. (2.40-42) in [15]:

$$ {}^{P}_{L}\mathbf{R}_i = \begin{bmatrix} \cos(\alpha_i) & -\sin(\alpha_i)\ \cos(\beta_i) & \sin(\alpha_i)\ \sin(\beta_i) \\ \sin(\alpha_i) & \cos(\alpha_i)\ \cos(\beta_i) & -\cos(\alpha_i)\ \sin(\beta_i) \\ 0 & \sin(\beta_i) & \cos(\beta_i) \end{bmatrix}. \tag{2.3} $$

2.4.2 P-frame to H-frame

This frame conversions only happen during initialization, when the H-frame origin is identical with P-frame origin (i.e., $\mathbf{T}_{HP} = \mathbf{0}$). The conversion from a P-frame point to a H-frame point during initialization is:

$$ {}^{H}\mathbf{p}_i = {}^{H}_{P}\mathbf{R}^{0}\ {}^{P}\mathbf{p}_i \tag{2.4} $$

where the rotation matrix ${}^{H}_{P}\mathbf{R}^{0}$ can be determined with various approaches, which will be discussed in Section 3.6.

2.4.3 P-frame to N-frame

When the initialization is complete, N-frame should be built and the conversion from P-frame to N-frame is used.

The conversion from a P-frame point to a N-frame point is:

$$ {}^{N}\mathbf{p}_i = {}^{N}_{P}\mathbf{R}^{k}\ {}^{P}\mathbf{p}_i + {}^{N}\mathbf{T}^{k}_{NP}, \tag{2.5} $$

where ${}^{N}_{P}\mathbf{R}^{k}$ and ${}^{N}\mathbf{T}^{k}_{NP}$ are the direction cosine matrix and the translation vector of the robot in N-frame at epoch k. These two variables represent the orientation and position of the robot at epoch k.

2.4.4 P-frame to A-frame

The conversion from a P-frame point to a A-frame point is:

$$ {}^{A}\mathbf{p}_i = {}^{N}_{P}\bar{\mathbf{R}}^{k}\ {}^{P}\mathbf{p}_i + {}^{N}\bar{\mathbf{T}}^{k}_{NP}, \tag{2.6} $$

where ${}^{N}_{P}\bar{\mathbf{R}}^{k}$ and ${}^{N}\bar{\mathbf{T}}^{k}_{NP}$ are the estimated variables of ${}^{N}_{P}\mathbf{R}^{k}$ and ${}^{N}\mathbf{T}^{k}_{NP}$, which will be discussed in Chapter 5.

2.4.5 H-frame to N-frame

The conversion from a H-frame point to a N-frame point is:

$$ {}^{N}\mathbf{p}_i = {}^{N}_{H}\mathbf{R}({}^{H}\mathbf{p}_i + {}^{H}\mathbf{T}_{HN}), \tag{2.7} $$

where

$$ {}^{N}_{H}\mathbf{R} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{2.8} $$

and ${}^{H}\mathbf{T}_{HN}$ is determined with the hatch extraction results, which will be discussed in Section 3.7. This conversion is used for localization initialization, which will be discussed in Section 3.8.

2.5 Iterative Closest Point: Point Cloud Registration

ICP is an algorithm [47] adopted to align one source point cloud ($\mathbf{P}_2$) to another target point cloud ($\mathbf{P}_1$). Denote each point in the source point cloud as $p_s \in \mathbf{P}_2$, its closest point p_t in $\mathbf{P}_1$ is searched such that the Euclidean distance is minimized: $\underset{t}{\operatorname{argmin}}\|p_s - p_t\|$. If $\|p_s - p_t\| < \zeta$, which is a user-defined threshold, the pair of these points (p_s, p_t) is regarded as one valid correspondence. The collection of points in $\mathbf{P}^s$ that have valid correspondence are denoted as a point cloud $\mathbf{P}_2^*$. The corresponding points of each point in $\mathbf{P}_2^*$ are denoted as $\mathbf{P}_1^*$, which is a subset of $\mathbf{P}_1$. Both $\mathbf{P}_2^*$ and $\mathbf{P}_1^*$ have N_j points and they can be expressed as matrices with three rows and N_j columns, where each column includes the coordinates of one point. Therefore, singular value decomposition (SVD) [5] can be used to calculate a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{T}$ such that:

$$\mathbf{R}_i, \mathbf{T}_i = \underset{\mathbf{R},\mathbf{T}}{\operatorname{argmin}} \sum_{j=1}^{N_j} \|\mathbf{R}\ p_{s,j} + \mathbf{T} - p_{t,j}\|, \tag{2.9}$$

where $p_{s,j} \in \mathbf{P}_2^*$, $p_{t,j} \in \mathbf{P}_1^*$ and $\mathbf{R}_i, \mathbf{T}_i$ are calculated as the transformation parameters of the i-th iteration of ICP. Then, each point $p_s \in \mathbf{P}_2$ is transformed using $\mathbf{R}_i, \mathbf{T}_i$ as: $p_s^i = \mathbf{R}_i\ p_s + \mathbf{T}_i$ and forms the transformed source point cloud $\mathbf{P}_2^i$. Several criterion (e.g., difference between consecutive transformation, averaged distance between corresponding points in the source and target point cloud, and so on) can be checked to determine whether the transformed point cloud $\mathbf{P}_2^i$ is successfully aligned to the target point cloud $\mathbf{P}_1$ in this iteration. If the criterion is not accomplished, $\mathbf{P}_2^i$ will be used to find its correspondence with $\mathbf{P}_1$, calculate the transformation parameters and transformed itself into $\mathbf{P}_2^{i+1}$ in a new iteration until the criterion is met. The transformation of each point of $\mathbf{P}_2$ to a point in the point cloud $\mathbf{P}_2^{N_i}$, which is the output of the last iteration before convergence, can be represented

as:

$$p_s^{N_i} = \mathbf{R}_{N_i}(\mathbf{R}_{N_i-1}(\dots\ \mathbf{R}_1\ p_s + \mathbf{T}_1)\dots\ + \mathbf{T}_{N_i-1}) + \mathbf{T}_{N_i}, \tag{2.10}$$

therefore, the rotation matrix can be denoted as:

$$\mathbf{R}^* = \mathbf{R}_{N_i}\ \mathbf{R}_{N_i-1}\ \dots\ \mathbf{R}_1, \tag{2.11}$$

and the translation vector can be denoted as:

$$\mathbf{T}^* = \mathbf{R}_{N_i}(\mathbf{R}_{N_i-1}\dots\ (\mathbf{R}_2(\mathbf{T}_1) + \mathbf{T}_2)\ \dots\ + \mathbf{T}_{N_i-1}) + \mathbf{T}_{N_i}. \tag{2.12}$$

The above iterative process to align a source point cloud $\mathbf{P}_2$ to another target point cloud $\mathbf{P}_1$ is the steps of the ICP algorithm. In this dissertation, when such processes is used to align a point cloud, the process will be expressed in the following form:

$$[R^*,\ T^*] = ICP(\mathbf{P}_1,\ \mathbf{P}_2), \tag{2.13}$$

and used in this dissertation.

The success of ICP relies on finding correct correspondences. Therefore, a common pre-processing is applying a prior transformation to $\mathbf{P}_2$ before using the ICP algorithm. In Chapter 5, several approaches will be discussed to obtain a prior transformation such that $\mathbf{P}_2$ can be roughly aligned to $\mathbf{P}_1$ and better correspondence can be found. The usage of ICP with prior transformation will be expressed as:

$$[R^*,\ T^*] = \underset{X_1, X_2, \dots, X_N}{ICP}(\mathbf{P}_1,\ \mathbf{P}_2), \tag{2.14}$$

where the symbols X_1 to X_N are the prior used to transform $\mathbf{P}_2$ for pre-processing.

2.6 Hough-Transform: 2D Line Detection

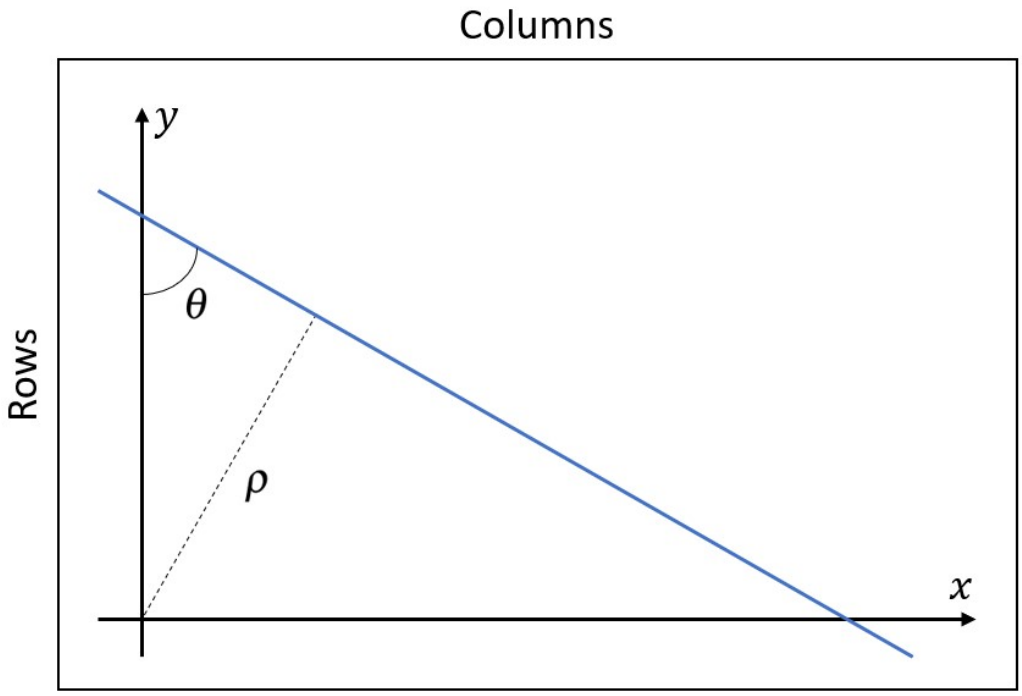

Figure 2.1: Schematic for a line represented in a polar coordinate system which is attached to a 2D matrix.

The HT algorithm [3, 13] is a voting procedure widely used for extracting geometries that can be parameterized (e.g., lines, planes, circles). In this dissertation, the HT algorithm is used to extract lines from 2D binary matrices (i.e., black-and-white images). The model for a line is:

$$\rho = x\,sin(\theta) + y\,cos(\theta), \tag{2.15}$$

where ρ is the distance from the line to the origin and θ is the angle of the line. A line is located in one row (column) of the 2D matrix when the angle parameter $\theta = \frac{\pi}{2}$ (0) as indicated in Figure 2.1.

When the HT algorithm is used for extracting lines from a 2D binary matrix $\mathbf{M}$, a voting table $\mathbf{B}$ is formed with $\frac{\bar{\rho}}{\Delta_\rho}$ rows and $\frac{\pi}{\Delta_\theta}$ columns, where Δ_ρ and Δ_θ are the resolution of ρ and θ.

Each entry $\mathbf{B}_{i,j}$ represents the votes of a line parameterized as (ρ_i, θ_j), where $\rho_i = i\Delta_\rho,\ i = 1, \ldots, \frac{\bar{\rho}}{\Delta_\rho}$ and $\theta_j = -\frac{\pi}{2} + j\Delta_\theta,\ j = 0, \ldots, \frac{\pi}{\Delta_\theta} - 1$.

The range of ρ is determined to be large enough for the voting of each data point (i.e., a pair of x and y) in $\mathbf{M}$. For each entry $\mathbf{M}_{x,y} = 1$, $\tilde{\rho}_i$ is calculated using eqn. (2.15): $\tilde{\rho}_i = x cos(\theta_i) + y sin(\theta_i)$. The calculated value $\tilde{\rho}_i$ is then rounded to the closest ρ_i, which is determined by the range of ρ and the resolution Δ_ρ, and votes for a candidate line (ρ_i, θ_i). As a result of the voting process, the entries in the voting table with votes more than a user-define threshold are extracted as the line parameters in the image.

Chapter 3

Frame Definition and Pose Initialization Theory

This chapter discusses the approach to extract the hatch from the point cloud generated by robot LiDAR. This chapter first introduces some relevant literature in Section 3.1, then Section 3.2 describes the data acquisition process used for hatch detection. Section 3.3 states the objective and assumptions, then defines several subproblems that will be solved to define an approach. Sections 3.4 to 3.9 describe the approach for detecting the hatch, and Chapter 4 shows the experimental results corresponding to this approach.

3.1 Literature Review

Hatch extraction for transloading using LiDAR is considered in [35, 36, 38] for ship loading. In that application, the hatch is empty when the process starts, which facilitates separation of the hatch from the cargo as they have very distinct ranges from the LiDAR. The hatch in that

application is aligned to the axes of the frame, which the point cloud is accumulated in, because the LiDAR installation can be tuned to accomplish this before the transloading work. Mi et al. [35] discuss a method that directly processes scanlines from a 2D SICK LiDAR. It compares the slope between every two consecutive points relative to a predefined change-of-slope threshold. When the slope change is sufficiently large, the two corresponding points are regarded as candidate hatch edge points. The hatch edge is determined by accumulating the extracted hatch edge points over multiple scanlines. Mi et al. [36] perform a whole ship scan with four 2D SICK LiDARs. The point cloud of the ship is preprocessed to remove cargo points. Then, their method calculates the histogram statistics of the remaining x and y values. Because the cargo points in the hatch have been removed, the histogram of x values is smaller in the hatch area than in other areas. Therefore, the rising edge and falling edge of the histogram determine the x-direction hatch edge positions. The same process is applied separately to determine the y-direction edge positions. The hatch edge positions are calculated once after scanning the whole ship once. Miao et al. [38] use a Livox Horizon solid state LiDAR, which generates a point cloud of the hatch area every 0.1 seconds. They rasterize each of these point clouds into an image, where the value of each image pixel is the range between a point and the LiDAR origin and the columns and rows of the image are defined by the laser emission angles (i.e., azimuth and elevation). An edge detection algorithm is then used to process this image. The detected edges are regarded as the hatch edges and used to determine the hatch location. The algorithm of [38] finds hatch edges for each point cloud. Due to their focus on ship loading, all three articles assumes the hatch is aligned in the frame of accumulating point cloud. This article presents an approach designed to work for both loading and unloading operations and for both the point cloud collected from the crane and robot LiDAR's. This paper proposes an approach that finds

the four vertical walls surrounding the hatch (i.e., vertical planes) such that the initial orientation of the robot relative to the hatch can be determined. Then the approach in our previous work [24] to extract the hatch from the crane point cloud can work with the adjusted robot point cloud to extract the hatch. Extracting the same hatch and defining a common reference frame is essential for cooperative transloading between a crane and a robot.

The literature contains three dominant categories of methods for extracting 3D planes from an unorganized point cloud: Random Sample Consensus (RANSAC) [16,64], Region-growing [1], and HT [3, 13, 22]. Choi et al. evaluated different variants of the RANSAC algorithm [11]. RANSAC works well when the desired model is known and most data are inliers of the model. In this application, the desired model for each hatch wall is a plane. However, the points on any single hatch wall are only a small portion of the whole point cloud so the probability is small of selecting three points all on one hatch wall plane. Therefore, using RANSAC to extract hatch edge planes is not reliable. Region-growing approaches are usually used for point cloud segmentation [58, 67]. Starting from a seed point, each region extends adding neighbor points that have similar characteristics (e.g., surface normal direction). The points in each region can then be processed to extract a plane models. This approach requires processing all scanned points to generate the normal of each point. The computation of estimating normals for a large 3D point cloud is expensive and may not finish soon on an PC of a robot. The HT is a voting algorithm for detecting parameterized shapes [3]. Each point of the input data is mapped to the parameter space and votes are accumulated for parameter values. The accumulator determines the best model as the one with the most votes after all points are processed. Kurdi et al. [54] compare the 3D HT and RANSAC approaches, claiming

that RANSAC works better for their problem because 3D HT is inefficient in both computation and space. Our approach will use the 2D HT on a novel rasterization of the point cloud.

Rasterization [26, 29] is a process commonly used to encode a point cloud into an image [2, 19]. Appropriate rasterization reduces the size of the data while retaining the interesting information. There are different strategies of rasterization. Lang et al. [26] introduce point pillars for their urban vehicle experiments. Each pillar is a collection of LiDAR reflections in a small horizontal area. In [26], the corresponding image pixel is determined by processing these points with a learned model. Li et al. [29] rasterize the point cloud by encoding each image pixel with the maximum z difference of points in each pillar region. Hackel et al. [20] also have similar idea of calculating the maximum z different of points in a pillar region.

3.2 Point Cloud Accumulation

The purpose of this section is to introduce the data acquisition process, the data types used in this method and the geofence used for limiting the amount of points during the point cloud accumulation.

After being placed in the workspace, the robot will stay at a position where it can scan the four vertical sides of the hatch, which represent the hatch position. The robot stays static and rotates the LiDAR (by rotating PT body or platform) to scan and accumulate a point cloud. There is no strict requirement on how the PT rotates, as long as the lasers can cover as many directions as possible. As the PT changes its pan (α) and tilt (β) angles, the LiDAR generates reflections from a sequence of points ${}^{L}\mathbf{p}_i$ from reflecting surfaces in the environment: cargo, hatch edges, and other

inner walls in the closed-space. The L-frame points ${}^{L}\mathbf{p}_i$ are transformed into P-frame points ${}^{P}\mathbf{p}_i$ in real-time using eqn. (2.1).

The methods described herein only use the coordinates of reflected points, not their intensity. The reflection intensity depends on several factors [17] e.g.: range, surface reflectivity, and angle-of-incidence. Because metal hatches and the ores on the floor do not show significant intensity differences relative to each other, the intensity measurement is not a reliable variable for clustering the points in this application.

To limit the extent of the point cloud and remove unneeded points, a geofence is defined to exclude points reflected from surfaces outside the container (through the hatch) or the time-varying ground:

$$ {}^{P}\mathbf{C} = \left\{ {}^{P}\mathbf{p}_i \,\middle|\, \underline{x}_S < {}^{P}x_i < \overline{x}_S, \underline{y}_S < {}^{P}y_i < \overline{y}_S, \underline{h}_e < {}^{P}z_i < \overline{h}_e \right\}_{i=1}^{N_s} \tag{3.1} $$

where the geofence bounds $\underline{x}_S, \overline{x}_S, \underline{y}_S, \overline{y}_S, \underline{h}_e, \overline{h}_e$ are user-defined, specifying the range similar to the container size. More about these constants will be discussed in Section 3.9. The point cloud ${}^{P}\mathbf{C}$ is defined as a conceptual input of the proposed approach, to represent the collection of all points processed within the geofence for hatch extraction. During experiments, each point ${}^{P}\mathbf{p}_i$ that satisfies the geofence constraints will be processed at the time that it is generated, which will be illustrated in Figure 3.2. The point cloud ${}^{P}\mathbf{C}$ is unnecessary for the functioning of the algorithm, but is useful for the purposes of visualization.

3.3 Problem Statement

The objective in this chapter (i.e., the hatch extraction stage) is to extract the hatch position from the P-frame point cloud ${}^{P}\mathbf{C}$. The hatch position is then used for definition of N-frame in the

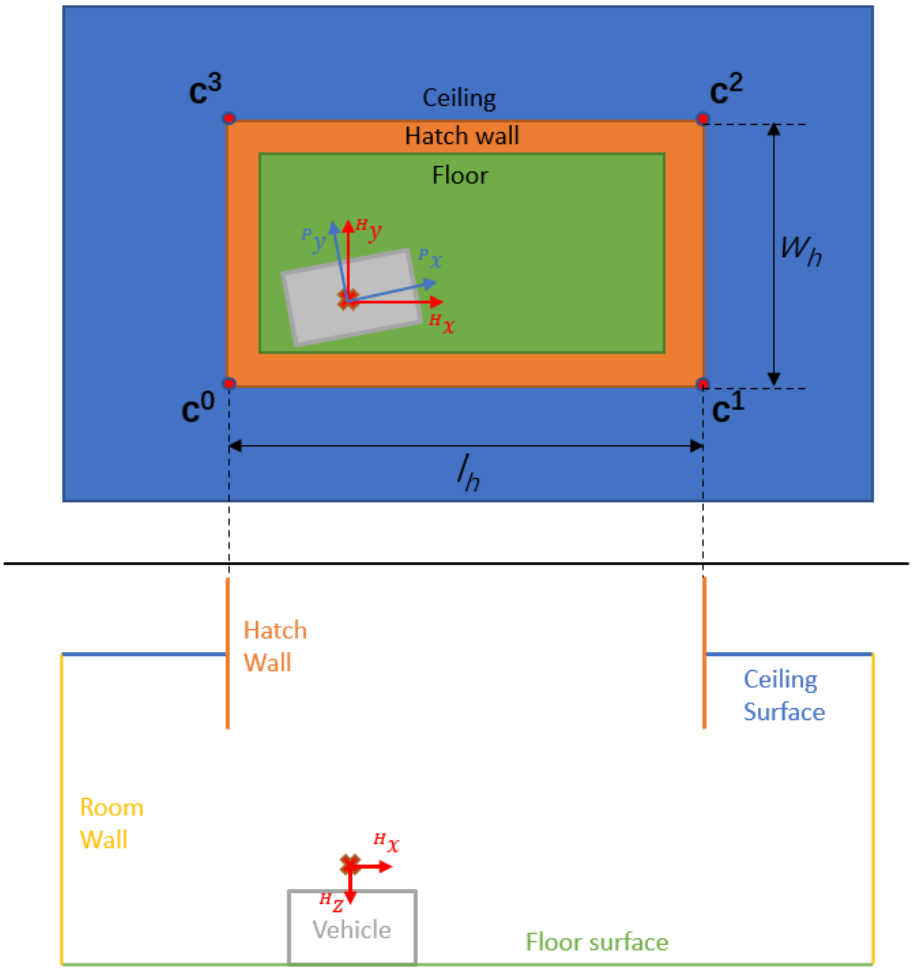

Figure 3.1: Schematic of the structure of a point cloud. (Top) View of the hatch along an vector pointing into the hatch (i.e., the P-frame z-axis). (Bottom) Side view depicting the y-z plane.

real-time positioning stage. This section states the hatch-extraction problem and assumptions, then defines the subproblems that will be solved to define an approach.

Figure 3.1 depicts the structure of the accumulated point cloud and the axes and origin of P-frame and H-frame. The origin and axes of H-frame (i.e., the location of the PT and LiDAR) are shown in red. The x and y axes of P-frame are shown in blue and the P-frame origin and z-axis are the same with the H-frame origin and z-axis. The point cloud portions corresponding to

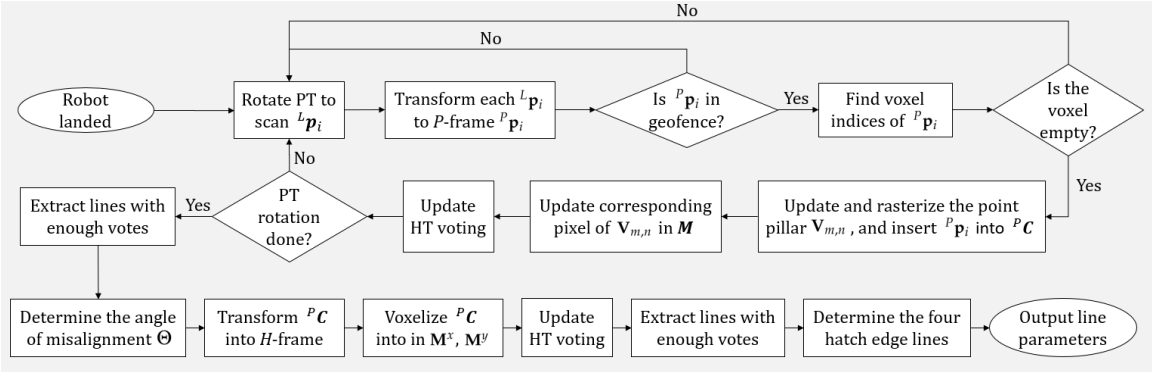

Figure 3.2: Flowchart of the hatch extraction process.

floor surfaces in the hatch are indicated in green. The point cloud portions corresponding to hatch edge walls are indicated in orange. The point cloud portions corresponding to the ceiling surfaces surrounding the hatch are indicated in blue. The point cloud portions corresponding to the room walls are indicated in yellow.

In transloading applications, the crane enters or exits the hatch approximately along the H-frame z-axis. Therefore, determining the robot position relative to the hatch in the x-y plane is sufficient to avoid collisions between the crane and the robot. Herein, determining the 3D hatch edge planes will be simplified to the problem of determining a 2D line in the x-y plane for each of the four edges. The corners of the hatch in the top-view will be denoted by $c^j \in \mathscr{R}^2$, $j = 0,1,2,3$. As shown in the top portion of Figure 3.1, the corner with the smallest distance to the P-frame origin is c^0. Starting from c^0, the other points are defined consecutively in counter-clockwise order. The edge line connecting c^j and the next counter-clockwise corner defines the line parameters $e_j = (\rho_j, \theta_j)$, where the (x,y) coordinates of point on each line satisfy:

$$\rho_j = x\,cos(\theta_j) + y\,sin(\theta_j). \tag{3.2}$$

Figure 3.2 illustrates the process of scanning the environment and extracting the hatch edges from the scanned points. In the following discussion, the geo-fenced point cloud ${}^{P}\mathbf{C}$ will be described as the input to the process. Note that in real application, each point ${}^{P}\mathbf{p}_i \in {}^{P}\mathbf{C}$ can be processed incrementally, at the time it is acquired, as illustrated in the flowchart.

3.3.1 Assumptions

The assumptions on the hatch are as follows.

1. The P-frame origin, which is defined by the scanning system location, is within the x-y-edges of the hatch so that it can scan all four hatch edge planes.

2. Each hatch edge is a segment of a plane.

   (a) Each plane defining an x-edge has a normal pointing approximately parallel to the H-frame y-axis.

   (b) Each plane defining a y-edge has a normal pointing approximately parallel to the H-frame x-axis.

   This implies that the edges of the hatch rectangle are approximately aligned to H-frame x and y axes. Therefore, the outline of the hatch in the H-frame x-y view is rectangular.

3. The internal vertical walls in the container are mostly aligned to any one of the hatch edge planes.

4. The z-extent of the hatch edge plane is deep enough so that the scanning system generates enough reflecting points on the hatch edge planes.

5. Upper bounds $\bar{\ell}$ and $\bar{w}$ are known for the hatch length and width, respectively.

6. The robot is initially placed on a nearly level floor such that the N-frame z-axis is nominally aligned to the P-frame z-axis.

The hatch shape assumption limits the application to rectangular hatches. The assumption of the relative orientation of the hatch to the H-frame is satisfied by properly parking the vehicle or using IMU and GPS to measure the misalignment.

3.3.2 Sub-problems

The problem of finding the hatch is divided into the following sub-problems.

Voxelization: Organize ${}^{P}\mathbf{C}$ into an occupancy matrix ${}^{P}\mathbf{G} \in \mathbb{B}^{N_x \times N_y \times N_z}$, which is defined as a 3D binary matrix with N_x rows, N_y columns, and N_z layers. The physical extent of the voxel structure can be determined by either the extent of ${}^{P}\mathbf{C}$ or by a smaller geofence. Each entry ${}^{P}\mathbf{G}_{m,n,q}$ of ${}^{P}\mathbf{G}$ indicates whether (${}^{P}\mathbf{G}_{m,n,q} = 1$) or not (${}^{P}\mathbf{G}_{m,n,q} = 0$) any point of ${}^{P}\mathbf{C}$ is located in the volume determined by the corresponding voxel. Herein, ${}^{P}\mathbf{G}$ will be referred to as the occupancy matrix. The blurred occupancy matrices have utility for improving performance in image-based edge detection [4, 14]. Point cloud voxelization is discussed further in Section 3.4.

Rasterization: Convert ${}^{P}\mathbf{G}$ from a 3D occupancy matrix to a 2D occupancy matrix (i.e., image) ${}^{P}\mathbf{M} \in \mathbb{B}^{N_x \times N_y}$, such that lines can be extracted from ${}^{P}\mathbf{M}$. Rasterization is discussed further in Section 3.5.

Point Cloud Alignement: Find the rotation matrix ${}^{H}_{P}\mathbf{R}^{0}$ such that the point cloud ${}^{P}\mathbf{C}$ can be transformed into a H-frame point cloud ${}^{H}\mathbf{C}$. The normal of each hatch edge plane in ${}^{H}\mathbf{C}$ is nom-

inally parallel to the N-frame x-axis or y-axis. Point cloud alignment is discussed further in Section 3.6.

Hatch Edge Extraction: Organize ${}^{H}\mathbf{C}$ into the x-blurred occupancy matrix ${}^{P}\mathbf{B}^{x}$ and y-blurred occupancy matrix ${}^{P}\mathbf{B}^{y}$. The blurred occupancy matrices have utility for improving performance in image-based edge detection [4, 14]. Then, Rasterizing the matrices ${}^{P}\mathbf{B}^{x}$ and ${}^{P}\mathbf{B}^{y}$ into ${}^{H}\mathbf{M}^{x}$ and ${}^{H}\mathbf{M}^{y}$, and processing them to extract lines approximately parallel to the H-frame x-axis and y-axis. Last, calculating the edge position and the hatch size from the extracted lines. Hatch edge extraction is discussed further in Section 3.7.

Voxelization is straightforward and widely used in the literature [32, 62]. The discussion of this chapter will be mainly on the point cloud alignment, rasterization, and hatch edge extraction.

3.4 Voxelization of Raw Point Cloud

The point density of the point cloud ${}^{P}\mathbf{S}$ is uneven, decreasing as the distance from the LiDAR increases. To achieve a desirable density near the edges of the processing region, the density may be too high in areas close to the scanning system. The process of voxelization [62] both organizes the point cloud and reduces the density where appropriate [20], without losing density in other areas. Furthermore, the occupancy matrix ${}^{P}\mathbf{G}$ provides an effective approach to retrieve points (i.e., voxels) within a small distance from a given voxel, or to extract all z-voxels with the same row and column indices, as will be necessary for rasterization to be discussed in Section 3.5.

The voxelization process is based on the geo-fence defined in eqn. (3.1) and the user-defined voxel cell size c. The minimum corner of ${}^{P}\mathbf{G}$ is ${}^{P}\underline{\mathbf{L}}_{S} = [\underline{x}_{S}, \underline{y}_{S}, \underline{h}_{e}]^{\top}$. The maximum corner of ${}^{P}\mathbf{G}$ is ${}^{P}\overline{\mathbf{L}}_{S} = [\overline{x}_{S}, \overline{y}_{S}, \overline{h}_{e}]^{\top}$. The number of voxels $\mathbf{N}_{G} = [N_{x}, N_{y}, N_{z}]^{\top}$ in each dimension of ${}^{P}\mathbf{G}$ is

calculated as:

$$\mathbf{N}_G = \left\lceil \frac{{}^P\overline{\mathbf{L}}_S - {}^P\underline{\mathbf{L}}_S}{c} \right\rceil. \tag{3.3}$$

The value of each cell ${}^P\mathbf{G}_{m,n,q}$ is initialized as 0.

Each point ${}^P\mathbf{p}_i \in {}^P\mathbf{S}$ is located within exactly one corresponding cell in ${}^P\mathbf{G}$. The row, column, and layer indices of the cell corresponding to ${}^P\mathbf{p}_i$ are calculated as:

$$[m_i, n_i, q_i]^\top = \left\lfloor \frac{{}^P\mathbf{p}_i - {}^P\underline{\mathbf{L}}_S}{c} \right\rfloor, \tag{3.4}$$

When ${}^P\mathbf{p}_i \in {}^P\mathbf{S}$ arrives, the value of ${}^P\mathbf{G}_{m_i,n_i,q_i}$ is changed from 0 (unoccupied) to 1 (occupied) because this cell is occupied by point ${}^P\mathbf{p}_i$. The occupancy matrix only tracks if a cell is occupied without saving the coordinates of the point(s) that occupies that cell. Multiple points located in the same cell leave the value of the voxel set to 1. Because the points in LiDAR scans are discretely spaced samples along reflective surfaces, some voxels in ${}^P\mathbf{G}$ can be non-occupied even when there are (undetected) objects in the region of those voxels. However, any occupied voxel means there was an object reflecting at least one point in its region.

When coordinates are required corresponding to any voxel, a point cloud can be generated from the 3D occupancy matrix ${}^P\mathbf{G}$. Any cell ${}^P\mathbf{G}_{m_k,n_k,q_k}$ that has value 1 (occupied) can be converted into a 3D point by:

$${}^P\mathbf{p}_k = {}^P\underline{\mathbf{L}}_S + \left[m_k + \frac{1}{2}, n_k + \frac{1}{2}, q_k + \frac{1}{2} \right]^\top c, \tag{3.5}$$

where ${}^P\mathbf{p}_k$ is the point located at the center of cell ${}^P\mathbf{G}_{m_k,n_k,q_k}$. The point cloud corresponding to ${}^P\mathbf{G}$ is the output of a voxel filter, which is only used for visualization.

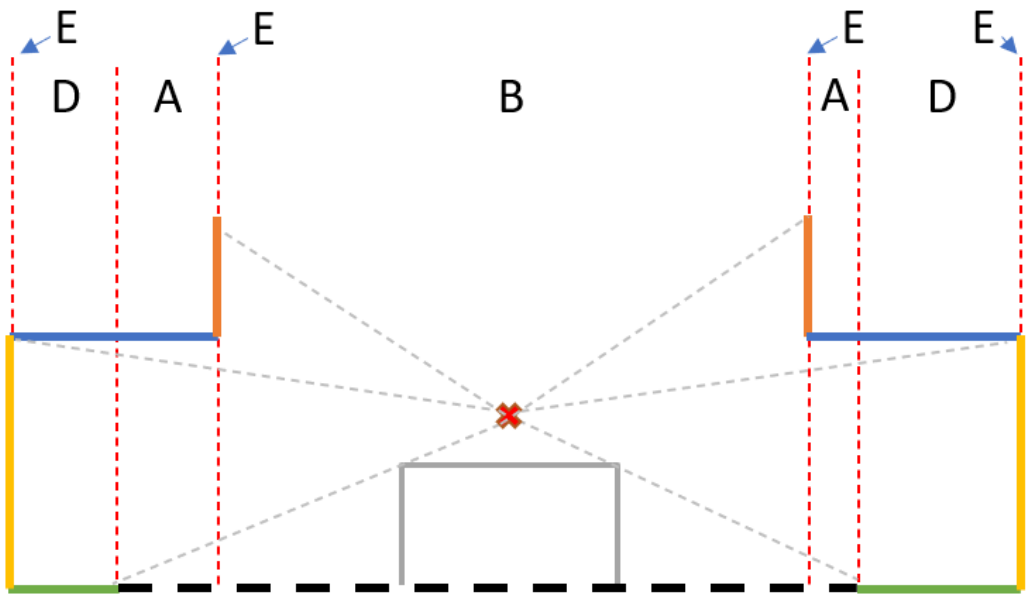

Figure 3.3: Schematic cross-section of the H-frame y-z plane depicting the scanned surfaces of the container from the robot.

3.5 Rasterization of Raw Point Cloud

Example surfaces corresponding to the point clouds scanned by the robot is depicted in Figure 3.3. The red cross indicates the LiDAR position. The solid green, orange, and blue curves represent LiDAR points reflected from the floor, hatch edge plane, and ceiling, respectively. The gray box only indicates the position of the robot, which is not scanned. The gray dash lines indicate various LiDAR ray-tracings. The black dash lines indicate surfaces that are unscanned due to occlusion. The red dash lines are separators between different types of pillar regions for rasterization. Those pillar regions are defined and discussed in next.

All entries ${}^{P}\mathbf{G}_{m,n,q}$ of ${}^{P}\mathbf{G}$ that have the same m and n are considered as one occupancy vector ${}^{P}\mathbf{V}^{x}_{m,n} = \{{}^{P}\mathbf{G}_{m,n,1}, ..., {}^{P}\mathbf{G}_{m,n,N_z}\} \in \mathbb{B}^{N_z}$. For each index pair (m, n), where $m = 1, ...N_x$, $n = 1, ...N_y$, the occupancy vector ${}^{P}\mathbf{V}^{x}_{m,n}$ physically represents the occupancy status of a pillar region

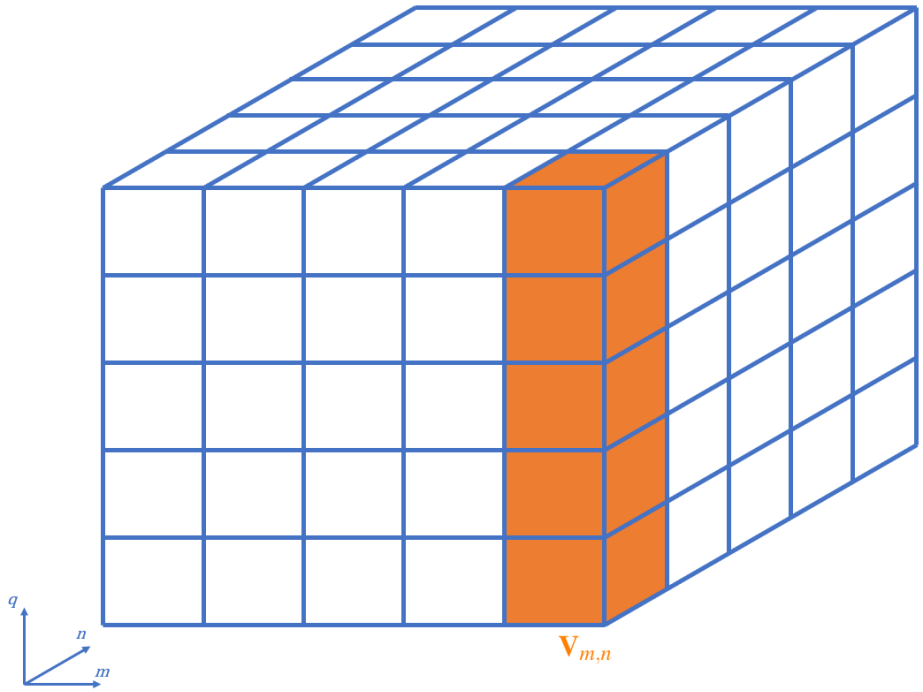

Figure 3.4: Schematic of a voxel grid and a pillar region. E.g., the orange voxels make one pillar region $V_{m,n}$ with $m = 5, n = 1$.

in ${}^{P}\mathbf{G}$. An example is shown in Figure 3.4. There are five types of pillar regions, with examples shown in Figure 3.3. Type-A includes only one cluster of points from the ceiling. Type-B is an empty region, due to occlusion. Type-C only includes one cluster of points from the floor. Type-D includes two clusters of points from both the ceiling and the floor. Type-E includes points from the hatch edge plane segment and floor.

This section presents an approach to define the boolean value of each image pixel ${}^{P}\mathbf{M}_{m,n}$ by processing the occupancy vector ${}^{P}\mathbf{V}_{m,n}$. The goal is to set ${}^{P}\mathbf{M}_{m,n}$ to 1 for type E and 0 for types A-D. For the rest of this section, the subscripts m and n, and the super-script P of ${}^{P}\mathbf{V}_{m,n}$ are dropped to simplify notation. Therefore, the representation of the occupancy vector ${}^{P}\mathbf{V}_{m,n}$ is simplified in this section as $\mathbf{V} = \{v_1, ..., v_{N_z}\}$.

The occupancy status in $\mathbf{V}$ is naturally ordered from the smallest to the largest z. The value of each element in $\mathbf{V}$ indicates if any surface was scanned in the voxel corresponding to that z-value in the pillar. The property that LiDAR points are discretely spaced in ${}^{P}\mathbf{C}$ also applies to $\mathbf{V}$. Let $l^{V}_{m,n}$ be the length of the longest block of consecutively filled cells in $\mathbf{V}_{m,n}$, allowing for gaps with tolerance of b_v pixels, where b_v is a user-defined non-negative integer. For example, given a vector $\mathbf{V} = [0, 1, 1, 0, 1, 0, 0, 1]^{\top}$ and a tolerance on one '0' for a gap, there are two blocks in the vector, i.e., the 2-nd to 5-th terms and the 8-th term. Therefore, the length of the longest block $l^{V}_{m,n}$ equals the length of the first block: $l^{V}_{m,n} = 4$. An occupancy vector $\mathbf{V}$ of the pillar region of Type-E is expected to have a higher $l^{V}_{m,n}$ than a pillar in regions of types A-D. With a lower bound b_e on the z-direction extent of the hatch edge, the value of $l^{V}_{m,n}$ is expected to have a value of at least $\frac{b_e}{c}$ for a pillar in a Type-E region. Therefore, the value of each ${}^{P}\mathbf{M}_{m,n}$ in ${}^{P}\mathbf{M}$ is:

$$ {}^{P}\mathbf{M}_{m,n} = \begin{cases} 1, & \text{if } l^{V}_{m,n} \geq \frac{b_e}{c} \\ 0, & \text{otherwise.} \end{cases} \tag{3.6} $$

The '1's in ${}^{P}\mathbf{M}$ indicate those entries detected as candidate points for vertical walls. Those candidate points will be used to detect lines in Section 3.6.2.

3.6 Raw Point Cloud Alignment

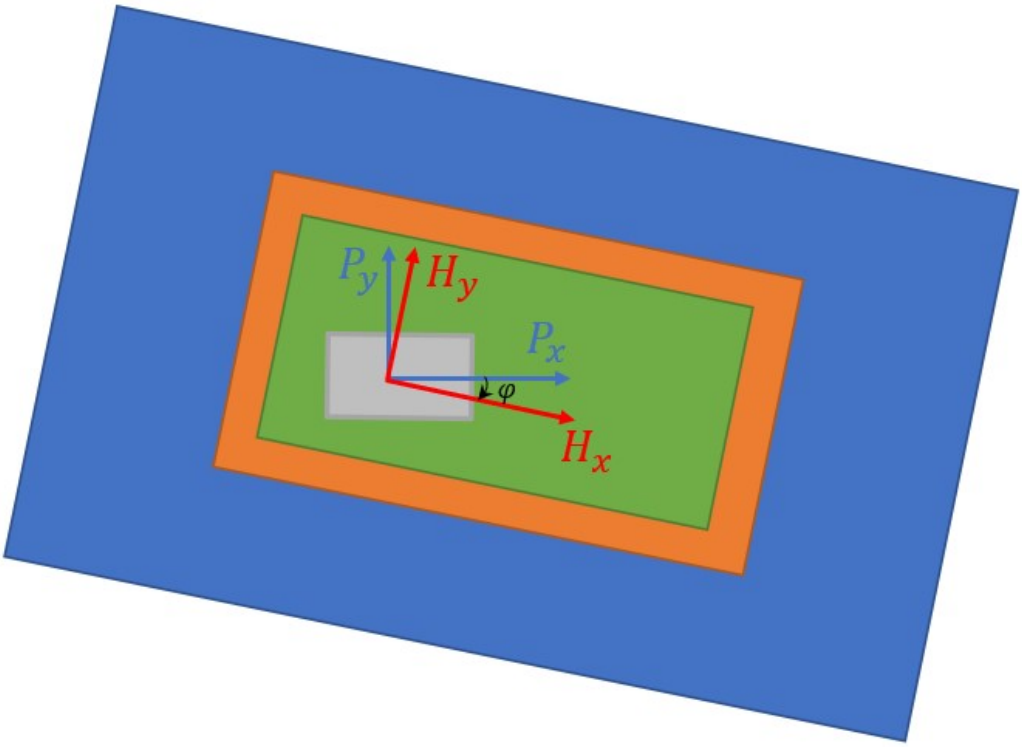

Figure 3.5: Schematic for the top-view of the point cloud ${}^{P}\mathbf{C}$ when referring to the robot orientation.

When the robot is initially placed in the room, the robot orientation is not precisely controlled. Therefore, the hatch rectangle usually does not align to the P-frame axes. The schematic top-view of the accumulated point cloud is shown in Figure 3.1 (Top). When referring to the robot orientation (i.e., P-frame axis directions), the schematic in Figure 3.1 can be rotated such that P-frame x-axis points right and P-frame y-axis points up in the page. The value of the rotation angle is the same with the value of the angle-of-misalignment φ as shown in Figure 3.5. With the Assumption 6, the rotation matrix ${}^{H}_{P}\mathbf{R}^{0}$ represents a rotation around the P-frame z-axis. This section discusses the approaches to determine the rotation matrix ${}^{H}_{P}\mathbf{R}^{0}$. Section 3.6.1 discusses a method to manually pick a few key points in ${}^{P}\mathbf{C}$ and calculate the matrix ${}^{H}_{P}\hat{\mathbf{R}}^{0}$. Section 3.6.2 discusses a

method to estimate the angle-of-misalignment $\hat{\varphi} = \varphi + \delta_\varphi$, where δ_φ is the difference between the estimated value and truth. Then the rotation matrix is calculated with $\hat{\varphi}$ as:

$$ {}^{H}_{P}\hat{\mathbf{R}}^{0} = \begin{bmatrix} \cos(\hat{\varphi}) & \sin(\hat{\varphi}) & 0 \\ -\sin(\hat{\varphi}) & \cos(\hat{\varphi}) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.7} $$

Each point of ${}^{P}\mathbf{C}$ can be transformed into H-frame by eqn. (2.4). When the alignment rotation matrix is calculated with either method in Sections 3.6.1 and 3.6.2 , the transformed point cloud ${}^{H}\mathbf{C}$ is computed as:

$$ {}^{H}\mathbf{C} = {}^{H}_{P}\hat{\mathbf{R}}^{0}\ {}^{P}\mathbf{C}. \tag{3.8} $$

3.6.1 Alignment Rotation Matrix: Manual Pick

After the robot scans the point cloud ${}^{P}\mathbf{C}$, the shape and density of the point cloud is sufficient for a human to distinguish objects and structure of the container. A straightforward approach is for the operator to manually pick four points in P-frame, which can be used to calculate three vectors corresponding to the axes of P-frame. The rotation matrix ${}^{H}_{P}\mathbf{R}^{0}$ satisfies:

$$ \begin{bmatrix} {}^{H}\mathbf{e}_x & {}^{H}\mathbf{e}_y & {}^{H}\mathbf{e}_z \end{bmatrix} = {}^{H}_{P}\mathbf{R}^{0} \begin{bmatrix} {}^{P}\mathbf{e}_x & {}^{P}\mathbf{e}_y & {}^{P}\mathbf{e}_z \end{bmatrix}, \tag{3.9} $$

where ${}^{H}\mathbf{e}_x = [1,0,0]^\top$, ${}^{H}\mathbf{e}_y = [0,1,0]^\top$, and ${}^{H}\mathbf{e}_z = [0,0,1]^\top$ are the three axis direction vectors of H-frame represented in H-frame. The unit vectors ${}^{P}\mathbf{e}_x$, ${}^{P}\mathbf{e}_y$, and ${}^{P}\mathbf{e}_z$ are the same H-frame axis direction vectors represented in P-frame at time $t_k = 0$.

Given the point cloud ${}^{P}\mathbf{C}$, the hatch corner points p_a^1, p_b^0, p_b^1, $p_c^1 \in \mathscr{R}^3$ (see Figure 1.1) can be manually selected from ${}^{P}\mathbf{C}$. The three edges passing p_b^1 can be represented as: $\mathbf{v}_{ba} = p_a^1 - p_b^1$, $\mathbf{v}_{bc} = p_c^1 - p_b^1$, and $\mathbf{v}_b^{10} = p_b^0 - p_b^1$. Note that $\mathbf{v}_b^{10}$ is nominally $[0, 0, 1]^\top$ because of Assumption 6. The vectors $\mathbf{v}_{ba}$, $\mathbf{v}_{bc}$ and $\mathbf{v}_b^{10}$ are perpendicular to each other because they are each parallel to an axis. Therefore, the axis direction vectors in P-frame are calculated as:

$$\begin{bmatrix} {}^{P}\hat{\mathbf{e}}_x & {}^{P}\hat{\mathbf{e}}_y & {}^{P}\hat{\mathbf{e}}_z \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{v}_{ba}}{\|\mathbf{v}_{ba}\|} & \frac{\mathbf{v}_{bc}}{\|\mathbf{v}_{bc}\|} & \frac{\mathbf{v}_b^{10}}{\|\mathbf{v}_b^{10}\|} \end{bmatrix}. \tag{3.10}$$

Substitute the calculated axis direction vectors into eqn. (3.9) yields:

$$\begin{bmatrix} {}^{H}\mathbf{e}_x & {}^{H}\mathbf{e}_y & {}^{H}\mathbf{e}_z \end{bmatrix} = {}^{H}_{P}\mathbf{R}^0 \begin{bmatrix} \frac{\mathbf{v}_{ba}}{\|\mathbf{v}_{ba}\|} & \frac{\mathbf{v}_{bc}}{\|\mathbf{v}_{bc}\|} & \frac{\mathbf{v}_b^{10}}{\|\mathbf{v}_b^{10}\|} \end{bmatrix}, \tag{3.11}$$

therefore, the rotation matrix ${}^{H}_{P}\hat{\mathbf{R}}^0$ is calculated as:

$${}^{H}_{P}\hat{\mathbf{R}}^0 = \begin{bmatrix} {}^{H}\mathbf{e}_x & {}^{H}\mathbf{e}_y & {}^{H}\mathbf{e}_z \end{bmatrix} \left(\begin{bmatrix} \frac{\mathbf{v}_{ba}}{\|\mathbf{v}_{ba}\|} & \frac{\mathbf{v}_{bc}}{\|\mathbf{v}_{bc}\|} & \frac{\mathbf{v}_b^{10}}{\|\mathbf{v}_b^{10}\|} \end{bmatrix} \right)^{-1} \tag{3.12}$$

The advantage of the manual-pick method is that this method can reliably return the rotation matrix ${}^{H}_{P}\hat{\mathbf{R}}^0$ as long as the operator can find the correct hatch corners. Also, this approach does not require the initial orientation of the robot. However, this method requires human intervention which is time-inefficient and preferred to be avoided.

3.6.2 Alignment Rotation Matrix: Manual Manipulation

This section introduces a method to automate the process of determining the angle-of-misalignment φ based on Assumption 6. When the robot is initially placed in the room, its heading can be roughly aligned to a predetermined direction (i.e., the direction of H-frame x-axis) with φ less than $\frac{\pi}{4}$ radian as shown in Figure 3.5. In this application, most of the walls inside the container,

and the hatch edge planes are perpendicular to the bottom of the container (see Figure 3.1 (Bottom)), and these walls inside the container (yellow) and the hatch edge planes (orange) are also aligned. These two types of plane segments make type-E pillar regions, which produces 1-pixel's during rasterization and leads to an image ${}^{P}\mathbf{M}$ as shown in Figure 3.6.

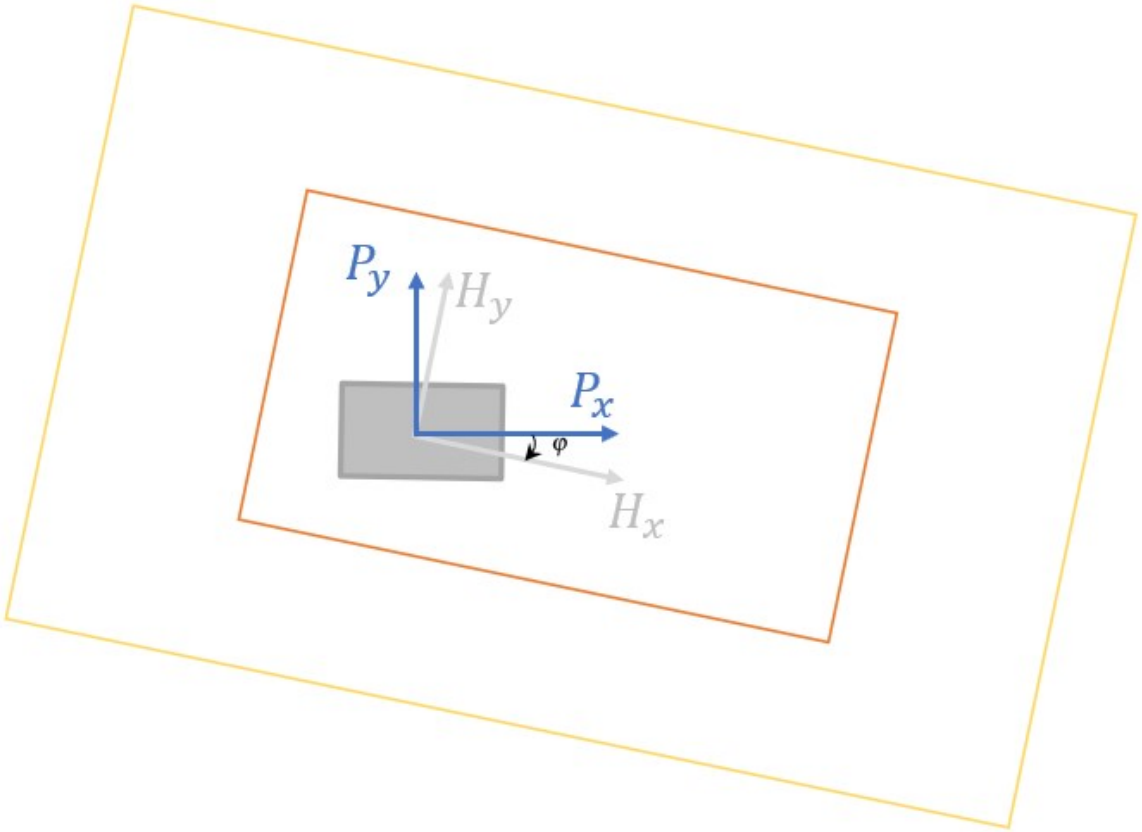

Figure 3.6: Schematic for the P-frame image ${}^{P}\mathbf{M}$. The grey rectangle and the axes are indicating the robot orientation, not in the image. The color of the orange and yellow rectangles only indicates they are 1-pixel's. There is not a real-value for each pixel.

The HT algorithm is set up for extracting the edge lines of the orange and yellow rectangles. The HT algorithm, as introduced in Section 2.6, is used to extract the lines of the orange and yellow rectangle edges in Figure 3.6. Because the angle-of-misalignment φ is in the range of $(-\frac{\pi}{4}, \frac{\pi}{4})$ by manual manipulation, the angle-range of the HT voting table is set to $\theta_x^0 \in (-\frac{\pi}{4}, \frac{\pi}{4})$ to extract

lines almost parallel to H-frame x-axis and $\theta_y^0 \in (\frac{\pi}{4}, \frac{3\pi}{4})$ to extract lines almost parallel to H-frame y-axis. The super-scripts of θ_x^0 and θ_y^0 indicate the ranges are set for the HT in the initialization stage. The resolution-of-distance Δ_ρ is set to the cell size c and the resolution-of-angle Δ_θ is set to $\frac{c}{\bar{\ell}}$.

When searching in the image ${}^P\mathbf{M}$ with HT range as θ_x^0, the HT algorithm returns an array of N_x^0 line candidates. Each of these line candidates is represented as (ρ_i, θ_i, u_i), where ρ_i is the line distance, u_i is the number of votes and θ_i is a line angle between -$\frac{\pi}{4}$ and $\frac{\pi}{4}$. Similarly, a second array of N_y^0 line candidates are extracted from ${}^P\mathbf{M}$ with the HT angle range as θ_y^0, which are lines nominally extracted from the rectangle edges parallel to H-frame y-axis. Each line of the second array is denoted as (ρ_j, θ_j, u_j). Because the x and y axes are perpendicular, rotating each line of (ρ_j, θ_j, u_j) by 90 degrees (i.e., $\tilde{\theta}_j = \theta_j - \frac{\pi}{2}$) should result in a line that is nearly parallel to H-frame x-axis, which can be represented as $(\rho_j, \tilde{\theta}_j, u_j)$.

The candidate line angle θ_i and $\tilde{\theta}_j$ are expected to be close to the angle-of-misalignment φ. A vector of candidate angle-of-misalignment $\Theta = [\underline{\theta}, \underline{\theta} + \Delta_\theta, \ldots, \bar{\theta}]$ is defined. The range of Θ is determined as the minimum and maximum angle among all line candidates, respectively:

$$\underline{\theta} = \min\{\min\{\theta_i : i = 1, \ldots, N_x^0\},\ \min\{\tilde{\theta}_j : j = 1, \ldots, N_y^0\}\}, \tag{3.13}$$

$$\bar{\theta} = \max\{\max\{\theta_i : i = 1, \ldots, N_x^0\},\ \max\{\tilde{\theta}_j : j = 1, \ldots, N_y^0\}\}. \tag{3.14}$$

A vector of votes $\mathbf{U} = \{U_k : k = 1, \ldots, N_u\}$ is defined with the same length $N_u = \frac{\bar{\theta} - \underline{\theta}}{\Delta_\theta}$ of Θ to count votes. Each element U_k represents the number of votes for k-th candidate angle $\Theta_k = \underline{\theta} + (k-1)\Delta_\theta$ in Θ. The value of U_k is the sum of the votes of all line candidates with the same angle with Θ_k:

$$U_k = \sum_i u_i[\theta_i = \Theta_k] \quad + \quad \sum_j u_j[\tilde{\theta}_j = \Theta_k], \tag{3.15}$$

where

$$[\theta_i = \Theta_k] = \begin{cases} 1 & \text{if } \theta_i = \Theta_k \\ 0 & \text{else} \end{cases}, \quad [\tilde{\theta}_j = \Theta_k] = \begin{cases} 1 & \text{if } \tilde{\theta}_j = \Theta_k \\ 0 & \text{else} \end{cases}. \tag{3.16}$$

The candidate angle Θ^*, which has the most votes $U^* = \max(\mathbf{U})$, is determined as the misalignment angle $\hat{\varphi} = \Theta^*$. The alignment rotation matrix ${}^{H}_{P}\hat{\mathbf{R}}^{0}$ is calculated by substituting $\hat{\varphi}$ into eqn. (3.6):

$${}^{H}_{P}\hat{\mathbf{R}}^{0} = \begin{bmatrix} \cos(\hat{\varphi}) & \sin(\hat{\varphi}) & 0 \\ -\sin(\hat{\varphi}) & \cos(\hat{\varphi}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

3.7 Hatch Edge Extraction

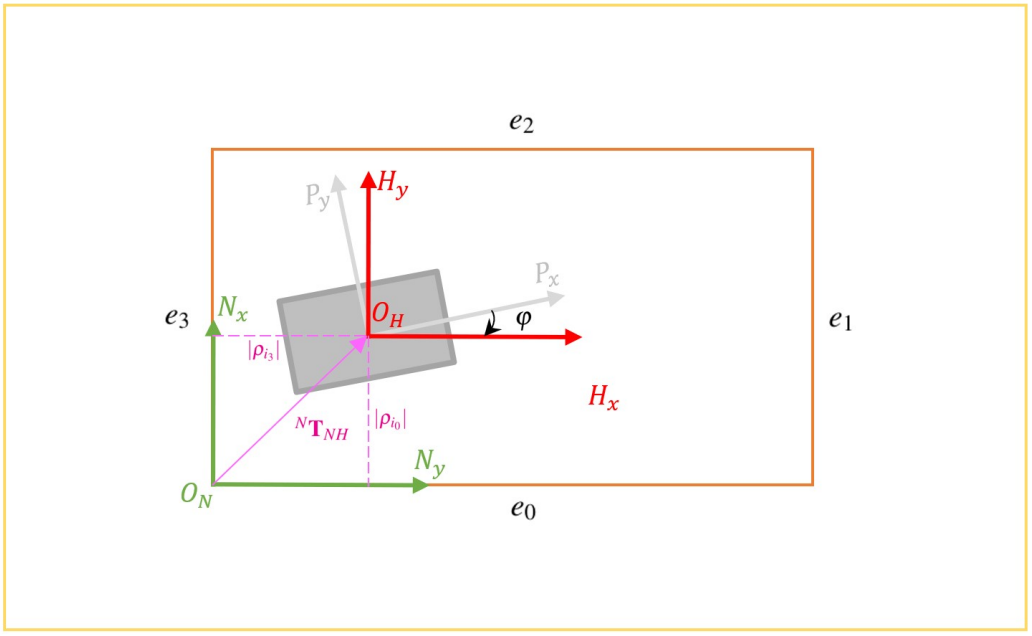

Figure 3.7: Schematic for the H-frame image ${}^{H}\mathbf{M}$. This image is produced by rotating the image ${}^{P}\mathbf{M}$.

The schematic of the image ${}^{H}\mathbf{M}$ is shown in Figure 3.7, which is the result of rotating ${}^{P}\mathbf{M}$ with eqn. (3.8). This section discusses using HT to extract the hatch edges, which are represented as the orange lines in ${}^{H}\mathbf{M}$ (see Figure 3.7). The hatch x-edges will be represented in the image ${}^{H}\mathbf{M}$ as vertical lines and y-edges in ${}^{H}\mathbf{M}$ as horizontal lines. The method in this section is presented only for extracting the x-edges. The approach applies to the y-edges extraction by analogy.

The hatch edge lines are represented by the eqn. (3.2), where each line is parameterized with a pair of (ρ, θ). The angular range of the HT is $\theta_x \in (-\delta_\theta, \delta_\theta)$ for x-edges, where δ_θ is a small angle determined considering the residual of the angle-of-misalignment δ_φ after the alignment (i.e., $\delta_\varphi \in (-\delta_\theta, \delta_\theta)$). The resolution-of-distance Δ_ρ is set to the cell size c and the resolution-of-angle Δ_θ is set to $\frac{c}{\ell}$.

All ${}^{H}\mathbf{M}$ image pixels with value 1 (i.e., the orange and yellow pixels in Figure 3.7) participate in HT voting for candidate lines. The parameters of the extracted candidate lines with sufficient votes are denoted as (ρ_i^x, θ_i^x) for $i = 1, \ldots, N_E^x$, where N_E^x is the number of extracted vertical lines from ${}^{H}\mathbf{M}$ and the lines are sorted in ascending order based on the value of line parameters ρ_i^x: $\rho_1^x \leq \ldots \leq \rho_{N_E^x}^x$. Based on Assumption 1, the two lines nearest to and on opposite sides of the origin should correspond to the hatch edges e_1 and e_3. Define

$$i_1 = \arg\max_{\rho_i > 0}(\rho_i) \quad \text{and} \quad i_3 = \arg\min_{\rho_i < 0}(\rho_i).$$

The line defining e_1 is parameterized as $(\rho_{i_1}, \theta_{i_1})$. The line defining e_3 is parameterized as $(\rho_{i_3}, \theta_{i_3})$.

The edges parallel to the P-frame y-axis are similarly extracted to define e_0 and e_2 by processing ${}^{H}\mathbf{M}$ in a manor analogous to that described above. The angle range of the HT for extracting y-edges is $\theta_y \in (\frac{\pi}{2} - \delta_\theta, \frac{\pi}{2} + \delta_\theta)$.

Because the edge lines in the aligned image ${}^{H}\mathbf{M}$ is either vertical or horizontal, taking a histogram of point number over the H-frame x-direction (y-direction) can also determine the distance from the edges to the origin in the x-direction (y-direction). When taking a histogram of the point number in the x-direction in Figure 3.7, there should be four spikes in the x-direction. Each of the spikes corresponds to one vertical line. Therefore, the distance from the edges to the origin can be determined by detecting the spikes and taking the closest two spikes on opposite sides of the origin in the histogram. The corresponding x values of the these two spikes can determine the distance of the two edges e_1 and e_3. Similar method can be applied in the y-direction to determine the distance of the other two edges e_0 and e_2. The method of this paragraph is not used in the dissertation but can be an alternative of the method in this section to extract the hatch edges.

3.8 Localization Initialization

This section focuses on the establishment of the navigation frame (i.e., N-frame) and the determination of the initial robot pose (i.e., orientation and position) based on the point cloud alignment results in Section 3.6 and hatch extraction results of Section 3.7. The navigation map is built after the determination of the initial position and orientation.

The initial position of the robot in N-frame can be represented by a vector from the N-frame origin to the P-frame origin: ${}^{N}\mathbf{T}^{0}_{NP}$. The initial orientation of the robot in N-frame can be represented by a rotation matrix for transforming a point from P-frame to N-frame: ${}^{N}_{P}\mathbf{R}^{0}$. The right superscripts of ${}^{N}\mathbf{T}^{0}_{NP}$ and ${}^{N}_{P}\mathbf{R}^{0}$ are 0 indicating that they are the initial position and orientation to be determined in this section.

The initial position ${}^{N}\mathbf{T}^{0}_{NP}$ can be manipulated as:

$$ {}^{N}\mathbf{T}^{0}_{NP} = {}^{N}\mathbf{T}_{NH} + {}^{N}\mathbf{T}^{0}_{HP}, \tag{3.18} $$

where ${}^{N}\mathbf{T}^{0}_{HP}$ is the vector from H-frame origin to P-frame origin during initialization. Therefore, the vector ${}^{N}\mathbf{T}^{0}_{HP} = \mathbf{0}$. The another vector ${}^{N}\mathbf{T}_{NH}$ is the vector from N-frame origin to H-frame origin represented in N-frame.

In Section 3.7, the line corresponding to edge e_0 is extracted and denoted as $(\rho_{i_0}, \theta_{i_0})$, and the line corresponding to edge e_3 is extracted and denoted as $(\rho_{i_3}, \theta_{i_3})$. Therefore, as shown in Figure 3.7, the vector ${}^{N}\mathbf{T}_{NH}$ is determined as:

$$ {}^{N}\mathbf{T}_{NH} = \begin{bmatrix} |\rho_{i_0}|, & |\rho_{i_3}|, & 0 \end{bmatrix}^{\top}, \tag{3.19} $$

where the third element is zero by definition in Section 2.2. Substituting eqn. (3.19) into eqn. (3.18) yields:

$$ \begin{aligned} {}^{N}\hat{\mathbf{T}}^{0}_{NP} &= {}^{N}\mathbf{T}_{NH} + \mathbf{0} \\ &= \begin{bmatrix} |\rho_{i_0}|, & |\rho_{i_3}|, & 0 \end{bmatrix}^{\top}, \end{aligned} \tag{3.20} $$

which is the result of the initial position of the robot in N-frame.

The rotation matrix for initial orientation ${}^{N}_{P}\mathbf{R}^{0}$ can be manipulated as:

$$ {}^{N}_{P}\mathbf{R}^{0} = {}^{N}_{H}\mathbf{R}\,{}^{H}_{P}\mathbf{R}, \tag{3.21} $$

where ${}^{N}_{H}\mathbf{R}$ is known from eqn. (2.8) and ${}^{H}_{P}\mathbf{R}$ is determined in Section 3.6 as ${}^{H}_{P}\hat{\mathbf{R}}^{0}$. Therefore, the initial orientation can be calculated as:

$$ {}^{N}_{P}\hat{\mathbf{R}}^{0} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} {}^{H}_{P}\hat{\mathbf{R}}^{0} \quad . \tag{3.22} $$

The navigation map herein is the same point cloud of ${}^{P}\mathbf{C}$ represented in N-frame: ${}^{N}\mathbf{C}$. Each point ${}^{P}\mathbf{p}_i \in {}^{P}\mathbf{C}$ is transformed into a N-frame point ${}^{N}\mathbf{p}_i$ using eqn. (2.5) with $k = 0$:

$$ {}^{N}\mathbf{p}_i = {}^{N}_{P}\mathbf{R}^{0}\ {}^{P}\mathbf{p}_i + {}^{N}\mathbf{T}^{0}_{NP}, \tag{3.23} $$

where the initial rotation matrix ${}^{N}_{P}\mathbf{R}^{0}$ and translation vector ${}^{N}\mathbf{T}^{0}_{NP}$ are determined in eqns. (3.20) and (3.22). Therefore, the map ${}^{N}\mathbf{C}$ is the result of transforming every points in ${}^{P}\mathbf{C}$:

$$ {}^{N}\mathbf{C} = {}^{N}_{P}\hat{\mathbf{R}}^{0}\ {}^{P}\mathbf{C} + {}^{N}\hat{\mathbf{T}}^{0}_{NP}, \tag{3.24} $$

3.9 Summary of Constant Parameters

This section discusses the various constants involved in the method and provides advice about selecting their values.

Geofence parameters $\underline{x}_S$, $\overline{x}_S$, $\underline{y}_S$, $\overline{y}_S$, $\underline{h}_e$, $\overline{h}_e$ $(meters)$: These constants are used in eqn. (3.1) to define the geofence. The purpose is to limit the extent of the accumulated point cloud while retaining all point reflected from surfaces depicted in Figure 3.1. The values are user-defined based on local knowledge of the container size.

Cell size c $(meters)$**:** The cell size is used in the definition of the three occupancy matrices ${}^{P}\mathbf{G}$, ${}^{H}\mathbf{B}^{x}$, and ${}^{H}\mathbf{B}^{y}$. It determines the resolution of these voxelized point clouds. For a given point cloud region, the memory requirements of each occupancy matrix scales cubically with $\frac{1}{c}$. Larger structures also require longer processing time. The cell size needs to be small enough to meet the localization accuracy requirement and large enough to be comparable with the accuracy of the raw data. Herein, the cell size is $10cm$ through all experiments which is comparable with the accuracy of the raw LiDAR range data.

Parameters for gaps: b_b, b_v $(pixels)$**:** The purpose of these parameters is to improve edge extraction performance [4, 14]. These constants are used in Sections 3.5 and 3.7.

Hatch edge plane depth lower bound b_e $(meters)$**:** This constant is used in eqn. (3.6) to check whether each occupancy vector (i.e., pillar) is deep enough in the z-direction to be regarded as a piece of a hatch edge plane. It needs to be large enough to reject small plane segments and small enough so that hatch edge points are recognized.

Hough-Transform angular search range δ_θ $(radians)$**:** This constant defines the searching range of HT. It is used in Sections 3.6.2 and 3.7 to limit the HT angular search to a proper range. In Section 3.6.2, this constant must be large enough to account for misalignment of the hatch edges with the P-frame x and y axes. It should be smaller than 45 degrees in Section 3.6.2 to avoid finding unexpected lines. In Section 3.7, this constant can be a small value because of the point cloud alignment. As it is increased, the HT computation time will increase.

Chapter 4

Frame Definition and Pose Initialization Experiments

This chapter discusses the experimental setup and results. For the hatch extraction objective, the sensors used is the LiDAR and PT. The LiDAR used to perform the experiments is a RS-BPearl (RSBP). The PT used is a normal industrial PT which can rotate up to 15 degrees per second.

The robot is first placed into the working area as in Figure 3.1 (top), then the robot remains stationary and the PT is at zero position (i.e., both pan and tilt angle are zero). The point cloud returned by the LiDAR at this time is mainly the upper part in Figure 3.1 (bot) (i.e., hatch wall, celing surface, and partial room wall). Once the point cloud accumulation starts, the PT will drive the LiDAR to rotate. First, the head will adjust the tilt angle to 75 degrees, and at this angle the LiDAR will scan mainly the right half of Figure 3.1 (bot) (i.e., the right side of the hatch wall, ceiling, walls, and floor). Next, the PT tilt angle is kept constant, the PT pans for two cycles (i.e.,

720 degrees) and then returns to zero position to end the point cloud accumulation. Then, the approach introduced in this chapter is applied to process the accumulated point cloud.

The choice of PT rotation pattern or sensor for this approach is not strict, as long as it generates a dense enough point cloud to meet the requirements.

4.1 Point Cloud Model

This section focuses on showing the data processed by the hatch extraction approach. Figure 4.1 shows a side view of the voxelized P-frame accumulated point cloud ${}^{P}\mathbf{G}$, where the coordinate system in the lower left corner only indicates the directions of the P-frame axes. Figure 4.2 shows the top view of ${}^{P}\mathbf{G}$, where the coordinate system indicates the P-frame origin position and the coordinate axis direction.

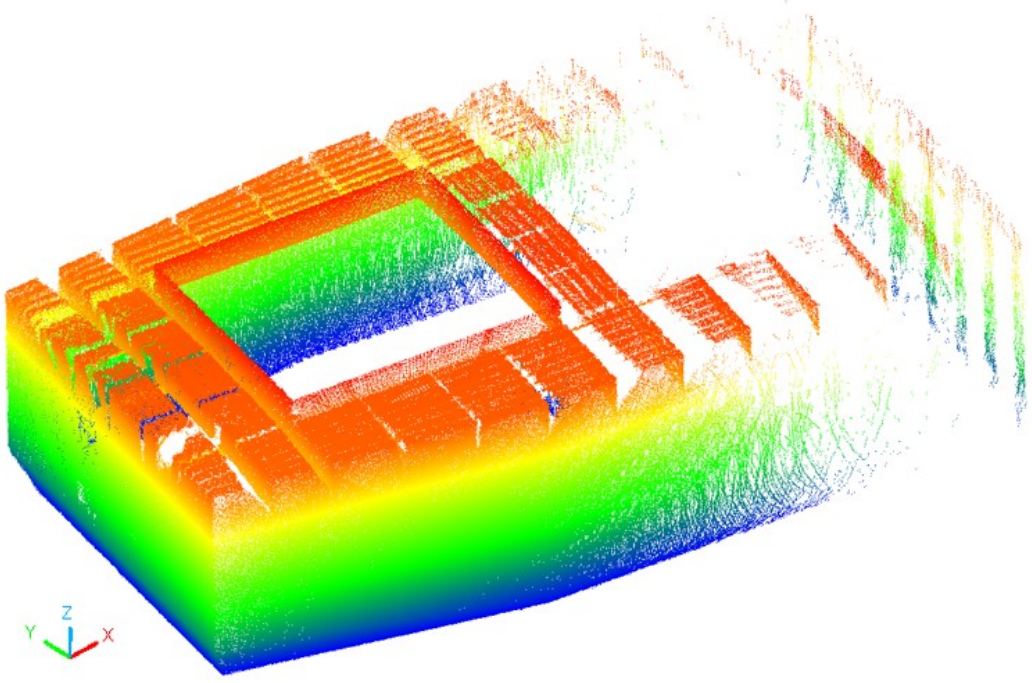

Figure 4.1: The voxelized P-frame accumulated point cloud ${}^{P}\mathbf{G}$. (The axes in the figure indicate only the direction, not the origin of P-frame.)

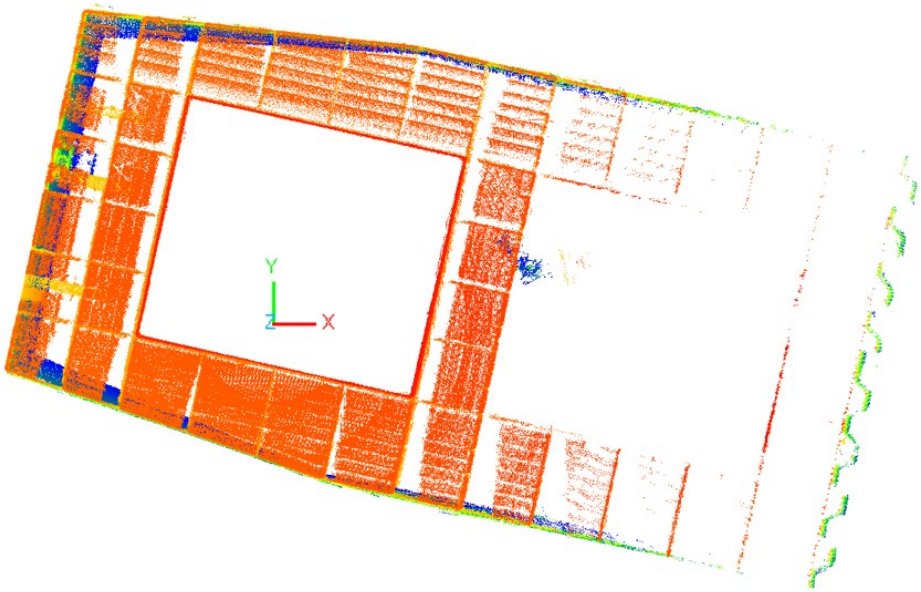

Figure 4.2: The top-view of the voxelized P-frame accumulated point cloud ${}^{P}\mathbf{G}$. (The axes in the figure indicate both the direction and origin of P-frame.)

The color of the point cloud is rendered using the z-value of each point for display purposes. The color of the points changes from blue to red, indicating the increase in z-value. Since the points on the floor are not needed for the hatch extraction and are not used to build the map for later positioning, the points with small z-values are excluded by the geofence in eqn. 3.1 and are not shown in Figures 4.1 and 4.2. If there is a need to manage the floor point cloud, the part of the points with small z-values can be saved separately in a separate point cloud.

4.2 Rasterization Results

With the voxelized point cloud ${}^{P}\mathbf{G}$, which is shown in Figures 4.1 and 4.2, this section discusses the results of rasterizing ${}^{P}\mathbf{G}$ into a black-and-white image. Note that the number of

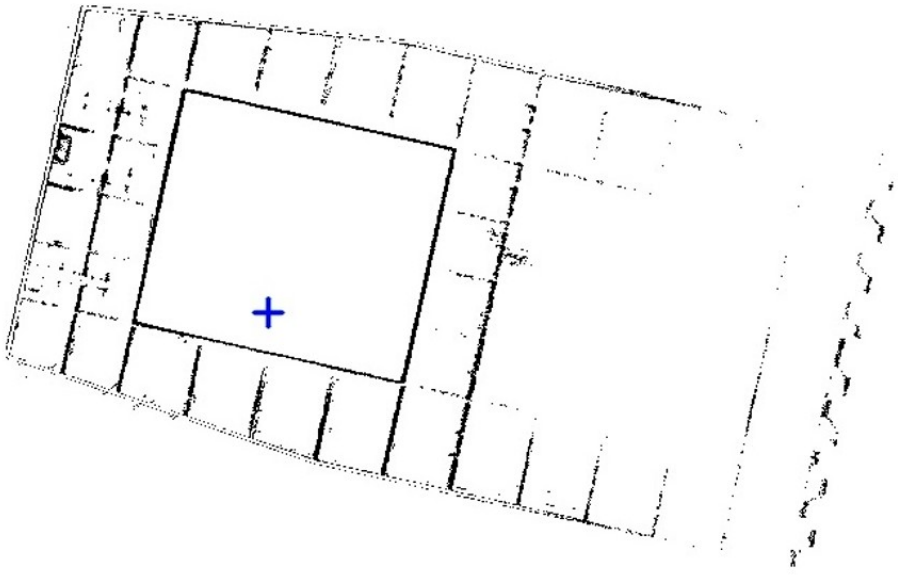

Figure 4.3: The image ${}^{P}\mathbf{M}$ rasterized from the point cloud ${}^{P}\mathbf{G}$.

hatch edge points is a small percentage of the overall point cloud ${}^{P}\mathbf{G}$. Therefore, methods such as RANSAC are unlikely to succeed.

The rasterization result ${}^{P}\mathbf{M}$ is obtained by applying the approach discussed in Section 3.5. The image ${}^{P}\mathbf{M}$ is shown in Figure 4.3. The blue cross in the Figure 4.3 represents the position of the P-frame origin in the image. Each pixel indicates if the pixel correspond to a 10×10 cm^2 type-E pillar region. The parameter b_e is 1 meter, which means each black pixel represents a pillar region containing a continuous length of at least 1 meter in this pillar region.

4.3 Alignment Results

The HT algorithm is applied twice to extract lines from the angular ranges θ_x^0 and θ_y^0 as discussed in Section 3.6.2. The angular resolution $\Delta_\theta = \frac{c}{\bar{\ell}} = \frac{0.1}{30} \approxeq 0.003$ rad, where $\bar{\ell}$ is the approximate hatch length in this application. In both line extraction, the extracted line candidates that are

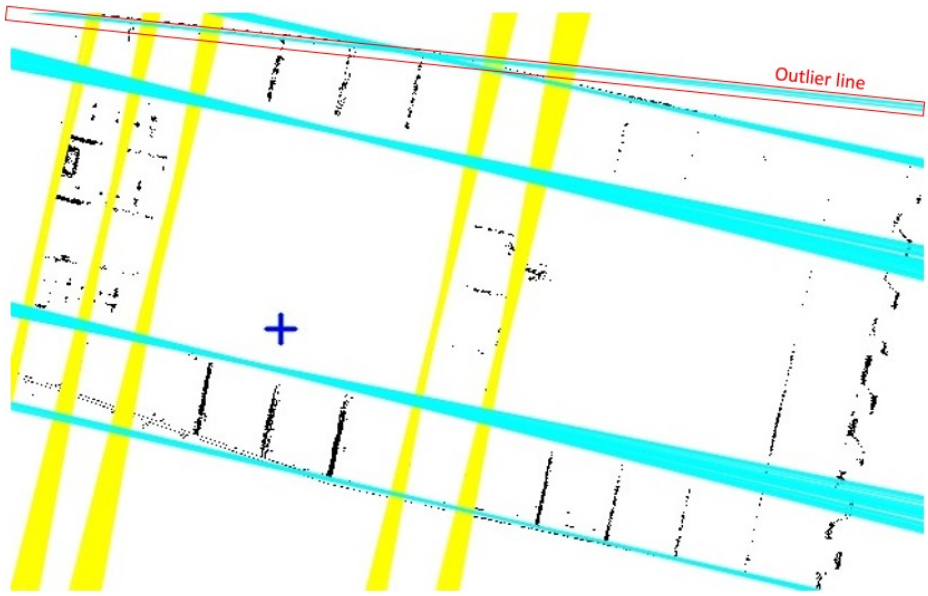

Figure 4.4: The image ${}^{P}\mathbf{M}$ with lines separately extracted in the range of θ_x^0 and θ_y^0.

the shorter than 10 meters are discarded to only keep lines rasterized from large plane segments. Each extracted line includes the line parameters and the votes of the line candidate: (ρ, θ, u).

In this section, the approach focus on determining the rotation matrix ${}^{H}_{P}\hat{\mathbf{R}}^0$ to align the point cloud. Therefore, only the angles θ of the extracted lines are analyzed. The extracted lines are depicted as several overlapping lines in Figure 4.4 each drawn with a line width corresponding to the cell size. The lines extracted from ${}^{P}\mathbf{M}$ with $\theta \in \theta_x^0$ are shown as yellow lines. The angles of these extracted lines θ_i are shown in Figure 4.5 as yellow points. The lines extracted from ${}^{P}\mathbf{M}$ with $\theta \in \theta_y^0$ are shown as cyan lines. The angles of these extracted lines θ_j are adjusted: $\tilde{\theta}_j = \theta_j - \frac{\pi}{2}$, and then shown in Figure 4.5 as cyan points. Note that the points in the red rectangle in Figure 4.5 correspond to the lines in the red rectangle in Figure 4.4, which was unexpected but needs to be avoid for determining the rotation matrix ${}^{H}_{P}\hat{\mathbf{R}}^0$.

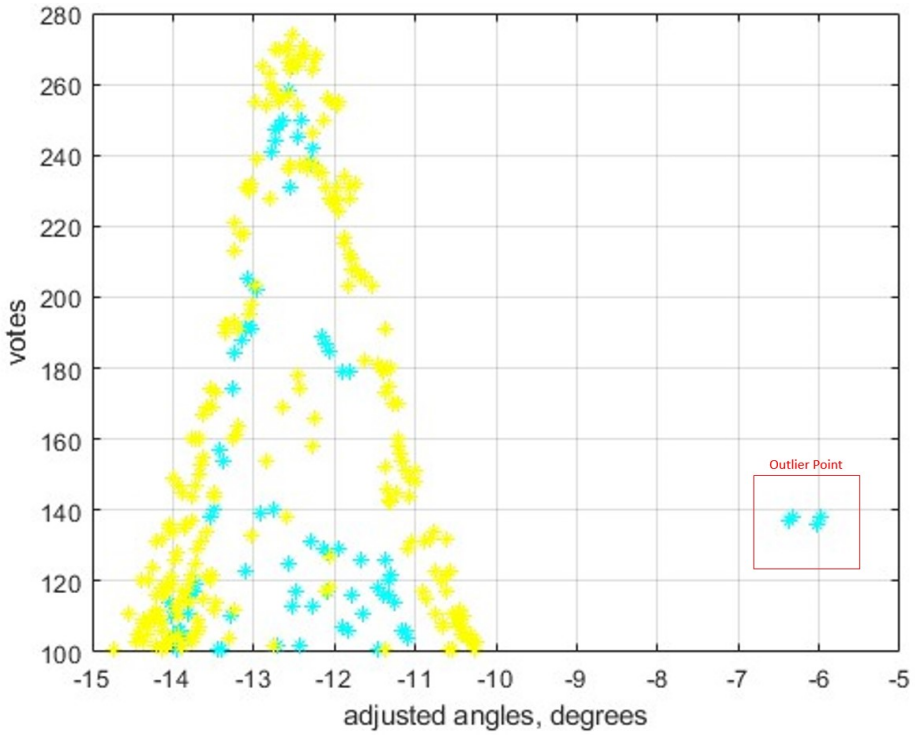

Figure 4.5: The votes of each candidate line angle θ_i and adjusted candidate line angle $\tilde{\theta}_j$ of the extracted lines in Figure 4.4.

According to Assumptions 2 and 3, the hatch is a rectangle and most of the vertical planes in the container are aligned with the hatch edge planes. Therefore, after being adjusted by 90 degrees, the angles of the lines extracted by the second HT should be roughly distributed in a smaller range from the angles of the lines extracted by the first HT, as most of the points in Figure 4.5 are distributed between -15 degrees and -10 degrees. A few vertical planes that are not aligned with any of the hatch edge planes will produce outliers that are not in this interval, e.g., the outliers shown in Figures 4.5 and 4.4). In Figure 4.5, the relationship between the angle of each extracted line corresponding to the number of votes during the HT is shown. Each point shown in Figure 4.5 has at least $\frac{10}{c} = \frac{10}{0.1} = 100$ votes. The more the line represented by each point in Figure 4.5 fits the data shown in Figure 4.3 (i.e., black pixels), the higher the number of votes it receives. If an

outlier line corresponds to a very long straight line in Figure 4.3, there is possibility that this line can have the highest votes during HT among all extracted lines. To avoid the effect of these outliers on determining the rotation matrix, the votes of lines are summed according to their corresponding angles: if the adjusted angles of several lines are the same, votes of the corresponding lines are summed up. The relationship of the summed votes corresponding to the adjusted angles is shown in Figure 4.6.

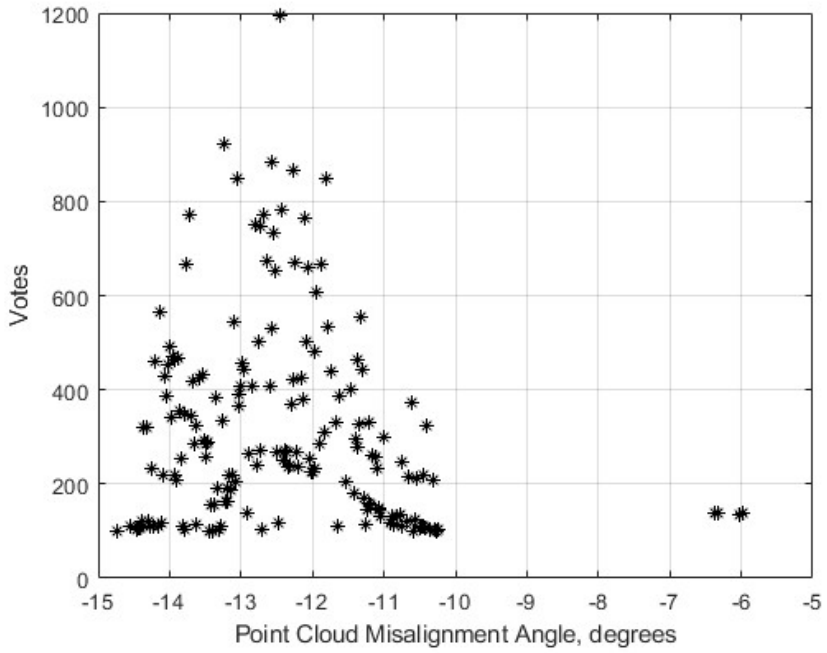

Figure 4.6: Votes of point cloud misalignment angle φ calculated from the cyan and yellow line extraction results.

The relationship between the cyan and yellow line angles θ and the misalignment angle φ was discussed in Section 3.6.2. The angle with most votes Θ^* are selected as the misalignment angle $\hat{\varphi}$: $\hat{\varphi} = \Theta^* = -12.46$ degrees = -0.217 rad. Therefore, the rasterized image ${}^{P}\mathbf{M}$ can be rotated by the angle $\hat{\varphi}$. The rotated image is shown in Figure 4.8.

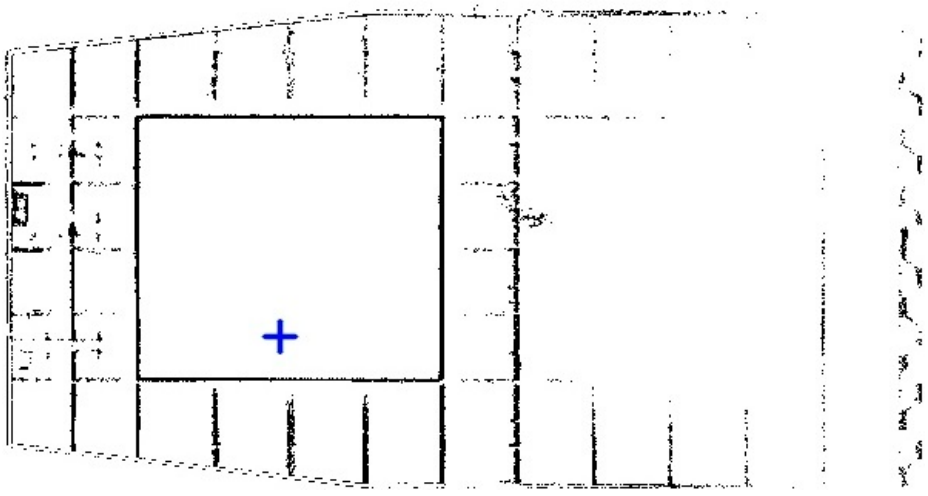

Figure 4.7: The image obtained by rotating the rasterization image ${}^{P}\mathbf{M}$ by the estimated misalignment angle $\hat{\varphi}$.

4.4 Hatch Edge Extraction Results

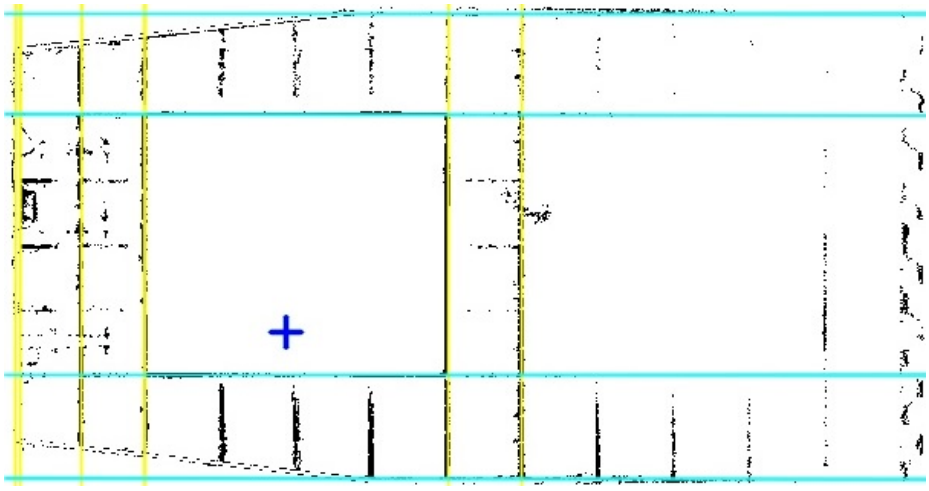

Figure 4.8: The HT line extraction results with $\delta_\theta = 0.1$ degree.

Figure 4.8 also shows the line extraction results after the rotation discussed in Section 3.7. The hatch edges are extracted as the two cyan lines closest to the blue cross on opposite sides in the y-direction and the two yellow lines closest to the blue cross on opposite sides in the x-direction.

The distance parameter of the four extracted edge lines are: $\rho_{i_0} = -2.6m$, $\rho_{i_1} = 9.1m$, $\rho_{i_2} = 14.1m$, and $\rho_{i_3} = -10.4m$.

4.5 Localization Initialization Results

The initial position is calculated with the hatch edge distances $|\rho_{i_0}| = 2.6m$, $|\rho_{i_3}| = 10.4m$ using eqn. (3.20): ${}^{N}\hat{\mathbf{T}}^{0}_{NP} = [2.6,\ 10.4,\ 0]^{\top}$. The alignment rotation matrix ${}^{H}_{P}\hat{\mathbf{R}}^{0}$ is calculated using eqn. (3.17):

$$ {}^{H}_{P}\hat{\mathbf{R}}^{0} = \begin{bmatrix} \cos(-12.46 \times \frac{\pi}{180}) & \sin(-12.46 \times \frac{\pi}{180}) & 0 \\ -\sin(-12.46 \times \frac{\pi}{180}) & \cos(-12.46 \times \frac{\pi}{180}) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.9764 & -0.2158 & 0 \\ 0.2158 & 0.9764 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.1} $$

Therefore, the initial orientation is calculated using eqn. (3.22):

$$ {}^{N}_{P}\hat{\mathbf{R}}^{0} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} {}^{H}_{P}\hat{\mathbf{R}}^{0} = \begin{bmatrix} 0.2158 & 0.9764 & 0 \\ 0.9764 & -0.2158 & 0 \\ 0 & 0 & -1.0000 \end{bmatrix}. \tag{4.2} $$

With the initial position and orientation, the N-frame map point cloud can be established with eqn. (3.24). The map point cloud is shown in Figure 4.9. The N-frame in Figure 4.9 is moved to avoid overlapping with the point cloud. The exact origin is below the corner and at the same height with the robot. The axis directions are the same as the axes shown in the Figure 4.9. Note that the color is still rendered using the z-value of each point and the color of the points changes from blue

to red, indicating the increase in z-value. Therefore, the points lower in the physical world is shown in warmer color in Figure 4.9.

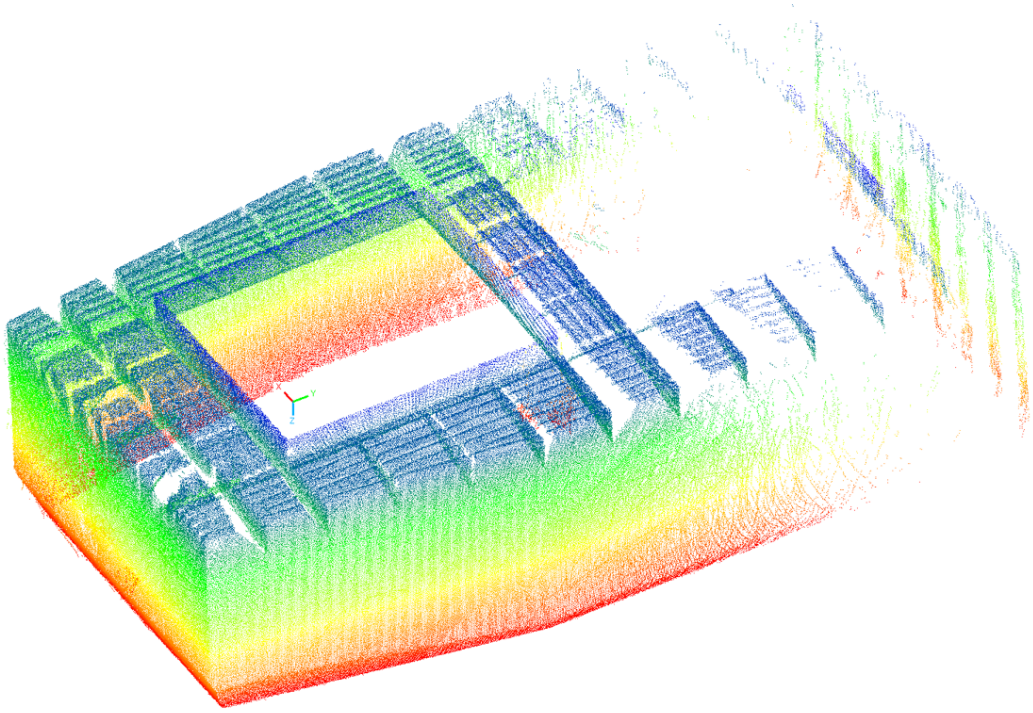

Figure 4.9: The navigation map point cloud ${}^{N}\mathbf{C}$. (The axes in the figure indicate only the direction, not the origin of N-frame.)

Chapter 5

Real-Time Localization Theory

In this chapter, several approaches of using estimated pose, velocity, angular rate and IMU measurements to calculate priors for reliable ICP registration are discussed and compared. The approach with the best performance on real-time localization is selected based on the experiment results.

The sections of this chapter are organized as follows: Section 5.1 introduces some related papers; Section 5.2 defines the problem discussed in this chapter; Section 5.3 discusses several approaches to determine the robot pose in real-time; Section 6.2 shows the evaluation of different IMU measurements because some types of the IMU measurements are not considered as reliable as the others in the experiments of this chapter; Section 6.3 shows and discusses the localization results of the various approaches introduced in Section 5.3.

5.1 Literature Review

LiDAR is widely used for localization because of its high ranging accuracy and capacity of working in some challenging scenario (e.g., night and underground environments) where GPS and camera are inappropriate. Trybala [56] evaluates an ICP-based approach only using LiDAR to localize the robot in a mine tunnel and build a point cloud map. Li et al. [27] design a method based on LiDAR and odometry to extract floors and walls and efficiently estimate the robot position for real-time localization. Zhang et al. [68] use IMU measurements to calculate the orientation and registering LiDAR scans to localize the LiDAR and build a point cloud map. Ye et al. [65] introduce a tightly-coupled approach using LiDAR and IMU sensors. The fusion of LiDAR and IMU sensors achieves better prior for point cloud matching and also results in a higher update rate of the localization. Zheng et al. [70] introduce a LiDAR-based localization approach efficiently storing point such that it can be used in large-scale scenario. The approaches above need to create the point cloud map simultaneously with localization, while the vehicle in our problem can scan the container to build the basemap before it moves. The pre-built map provides the target that the scans can align to for localizing the vehicle. Xu et al. [61] propose an approach with stereo camera for localization on a point cloud map. Pfrunder et al. [39] use LiDAR and IMU with a basemap for localization. For the problem of localizing within a basemap, the problem can be regarded as a point cloud alignment problem. As a widely used algorithm for point cloud registration, ICP [6] has many variants [18, 28, 30, 31, 47, 49, 60] and most of the paper introduces above use a kind of ICP algorithm for the point cloud matching. Men et al. [33] use an addition dimension calculated from the RGB values beside of the coordinates of each point. Chen [10], Low [31] use point-to-plane metric and Censi [8] uses point-to-line metric for distance calculation instead of the point-to-point

metric. Pomerleau et al [40, 41] present a review of ICP algorithms and compare different variants. In our problem, the LiDAR data includes the 3D coordinates and the normal of the basemap points are estimated after accumulation. Therefore, we will use an ICP algorithm implemented with the point-to-plane metric.

5.2 Problem Statement

To coordinate its work with the crane, the robot needs to provide its real-time position and orientation to the crane within a common reference frame. This common reference frame (i.e., N-frame) is defined in Section 3.8. The approach used by the crane to define this common reference frame is discussed in [24]. The focus of this chapter is determination of the robot position and orientation in N-frame during the transloading work. In this chapter, the robot position is represented as ${}^{N}\mathbf{P}^{k}$, where ${}^{N}\mathbf{P}^{k} = {}^{N}\mathbf{T}^{k}_{NP}$ as introduced in Section 2.3, and the robot orientation is represented with ${}^{N}_{P}\mathbf{R}^{k}$. The pose determination at t_0 for the initial position ${}^{N}\mathbf{P}^{0}$ (i.e.,${}^{N}\mathbf{T}^{0}_{NP}$) and orientation ${}^{N}_{P}\mathbf{R}^{0}$ of the robot are discussed in Section 3.8. Therefore, this chapter only discusses updating the position ${}^{N}\mathbf{P}^{k}$ and orientation ${}^{N}_{P}\mathbf{R}^{k}$ for $k > 0$.

During initialization, the LiDAR points are accumulated, voxelized, and transformed to create a point cloud of the environment (i.e., a basemap) ${}^{N}\mathbf{C}$ as defined in Section 3.8. A new LiDAR scan ${}^{P}\mathbf{S}^{k}$ is generated every $\frac{1}{f_l}$ second at t_k (i.e., the start time of the k-th epoch) as introduced in Section 2.1. The problem is to estimate the position ${}^{N}\mathbf{P}^{k}$ and orientation ${}^{N}_{P}\mathbf{R}^{k}$ at the t_k such that the points of ${}^{P}\mathbf{S}^{k}$ can be transformed using eqn. (2.5) and the transformed point cloud matches the basemap ${}^{N}\mathbf{C}$.

The IMU measurement frequency is $\frac{f_m}{f_l} = 20$ times the scan-rate of the LiDAR as introduced in Section 2.1. Therefore, the IMU measurements can be used to maintain the position ${}^{N}\mathbf{T}_{NP}^{k,i}$ and orientation ${}^{N}_{P}\mathbf{R}^{k,i}$ between two LiDAR scans (i.e., scans ${}^{P}\mathbf{S}^{k}$ and ${}^{P}\mathbf{S}^{k+1}$), where the superscript i is the index of the IMU measurements in the k-th epoch. The approaches to register the scan ${}^{P}\mathbf{S}^{k}$ to the basemap ${}^{N}\mathbf{C}$ using different measurements and prior knowledge will be discussed next in Section 5.3.

5.3 Pose Update

This section discusses the process to update the robot pose in real-time using LiDAR scans when the robot is moving. The new scan collected at time t_k in P-frame is denoted as ${}^{P}\mathbf{S}^{k}$. The scan ${}^{N}\mathbf{S}^{k}$ is the same scan as ${}^{P}\mathbf{S}^{k}$, after it is correctly transformed and expressed in N-frame. The goal of this section is to estimate ${}^{N}\mathbf{T}_{NP}^{k}$ and ${}^{N}_{P}\mathbf{R}^{k}$ for each $k > 0$ when the robot may have translated or rotated relative to its original pose such that ${}^{P}\mathbf{S}^{k}$ can be transformed into ${}^{N}\mathbf{S}^{k}$ using eqn. (2.5) and aligned to the basemap ${}^{N}\mathbf{C}$.

Section 5.3.1 discusses the feasibility of estimating the pose without using any prior information. Sections 5.3.2 to 5.3.4 use different priors to transform the scan ${}^{P}\mathbf{S}^{k}$ into an A-frame scan ${}^{A}\mathbf{S}^{k}$, which is roughly aligned to the basemap, using eqn. (2.6). As a prior of the pose update at t_k, Section 5.3.2 uses the estimated pose at t_{k-1}; Section 5.3.3 uses the estimated pose at t_{k-1} and the estimated velocity, which is calculated using the estimated pose at t_{k-1} and t_{k-2}; Section 5.3.4 uses the estimated pose at t_{k-1} and the angular velocity from IMU.

5.3.1 ICP: No prior

Without prior knowledge of the vehicle pose, the goal is to find the correct transformation to match the scan ${}^{P}\mathbf{S}^{k}$ to the basemap ${}^{N}\mathbf{C}$ directly:

$$[{}^{N}_{P}\hat{\mathbf{R}}^{k}, {}^{N}\hat{\mathbf{T}}^{k}_{NP}] = ICP({}^{N}\mathbf{C}, {}^{P}\mathbf{S}^{k}), \tag{5.1}$$

and get the rotation matrix ${}^{N}_{P}\hat{\mathbf{R}}^{k}$ and translation ${}^{N}\hat{\mathbf{T}}^{k}_{NP}$ to transform a point cloud from P-frame to N-frame at time t_k.

The process discussed above tries to register a scan collected with changed vehicle pose to the basemap that built at the initial robot pose. For our application, the map ${}^{N}\mathbf{C}$ can be dense enough by rotating the PT slowly to scan the environment. But the scan ${}^{P}\mathbf{S}^{k}$ is point cloud generated in one LiDAR scan, so that the point cloud is sparse and less informative than the map. In the case that the robot rotates 180 degrees after building the map, it is unreasonable to expect that the ICP algorithm can converge and get a transformation to rotate the scan by 180 degrees and match the map. More likely, the algorithm will converge to a local-minimum. Then the calculated transformation rotates the scan to a random place, which also minimizes the spatial distance between the scan and the map. But it is not the correct place for the scan in the N-frame.

5.3.2 P-ICP: Estimated Pose

This section introduces the steps of updating the pose at t_k with the estimated pose at time t_{k-1}. The problem of estimating ${}^{N}\mathbf{P}^{k}$ and ${}^{P}_{N}\mathbf{R}^{k}$ can be reorganized to a problem of estimating the difference between ${}^{N}\mathbf{P}^{k}$ and ${}^{N}\mathbf{P}^{k-1}$, and the difference between ${}^{P}_{N}\mathbf{R}^{k}$ and ${}^{P}_{N}\mathbf{R}^{k-1}$. The robot position at t_{k-1} and t_k are ${}^{N}\mathbf{P}^{k-1} = {}^{N}\mathbf{T}^{k-1}_{NP}$ and ${}^{N}\mathbf{P}^{k} = {}^{N}\mathbf{T}^{k}_{NP}$, respectively. The difference of these two position

is defined as ${}^{N}\mathbf{P}^{k}_{k-1} = {}^{N}\mathbf{P}^{k} - {}^{N}\mathbf{P}^{k-1}$. Therefore, the position at t_k can be represented as:

$$
\begin{aligned}
{}^{N}\mathbf{T}^{k}_{NP} &= {}^{N}\mathbf{P}^{k} && (5.2)\\
&= {}^{N}\mathbf{P}^{k}_{k-1} + {}^{N}\mathbf{P}^{k-1}. && (5.3)
\end{aligned}
$$

The rotation matrix ${}^{P}_{N}\mathbf{R}^{k}$ represents the rotation from the N-frame at t_k to the P-frame at t_k. The rotation matrix ${}^{P}_{N}\mathbf{R}^{k-1}$ represents the rotation from the N-frame at t_{k-1} to the P-frame at t_{k-1}. The rotation matrix ${}^{P}\mathbf{R}^{k}_{k-1}$ represents the rotation from the P-frame at t_{k-1} to the P-frame at t_k. Therefore, the rotation from N-frame at t_k to P-frame at t_{k-1} can be expressed as:

$$
{}^{P}_{N}\mathbf{R}^{k} = {}^{P}\mathbf{R}^{k}_{k-1}\ {}^{P}_{N}\mathbf{R}^{k-1}. \tag{5.4}
$$

Because the transpose of a rotation matrix coincides its inverse: $({}^{P}_{N}\mathbf{R}^{k-1})^{-1} = {}^{N}_{P}\mathbf{R}^{k-1}$, transposing both sides of eqn. (5.4) yields:

$$
\begin{aligned}
{}^{N}_{P}\mathbf{R}^{k} &= ({}^{P}_{N}\mathbf{R}^{k-1})^{\top}\ ({}^{P}\mathbf{R}^{k}_{k-1})^{\top}\\
&= {}^{N}_{P}\mathbf{R}^{k-1}\ {}^{P}\mathbf{R}^{k-1}_{k}.
\end{aligned} \tag{5.5}
$$

Substituting eqns. (5.3) and (5.5) into eqn. (2.5) yields:

$$
\begin{aligned}
{}^{N}\mathbf{p}_i &= {}^{N}_{P}\mathbf{R}^{k}\ {}^{P}\mathbf{p}_i + {}^{N}\mathbf{T}^{k}_{NP}\\
&= {}^{N}_{P}\mathbf{R}^{k-1}\ {}^{P}\mathbf{R}^{k-1}_{k}\ {}^{P}\mathbf{p}_i + {}^{N}\mathbf{P}^{k}_{k-1} + {}^{N}\mathbf{T}^{k-1}_{NP}.
\end{aligned} \tag{5.6}
$$

where ${}^{N}_{P}\mathbf{R}^{k-1}$ and ${}^{N}\mathbf{T}^{k-1}_{NP}$ are determined at the epoch of time t_{k-1}, and the point ${}^{P}\mathbf{p}_i$ refer to each point of the scan ${}^{P}\mathbf{S}^{k}$.

To reduce the misalignment between the scan ${}^{P}\mathbf{S}^{k}$ and the basemap ${}^{N}\mathbf{C}$ and find better correspondence between them for running ICP algorithm, A-frame is defined as a temporary frame at t_k for the roughly alignment as introduced in Section 2.2. The transformation from P-frame to A-frame is defined in eqn. (2.6). In this section, the estimated position ${}^{N}\hat{\mathbf{P}}^{k-1}$ and orientation ${}^{P}_{N}\hat{\mathbf{R}}^{k-1}$

at t_{k-1} are used as the prior knowledge: ${}^{N}\bar{\mathbf{P}}^{k} = {}^{N}\hat{\mathbf{P}}^{k-1}$ and ${}^{P}_{N}\bar{\mathbf{R}}^{k} = {}^{P}_{N}\hat{\mathbf{R}}^{k-1}$. Therefore, the scan ${}^{P}\mathbf{S}^{k}$ is roughly aligned to ${}^{N}\mathbf{C}$ as:

$$ {}^{A}\mathbf{p}_i = {}^{N}_{P}\hat{\mathbf{R}}^{k-1}\ {}^{P}\mathbf{p}_i + {}^{N}\hat{\mathbf{P}}^{k-1}, \tag{5.7} $$

where A-frame is defined as a temporary frame at t_k for the rough alignment as introduced in Section 2.2. After the rough alignment, the scan ${}^{P}\mathbf{S}^{k}$ is transformed into ${}^{A}\mathbf{S}^{k}$. The ICP algorithm using the prior estimated position ${}^{N}\hat{\mathbf{P}}^{k-1}$ and orientation ${}^{P}_{N}\hat{\mathbf{R}}^{k}$ to register ${}^{P}\mathbf{S}^{k}$ into ${}^{N}\mathbf{C}$ is expressed as:

$$ [\mathbf{R}_r,\ \mathbf{T}_r] = \underset{{}^{N}_{P}\hat{\mathbf{R}}^{k-1},{}^{N}\hat{\mathbf{P}}^{k-1}}{ICP}({}^{N}\mathbf{C}, {}^{A}\mathbf{S}^{k}), \tag{5.8} $$

which is used to calculate a rotation matrix $\mathbf{R}_r$ and a translation vector $\mathbf{T}_r$ such that the scan ${}^{A}\mathbf{S}^{k}$ can be transformed with $\mathbf{R}_r$ and $\mathbf{T}_r$ to match the basemap ${}^{N}\mathbf{C}$:

$$ {}^{N}\mathbf{p}_i = \mathbf{R}_r\ {}^{A}\mathbf{p}_i + \mathbf{T}_r, \tag{5.9} $$

$$ = \mathbf{R}_r\ ({}^{N}_{P}\bar{\mathbf{R}}^{k}\ {}^{P}\mathbf{p}_i + {}^{N}\bar{\mathbf{T}}^{k}_{NP}) + \mathbf{T}_r, \tag{5.10} $$

$$ = \mathbf{R}_r\ {}^{N}_{P}\bar{\mathbf{R}}^{k}\ {}^{P}\mathbf{p}_i + \mathbf{R}_r\ {}^{N}\bar{\mathbf{T}}^{k}_{NP} + \mathbf{T}_r. \tag{5.11} $$

Comparing eqn. (5.11) with eqn. (2.5) yields:

$$ {}^{N}\mathbf{P}^{k} = \mathbf{R}_r\ {}^{N}\bar{\mathbf{T}}^{k}_{NP} + \mathbf{T}_r, \tag{5.12} $$

$$ = \mathbf{R}_r\ {}^{N}\bar{\mathbf{P}}^{k} + \mathbf{T}_r, \tag{5.13} $$

and

$$ {}^{N}_{P}\mathbf{R}^{k} = \mathbf{R}_r\ {}^{N}_{P}\bar{\mathbf{R}}^{k}, \tag{5.14} $$

therefore, in this section, the position at t_k is calculated as:

$$ {}^{N}\mathbf{P}^{k} = \mathbf{R}_r\ {}^{N}\mathbf{P}^{k-1} + \mathbf{T}_r. \tag{5.15} $$

and the orientation at t_k is calculated as:

$$ {}^{N}_{P}\mathbf{R}^{k} = \mathbf{R}_r \, {}^{N}_{P}\mathbf{R}^{k-1}. \tag{5.16} $$

5.3.3 PV-ICP: Estimated Pose, Velocity, and Angular Rate

Building on the discussion in Section 5.3.2 of using the robot pose at t_{k-1} as a prior of the robot pose at t_k, this section further discusses the use of the robot pose at t_{k-1}, and the robot velocity estimated using the robot pose at t_{k-1} and t_{k-2} to calculate a prior of the robot pose at t_k.

5.3.3.1 Orientation Prior

The orientations estimated at time t_{k-2} and t_{k-1} are ${}^{P}_{N}\hat{\mathbf{R}}^{k-2}$ and ${}^{P}_{N}\hat{\mathbf{R}}^{k-1}$, which are used for calculating the orientation difference next. The orientation difference between t_{k-1} and t_{k-2} is defined as ${}^{P}\mathbf{R}^{k-1}_{k}$. Taking the orientation at t_{k-1} to the left side of eqn. (5.5) and then swap the two sides yields:

$$ {}^{P}\mathbf{R}^{k-1}_{k} = ({}^{N}_{P}\mathbf{R}^{k-1})^{-1} \, {}^{N}_{P}\mathbf{R}^{k}. \tag{5.17} $$

For the orientation difference between t_{k-1} and t_{k-2}, $k-1$ is used instead of k in eqn. (5.17) to make:

$$ {}^{P}\mathbf{R}^{k-2}_{k-1} = ({}^{N}_{P}\mathbf{R}^{k-2})^{-1} \, {}^{N}_{P}\mathbf{R}^{k-1}, \tag{5.18} $$

therefore, the orientation difference between t_{k-2} and t_{k-1} can be calculated with ${}^{P}_{N}\hat{\mathbf{R}}^{k-1}$ and ${}^{P}_{N}\hat{\mathbf{R}}^{k-2}$ as:

$$ {}^{P}\mathbf{R}^{k-2}_{k-1} = ({}^{N}_{P}\hat{\mathbf{R}}^{k-2})^{-1} \, {}^{N}_{P}\hat{\mathbf{R}}^{k-1}, \tag{5.19} $$

which should be a rotation matrix of a small angle. This rotation matrix is used to calculate the rotation velocity matrix next. The eqn. (2.52) in [15] defines the relation between a small angle

rotation matrix and its corresponding rotation velocity matrix:

$$\mathbf{R}_{b(t)}^{b(t+\delta t)} = \mathbf{I} - \mathbf{\Omega}_{ab}^{b}\delta t. \tag{5.20}$$

To first order, the inverse of eqn. (5.20) is:

$$\mathbf{R}_{b(t+\delta t)}^{b(t)} = \mathbf{I} + \mathbf{\Omega}_{ab}^{b}\delta t, \tag{5.21}$$

where the '$b(t)$', '$b(t+\delta t)$', 'a' and 'δt' correspond to the P-frame at t_{k-2}, the P-frame at t_{k-1}, N-frame and the time difference $\delta t_{k-1} = t_{k-1} - t_{k-2}$. Therefore, the small angle rotation in this section can be expressed as:

$${}^{P}\mathbf{R}_{k-1}^{k-2} = \mathbf{I} + {}^{P}\mathbf{\Omega}_{NP}^{k-1}\delta t_{k-1}, \tag{5.22}$$

where the superscripts of $\mathbf{\Omega}$ are adjusted following the notation of this dissertation to represent the matrix-form rotation velocity of P-frame with respect to N-frame represented in P-frame at t_{k-1}. Substituting eqn. (5.22) into eqn. (5.19), the equation can be manipulated as:

$$\mathbf{I} + {}^{P}\mathbf{\Omega}_{NP}^{k-1}\delta t_{k-1} = ({}_{P}^{N}\mathbf{R}^{k-2})^{-1}\,{}_{P}^{N}\mathbf{R}^{k-1}, \tag{5.23}$$

$${}^{P}\mathbf{\Omega}_{NP}^{k-1}\delta t_{k-1} = ({}_{P}^{N}\mathbf{R}^{k-2})^{-1}\,{}_{P}^{N}\mathbf{R}^{k-1} - \mathbf{I}, \tag{5.24}$$

$${}^{P}\mathbf{\Omega}_{NP}^{k-1} = \frac{({}_{P}^{N}\mathbf{R}^{k-2})^{-1}\,{}_{P}^{N}\mathbf{R}^{k-1} - \mathbf{I}}{\delta t_{k-1}}. \tag{5.25}$$

The rotation velocity at t_{k-1} can be approximately calculated with ${}_{N}^{P}\hat{\mathbf{R}}^{k-2}$ and ${}_{N}^{P}\hat{\mathbf{R}}^{k-1}$ using eqn. (5.25) as:

$${}^{P}\hat{\mathbf{\Omega}}_{NP}^{k-1} = \frac{({}_{P}^{N}\hat{\mathbf{R}}^{k-2})^{-1}\,{}_{P}^{N}\hat{\mathbf{R}}^{k-1} - \mathbf{I}}{\delta t_{k-1}}. \tag{5.26}$$

The orientation difference between t_k and t_{k-1} is approximately calculated with eqn. (5.22) as:

$${}^{P}\hat{\mathbf{R}}_{k}^{k-1} = \mathbf{I} + {}^{P}\hat{\mathbf{\Omega}}_{NP}^{k-1}\delta t_{k}, \tag{5.27}$$

therefore, the prior of the robot orientation in eqn. (2.6) is then calculated using eqn. (5.5) as:

$$ {}^{N}_{P}\bar{\mathbf{R}}^{k} = {}^{N}_{P}\mathbf{R}^{k-1}\ {}^{P}\hat{\mathbf{R}}^{k-1}_{k} \tag{5.28} $$

$$ = {}^{N}_{P}\mathbf{R}^{k-1}\ (\mathbf{I} + {}^{P}\hat{\boldsymbol{\Omega}}^{k-1}_{NP}\,\delta t_k). \tag{5.29} $$

5.3.3.2 Position Prior

The positions estimated at time t_{k-2} and t_{k-1} are ${}^{P}_{N}\hat{\mathbf{R}}^{k-2}$ and ${}^{P}_{N}\hat{\mathbf{R}}^{k-1}$, which are used for calculating the approximate translation velocity of the robot:

$$ {}^{N}\hat{\mathbf{V}}_{k-1} = \frac{{}^{N}\hat{\mathbf{P}}^{k-1} \_ {}^{N}\hat{\mathbf{P}}^{k-2}}{t_{k-1} - t_{k-2}}. \tag{5.30} $$

The translation vector from t_{k-1} to t_k is approximately calculated as ${}^{N}\hat{\mathbf{T}}^{k}_{k-1} = {}^{N}\hat{\mathbf{V}}_{k-1}\delta t_k$, therefore, The prior position in eqn. (2.6) is calculated as:

$$ {}^{N}\bar{\mathbf{T}}^{k}_{NP} = {}^{N}\bar{\mathbf{P}}^{k} \tag{5.31} $$

$$ = {}^{N}\hat{\mathbf{T}}^{k}_{k-1} + {}^{N}\mathbf{P}^{k-1} \tag{5.32} $$

$$ = {}^{N}\hat{\mathbf{V}}_{k-1}\delta t_k + {}^{N}\mathbf{P}^{k-1}. \tag{5.33} $$

5.3.3.3 Registration and Pose Update

With the prior position ${}^{N}\bar{\mathbf{T}}^{k}_{NP}$ calculated by eqn. (5.33) and orientation ${}^{N}_{P}\bar{\mathbf{R}}^{k}$ calculated by eqn. (5.29), the scan ${}^{P}\mathbf{S}^{k}$ is transformed into A-frame using eqn. (2.6). The transformed scan ${}^{A}\mathbf{S}^{k}$ is registered to ${}^{N}\mathbf{C}$ using ICP algorithm, which is expressed as:

$$ [\mathbf{R}_r,\ \mathbf{T}_r] = \underset{{}^{N}_{P}\hat{\mathbf{R}}^{k-1}, {}^{N}\hat{\mathbf{P}}^{k-1}, {}^{P}\hat{\boldsymbol{\Omega}}^{k-1}_{NP}, {}^{N}\hat{\mathbf{V}}_{k-1}}{ICP}({}^{N}\mathbf{C}, {}^{A}\mathbf{S}^{k}), \tag{5.34} $$

where the registration results of $\mathbf{R}_r$ and $\mathbf{T}_r$ can be used transform ${}^{A}\mathbf{S}^k$ to match ${}^{N}\mathbf{C}$. Then, the position at t_k is calculated with eqn. (5.12) as:

$$
\begin{aligned}
{}^{N}\mathbf{P}^k &= \mathbf{R}_r \, {}^{N}\bar{\mathbf{T}}^k_{NP} + \mathbf{T}_r, && (5.35)\\
&= \mathbf{R}_r \, ({}^{N}\hat{\mathbf{V}}_{k-1}\delta t_k + {}^{N}\mathbf{P}^{k-1}) + \mathbf{T}_r. && (5.36)
\end{aligned}
$$

The orientation matrix at t_k is calculated with eqn. (5.14) as:

$$
\begin{aligned}
{}^{N}_{P}\mathbf{R}^k &= \mathbf{R}_r \, {}^{N}_{P}\bar{\mathbf{R}}^k, && (5.37)\\
&= \mathbf{R}_r \, {}^{N}_{P}\mathbf{R}^{k-1} \, (\mathbf{I} + {}^{P}\hat{\mathbf{\Omega}}^{k-1}_{NP}\delta t_k). && (5.38)
\end{aligned}
$$

5.3.4 PI-ICP: Estimated Pose, Velocity and IMU Measurements

Sections 5.3.1 to 5.3.3 have discussed only using the LiDAR measurements to update the robot pose. This section further discusses using the IMU acceleration and angular rate measurements to aid the process of updating the robot pose. In this section, the orientation difference ${}^{P}\mathbf{R}^{k-1}_{k}$ between LiDAR scan times t_{k-1} and t_k is calculated with the IMU measurements of angular rate, instead of predicting ${}^{P}\mathbf{R}^{k-1}_{k}$ using the angular rate matrix ${}^{P}\mathbf{\Omega}^{k-1}_{NP}$ estimated purely from LiDAR data as discussed in Section 5.3.3. The estimated velocity ${}^{N}\hat{\mathbf{V}}_{k-1}$ is updated with each acceleration measurement and used to calculate the translation between t_{k-1} and t_k.

5.3.4.1 Orientation Prior

As a reference of using IMU measurements for calculating the IMU rotation, the orientation update using IMU angular rate measurements $\boldsymbol{w}^a_{ba}$ from t_{k-1} to t_k is defined by eqn. (2.70) in [15] as:

$$
\mathbf{R}^a_b(t_k) = (\mathbf{I} - \frac{\sin(\|\boldsymbol{\nu}\|)}{\|\boldsymbol{\nu}\|}\boldsymbol{\Upsilon} + \frac{1-\cos(\|\boldsymbol{\nu}\|)}{\|\boldsymbol{\nu}\|^2}\boldsymbol{\Upsilon}^2)\,\mathbf{R}^a_b(t_{k-1}), \tag{5.39}
$$

where $\mathbf{R}_b^a(t_k)$ and $\mathbf{R}_b^a(t_{k-1})$ are the orientation matrix at t_k and t_{k-1}, respectively. The vector $\boldsymbol{\nu} = [\nu_x,\ \nu_y,\ \nu_z]^\top$ is calculated as the integral of the angular rate from t_{k-1} to t_k: $\boldsymbol{\nu} = \int_{t_{k-1}}^{t_k} \boldsymbol{w}_{ba}^a(\tau) d\tau \in \mathscr{R}^3$, and $\boldsymbol{w}_{ba}^a \in \mathscr{R}^3$ is the angular rate of 'a'-frame relative to 'b'-frame represented in 'a'-frame. The symbol $\boldsymbol{\Upsilon}$ is calculated as:

$$\boldsymbol{\Upsilon} = \begin{bmatrix} 0 & -\nu_z & \nu_y \\ \nu_z & 0 & -\nu_x \\ -\nu_y & \nu_x & 0 \end{bmatrix}. \tag{5.40}$$

In this dissertation, P-frame rotates relative to N-frame and the rotation during the time interval (t_{j-1}, t_j) is expressed as:

$$\mathbf{R}_N^P(t_j) = (\mathbf{I} - \frac{\sin(\|\boldsymbol{\nu}_j\|)}{\|\boldsymbol{\nu}_j\|}\boldsymbol{\Upsilon} + \frac{1-\cos(\|\boldsymbol{\nu}_j\|)}{\|\boldsymbol{\nu}_j\|^2}\boldsymbol{\Upsilon}^2)\ \mathbf{R}_N^P(t_{j-1}), \tag{5.41}$$

where 'j' is used to indicate the IMU measurement index between the LiDAR scans at t_{k-1} and t_k. The time between two IMU measurements is $\Delta t = \frac{1}{f_m}$. The variable $\boldsymbol{\nu}_j$ is calculated with the IMU angular rate of the j-th measurement ${}^P\boldsymbol{w}_{NP}^j$ as: $\boldsymbol{\nu}_j = {}^P\boldsymbol{w}_{NP}^j \Delta t$. The range of j is $j = 0, 1, 2, \ldots, \frac{t_k - t_{k-1}}{\Delta t}$. The orientation difference ${}^P\mathbf{R}_{j-1}^j$ during the time interval (t_{j-1}, t_j) is represented as:

$${}^P\mathbf{R}_{j-1}^j = (\mathbf{I} - \frac{\sin(\|\boldsymbol{\nu}_j\|)}{\|\boldsymbol{\nu}_j\|}\boldsymbol{\Upsilon}_j + \frac{1-\cos(\|\boldsymbol{\nu}_j\|)}{\|\boldsymbol{\nu}_j\|^2}\boldsymbol{\Upsilon}_j^2). \tag{5.42}$$

Then, $\boldsymbol{\Upsilon}_j$ is calculated with $\boldsymbol{\nu}_j$ using eqn. (5.40) and the rotation of P-frame between t_j and t_{j-1} is calculated using eqn. (5.42). The robot orientation difference from t_{k-1} to t_k is calculated as the product of the rotation calculated with each IMU measurement in the period:

$${}^P\mathbf{R}_{k-1}^k = {}^P\mathbf{R}_{j_{n-1}}^k \ \cdots\ {}^P\mathbf{R}_{j_1}^{j_2}\ {}^P\mathbf{R}_{k-1}^{j_1} \tag{5.43}$$

$$= {}^P\mathbf{R}_{j_{n-1}}^{j_n} \ \cdots\ {}^P\mathbf{R}_{j_1}^{j_2}\ {}^P\mathbf{R}_{j_0}^{j_1}, \tag{5.44}$$

where $t_{j_0} = t_{k-1}$ and $t_{j_n} = t_k$. With the N measurements at t_k, the prior of robot orientation is calculated as:

$$ {}^{N}_{P}\bar{\mathbf{R}}^{k} = ({}^{P}_{N}\bar{\mathbf{R}}^{k})^{\top}, \tag{5.45} $$

where $({}^{P}_{N}\bar{\mathbf{R}}^{k})^{\top}$ is calculated as:

$$ {}^{P}_{N}\bar{\mathbf{R}}^{k} = {}^{P}\mathbf{R}^{k}_{k-1}\ {}^{P}_{N}\mathbf{R}^{k-1} \tag{5.46} $$

$$ = {}^{P}\mathbf{R}^{j_n}_{j_{n-1}}\ \cdots\ {}^{P}\mathbf{R}^{j_2}_{j_1}\ {}^{P}\mathbf{R}^{j_1}_{j_0}\ {}^{P}_{N}\mathbf{R}^{k-1} \tag{5.47} $$

5.3.4.2 Position Prior

After estimating the velocity ${}^{N}\hat{\mathbf{V}}_{k-1}$ with the last LiDAR scan at t_{k-1}, the velocity vector can be expressed in P-frame as:

$$ {}^{P}\hat{\mathbf{V}}_{k-1} = {}^{P}_{N}\mathbf{R}^{k-1}\ {}^{N}\hat{\mathbf{V}}_{k-1}, \tag{5.48} $$

where ${}^{P}_{N}\mathbf{R}^{k-1}$ is the transpose of the robot orientation matrix ${}^{N}_{P}\mathbf{R}^{k-1}$ estimated with the LiDAR scan at t_{k-1}. When a new IMU measurement arrives at t_j, the robot velocity can be updated with the IMU measurement as:

$$ {}^{P}\hat{\mathbf{V}}_{j} = {}^{P}\hat{\mathbf{V}}_{j-1} + {}^{P}\boldsymbol{a}_{j}\,\Delta t, \tag{5.49} $$

where ${}^{P}\boldsymbol{a}_{j} \in \mathscr{R}^3$ is the acceleration measurement vector at t_j after deducting the portions of gravity from it. For any period between two IMU measurement, the velocity is assumed constant. Therefore, the translation is calculated as:

$$ {}^{P}\mathbf{T}^{j}_{j-1} = {}^{P}\hat{\mathbf{V}}_{j-1}\,\Delta t, \tag{5.50} $$

The transpose of the robot orientation matrix ${}^{N}_{P}\mathbf{R}^{j}$ is calculated referring to eqn. (5.44). Therefore, the translation vector ${}^{P}\mathbf{P}^{j}_{j-1}$ can be transformed and expressed in N-frame as:

$$
\begin{aligned}
{}^{N}\mathbf{T}^{j}_{j-1} &= {}^{N}_{P}\mathbf{R}^{j}\ {}^{P}\mathbf{T}^{j}_{j-1} && (5.51)\\
&= ({}^{P}_{N}\mathbf{R}^{j})^{\top}\ {}^{P}\mathbf{T}^{j}_{j-1} && (5.52)\\
&= ({}^{P}\mathbf{R}^{j}_{j-1}\ \cdots\ {}^{P}\mathbf{R}^{j_1}_{j_0}\ {}^{P}_{N}\mathbf{R}^{k-1})^{\top}\ {}^{P}\mathbf{T}^{j}_{j-1} && (5.53)
\end{aligned}
$$

The robot position at t_k is calculated as the summation of the robot position at t_{k-1} and the translation vector calculated with each IMU measurement from t_{k-1} to t_k. Therefore, the displacement is calculated as:

$$
{}^{N}\mathbf{T}^{k}_{k-1} = {}^{N}\mathbf{T}^{j_1}_{j_0} + {}^{N}\mathbf{T}^{j_2}_{j_1} + \cdots + {}^{N}\mathbf{T}^{j_n}_{j_{n-1}}, \tag{5.54}
$$

and the prior for robot position at t_k is calculated as:

$$
{}^{N}\bar{\mathbf{P}}^{k} = {}^{N}\mathbf{P}^{k-1} + {}^{N}\mathbf{T}^{k}_{k-1}. \tag{5.55}
$$

5.3.4.3 Registration and Pose Update

The priors ${}^{N}\bar{\mathbf{P}}^{k}$ and ${}^{N}_{P}\bar{\mathbf{R}}^{k}$ are used to transform points in ${}^{P}\mathbf{S}^{k}$ using eqn. (2.6) and the transformed scan ${}^{A}\mathbf{S}^{k}$ is registered into ${}^{N}\mathbf{C}$ using ICP algorithm to obtain the registration parameters $\mathbf{R}_r$ and $\mathbf{T}_r$. In this section, the ICP algorithm is expressed as:

$$
[\mathbf{R}_r,\ \mathbf{T}_r] = \underset{{}^{N}_{P}\hat{\mathbf{R}}^{k-1}, {}^{N}_{P}\hat{\mathbf{R}}^{k-2}, {}^{N}\hat{\mathbf{P}}^{k-1}, {}^{N}\hat{\mathbf{P}}^{k-2}, \boldsymbol{a}, \boldsymbol{w}}{ICP}({}^{N}\mathbf{C}, {}^{A}\mathbf{S}^{k}). \tag{5.56}
$$

At last, the position at t_k is calculated with eqn. (5.12) as:

$$
\begin{aligned}
{}^{N}\mathbf{P}^{k} &= \mathbf{R}_r\ {}^{N}\bar{\mathbf{T}}^{k}_{NP} + \mathbf{T}_r && (5.57)\\
&= \mathbf{R}_r\ ({}^{N}\mathbf{P}^{k-1} + {}^{P}\mathbf{T}^{j_1}_{j_0} + {}^{N}\mathbf{T}^{j_2}_{j_1} + \cdots + {}^{P}\mathbf{T}^{j_n}_{j_{n-1}}) + \mathbf{T}_r. && (5.58)
\end{aligned}
$$

The orientation matrix at t_k is calculated with eqn. (5.14) as:

$$
\begin{aligned}
{}^{N}_{P}\mathbf{R}^{k} &= \mathbf{R}_r \, {}^{N}_{P}\bar{\mathbf{R}}^{k} && (5.59)\\
&= \mathbf{R}_r \left({}^{P}\mathbf{R}^{k}_{j_N} \, {}^{P}\mathbf{R}^{j_N}_{j_N-1} \cdots {}^{P}\mathbf{R}^{j_2}_{j_1} \, {}^{P}\mathbf{R}^{j_1}_{k-1} \left({}^{N}_{P}\mathbf{R}^{k-1} \right)^{\top} \right)^{\top}. && (5.60)
\end{aligned}
$$

Chapter 6

Real-Time Localization Experiments

6.1 Experiment Design

This section discusses some experiments on real-time localization, which consists of two parts. Section 6.1.1 introduces IMU experiments, where we will collect and analyze IMU data from the same model of IMU in the stationary and actual working environments. These experiments focus on the effects of the environments to the IMU measurements. Section 6.1.2 introduces experiments to compare the performance of each approach on calculating the prior pose. The experiments focus on the real-time performance and the accuracy of the prior pose calculated with each approach.

6.1.1 IMU experiments

In Section 5.3.4, the IMU acceleration measurements ${}^{P}\boldsymbol{a}$ and angular rate measurements ${}^{P}\boldsymbol{w}_{NP}$ are used for calculating the pose prior. The notation of ${}^{P}\boldsymbol{a}$ was simplified for easier expression. With subscripts indicating the frames, the acceleration should be denoted as ${}^{P}\boldsymbol{a}_{NP}$, which represents the relative acceleration of P-frame relative to N-frame represented in P-frame. How-

ever, the raw IMU measurements represent the acceleration and angular rate of the IMU body (i.e., P-frame) relative to the inertial frame (i.e., g-frame). The raw IMU measurements obtained is ${}^{P}\boldsymbol{a}_{gP}$ and ${}^{P}\boldsymbol{w}_{gP}$, which represent the relative acceleration and angular rate of the P-frame referring to the inertial frame represented in P-frame, respectively. To obtain ${}^{P}\boldsymbol{a}_{NP}$ and ${}^{P}\boldsymbol{w}_{NP}$, which are used in Section 5.3.4, the measurements of N-frame referring to g-frame is needed to calculate:

$$ {}^{P}\boldsymbol{a}_{NP} = {}^{P}\boldsymbol{a}_{gP} - {}^{P}\boldsymbol{a}_{gN}, \tag{6.1} $$

and

$$ {}^{P}\boldsymbol{w}_{NP} = {}^{P}\boldsymbol{w}_{gP} - {}^{P}\boldsymbol{w}_{gN}. \tag{6.2} $$

These measurements of N-frame requires an IMU rigidly installed with N-frame, which is physically the hatch corner as discussed in Chapter 3. However, modifications on any parts of the container are prohibited in this application. Therefore, the measurements ${}^{P}\boldsymbol{a}_{gN}$ and ${}^{P}\boldsymbol{w}_{gN}$ are unavailable, and they will be assumed as zero (i.e., ${}^{P}\boldsymbol{a}_{gN} = \boldsymbol{0}$ and ${}^{P}\boldsymbol{w}_{gN} = \boldsymbol{0}$) in the application. Several assessments are applied to evaluate the effects and will be discussed next.

Two sets of IMU data are collected for assessments. The measurements collected in the stationary environment is marked with a left-subscript 'r': ${}^{P}_{r}\boldsymbol{w}_{gP}$, ${}^{P}_{r}\boldsymbol{a}_{gP}$, and called the *reference data*. The mean value of the reference data are calculated and denoted as ${}^{P}_{r}\bar{\boldsymbol{w}}_{gP}$ and ${}^{P}_{r}\bar{\boldsymbol{a}}_{gP}$. The standard-deviation (STD) of the reference data are calculated and denoted as ${}_{r}\boldsymbol{\sigma}_{w}$ and ${}_{r}\boldsymbol{\sigma}_{a}$. Similarly, the measurements collected in the non-stationary environment is marked with a left-subscript 'e': ${}^{P}_{e}\boldsymbol{w}_{gP}$, ${}^{P}_{e}\boldsymbol{a}_{gP}$, and called the *experiment data*. The mean and STD of the experiment data are calculated and denoted as: ${}^{P}_{e}\bar{\boldsymbol{w}}_{gP}$, ${}^{P}_{e}\bar{\boldsymbol{a}}_{gP}$, ${}_{e}\boldsymbol{\sigma}_{w}$ and ${}_{e}\boldsymbol{\sigma}_{a}$.

First, when the robot is initially placed in the container, it does not move. The IMU data collected during this initial stationary interval is used to compute ${}_{e}\boldsymbol{\sigma}_{w}$ and ${}_{e}\boldsymbol{\sigma}_{a}$. These are compared

with the ${}_r\sigma_w$ and ${}_r\sigma_a$ to analyze the effects of the engine vibration relative to the values in the non-stationary container. Theoretically, the STD of the experiment data with engine vibration should be larger than the reference data STD. The experimental results will be discussed in Section 6.2.1.

Second, when the robot is known to be not moving in the container while the engine is on, the IMU measurements can be integrated to evaluate the rotation and displacement of integrating IMU measurements. The orientation difference can be calculated referring to eqn. (5.44) and the displacement can be calculated referring to eqn. (5.54) as discussed in Section 5.3.4. Because the robot is not moving, ${}^P\mathbf{R}^k_{k-1}$ and ${}^N\mathbf{T}^k_{k-1}$ are expected to be $\mathbf{I}$ and $\mathbf{0}$, respectively. The difference between ${}^P\mathbf{R}^k_{k-1}$ and $\mathbf{I}$, and ${}^N\mathbf{P}^k_{k-1}$ and $\mathbf{0}$ are assessed for the error accumulated by integrating IMU measurements when the robot is not moving. The experimental results will be discussed in Section 6.2.2.

6.1.2 Prior Pose Experiments

When the robot moves, prior pose can be calculated in one of the P-ICP, PV-ICP, and PI-ICP approaches as discussed in Section 5.3. The scan is roughly aligned to the basemap with the selected prior pose. After the rough-alignment, the ICP algorithm is applied for more accurate alignment and outputs the transformation parameters $\mathbf{R}_r$ and $\mathbf{T}_r$, which are regarded as the corrections to the roughly-aligned scan of each approach. Denote the ICP output transformation parameters of (5.8) for P-ICP as $({}_p\mathbf{R}_r, {}_p\mathbf{T}_r)$, of (5.34) for PV-ICP as $({}_v\mathbf{R}_r, {}_v\mathbf{T}_r)$, and of (5.56) for PI-ICP as $({}_i\mathbf{R}_r, {}_i\mathbf{T}_r)$. The rotation matrices R_r can be represented in the angle-axis form [23], where the angle is calculated and used as a numerical metric to evaluate and compare each prior:

$$|\theta| = \text{acos}\left(\frac{tr(\mathbf{R})-1}{2}\right), \tag{6.3}$$

where in this equation, θ is the rotation angle, $\mathbf{R}$ is the rotation matrix, and '$tr(\mathbf{R})$' represents the trace of the matrix $\mathbf{R}$. Small rotation angle indicates the prior pose required only a small rotational correction. Furthermore, the norm of the translation parameter T_r is used to compare the translation error for each prior pose. To evaluate the accuracy of the calculated prior, the ICP output transformation parameters of each approach are compared to evaluate which prior better roughly aligns scan to the map. Moreover, the time consumed for aligning each scan using each approach are compared to evaluate which approaches can be used in real-time applications. The experimental results will be discussed in Section 6.3.

6.2 Experiment Results: IMU Performance Evaluation

This section shows the experimental results of the assessments discussed in Section 6.1. The sampling rate of the IMU evaluated in this section is 200 Hz, while the bandwidth is not provided by the manufacturer for reference. The robot in this application is a ground vehicle, where the IMU and other sensors are mounted on top of it such that the LiDAR can scan the environment above it without occlusion.

6.2.1 Laboratory Measurements V.S. in-container Measurements

In this experiment, an IMU of the same model is placed in the laboratory that ensures the IMU is in a stationary environment and does not move during the data collection. As discussed in Section 6.1, the data in the laboratory (i.e., reference data) is used to calculate the STD ${}_r\sigma_w$ and ${}_r\sigma_a$ for this model of IMU. In each subplot of Figure 6.1, there are two black horizontal lines represent

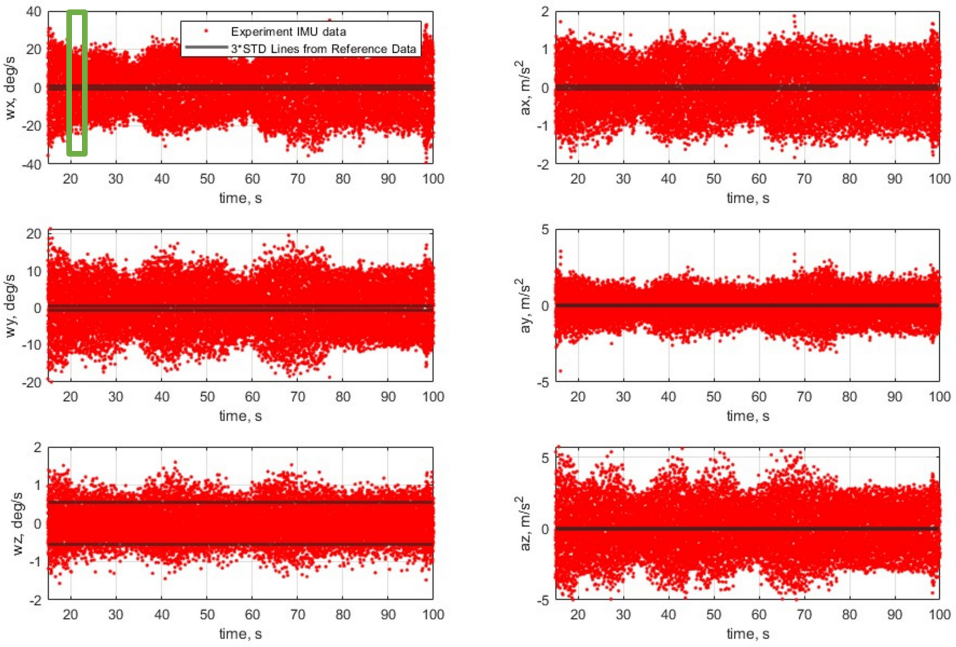

Figure 6.1: IMU measurements in the container when the vehicle is not moving.

$\pm$ 3 times the STD of each term of ${}_r\sigma_w$ and ${}_r\sigma_a$. These black lines are drawn as the reference for comparison to the IMU data collected in the container. The vehicle IMU data (i.e., experiment data) are collected with the vehicle engine on and shown as red points in Figure 6.1. All terms except for w_z of the experiment data have an obvious greater variance than the reference data by visually comparing the red points to the black lines. As results of the data shown in Figure 6.1, for the reference data, the angular rate STD are 0.2021, 0.1856, and 0.1817 $degree/s$ and the acceleration STD are 0.0157, 0.0164, and 0.0132 m/s^2 . For the experiment data, the angular rate STD are 11.9662, 6.1080, and 0.4329 $degrees/s$ and the acceleration STD are 0.6291, 0.8479, and 1.6787 m/s^2. These calculated STD of the two datasets results in a ratio of $r_{w_x} = 59.2218$, $r_{w_y} = 32.9124$, $r_{w_z} = 2.3819$, $r_{a_x} = 40.1494$, $r_{a_y} = 51.8374$, $r_{a_z} = 127.0386$ for each term of IMU measurements.

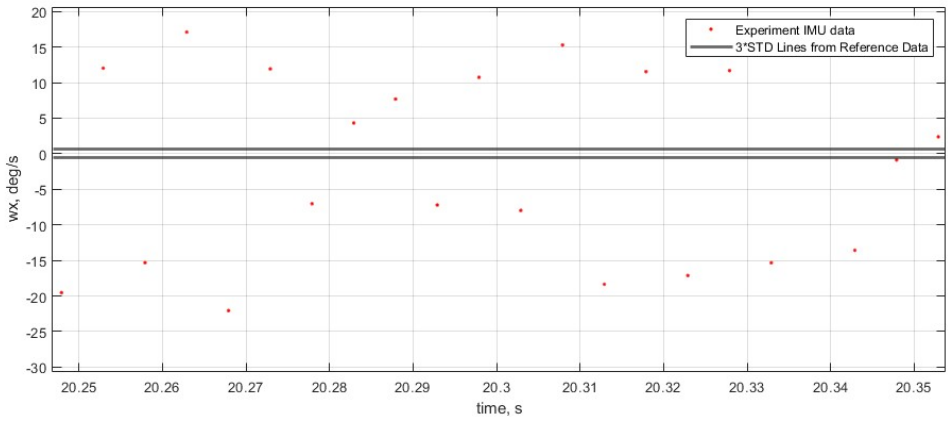

Figure 6.2: A short period of the angular rate w_x component in Figure 6.1.

The angular rate component w_z has the most comparable values with the same term measured in the laboratory. Therefore, in the following experiments of PI-ICP, there will be an additional run to compare the results of PI-ICP using $[w_x, w_y, w_z]^\top$ or $[0, 0, w_z]^\top$ for the measured angular rate.

Furthermore, taking the angular rate as an example, the angular rate ${}^P\boldsymbol{w}_{gN}$ represents the angular rate of the container, which in this case is a hull of a docked ship, and therefore refers to the angular rate of the ship. The angular rate ${}^P\boldsymbol{w}_{gP}$ represents the angular rate of the vehicle which is in the hull. Therefore, the larger angular rate variances are the combined results of the ship and vehicle motion. Taking the w_x component of ${}^P\boldsymbol{w}_{gP}$ as an example for evaluating the ship and vehicle motion, Figure 6.2 shows the a part of red points in the green box in Figure 6.1, where a high frequency (i.e., 100 Hz) component in w_x with an amplitude up to around 20 $degrees/s$ can be observed. A docked ship should not produce such a component, so this part can only be a component belonging to the vehicle motion. The subplots of Figure 6.1 also show a synchronous amplitude increase and decrease. Therefore, the main sources of variances are possibly the same. The most likely source is the vibration of the vehicle engine. The data in Figure 6.2 shows that w_x is

changing too fast with respect to the sampling rate (i.e., 200 Hz) of this IMU. In fact, the $\sim$ 100 Hz signal oscillation is near the IMU Nyquist frequency of 100 Hz, which indicates that this sensor is not appropriate for this application. Future work will include using an IMU with a higher bandwidth and sampling rate or evaluating IMU data when the engine is off, to confirm with more evidence for the effect of vehicle vibration and ship motion on the IMU measurements. In this dissertation, the data of this IMU is still used for calculating the prior poses for each approach and demonstrating the process of evaluating the performance of different priors.

6.2.2 Integration of IMU Angular Rate for Stationary Vehicle

This section shows the results of calculating the vehicle rotation angle every 0.1 second (i.e., the time between two LiDAR scans) using the IMU angular rate measured in the container during initialization when the engine is on. Since the vehicle is not moving in the container at this point, its rotation angle should be 0, so the angle calculated using the IMU measurements shows the error of each approach. Every $\delta t = 0.1$ seconds, about 20 IMU angular rate measurements are used to calculate the vehicle rotation matrix ${}^{P}\hat{\mathbf{R}}_{k-1}^{k}$. The rotation matrix ${}^{P}\hat{\mathbf{R}}_{k-1}^{k}$ can be converted into the axis-angle form, where the magnitude of the rotation angle can be calculated using eqn. (6.3). There are four approach to be evaluated by comparing the rotation angle calculated from ${}^{P}\hat{\mathbf{R}}_{k-1}^{k}$:

1. Assuming the vehicle rotates at a uniform speed during this 0.1 second period, the averaged angular rate is calculated as $\underline{\mathbf{w}} = [\underline{w_x}, \underline{w_y}, \underline{w_z}]$, where $\underline{w_x}$, $\underline{w_y}$, and $\underline{w_z}$ are the averaged angular rate of each term. The rotation matrix can be calculated with $\boldsymbol{\nu} = \underline{\mathbf{w}}\delta t$ using eqns. (5.40) and (5.42). The rotation angles are shown as *black* '+' in Figures 6.3 and 6.4.

2. Assuming the vehicle rotates at a uniform speed during this 0.1 second period, the averaged angular rate around z-axis is calculated as $\underline{\mathbf{w}_z} = [0, 0, \underline{w_z}]$. The rotation matrix can be calculated with $\boldsymbol{\nu} = \underline{\mathbf{w}_z}\delta t$ using eqns. (5.40) and (5.42). The rotation angles are shown as *red* '+' in Figures 6.3 and 6.4.

3. Assuming the vehicle rotates at a uniform speed during each $\Delta t = 0.005$ second period (i.e., time between two IMU measurements), eqn (5.42) is used to calculate the rotation matrix with each angular rate measurement $\mathbf{w} = [w_x, w_y, w_z]^\top$ and eqn. (5.44) is used to integrate the rotation matrix calculated with each $\mathbf{w}$ in this 0.1 second period. The rotation angles are shown as *black* 'o' in Figures 6.3 and 6.4.

4. Assuming the vehicle rotates at a uniform speed during each $\Delta t = 0.005$ second period, eqn (5.42) is used to calculate the rotation matrix with each angular rate measurement $\mathbf{w}_z = [0, 0, w_z]^\top$ and eqn. (5.44) is used to integrate the rotation matrix calculated with each $\mathbf{w}_z$ in this 0.1 second period. The rotation angles are shown as *black* 'o' in Figures 6.3 and 6.4.

As shown in Figures 6.3 and 6.4, for each time window of 0.1 second, Approach-1 and Approach-3 (Approach-2 and Approach-4) result in nearly the same rotation angles such that the black circles and black crosses (red circles and red crosses) overlap. Numerically, Approach-1 shows an average (maximum) of 0.0585 (0.3922) degrees, Approach-2 shows an average (maximum) of 0.0037 (0.0133) degrees, Approach-3 shows an average (maximum) of 0.0584 (0.3922) degrees, Approach-4 shows an average (maximum) of 0.0037 (0.0133) degrees, Based on the results of this analysis, Approach-2 and 4 accumulates smaller angular error while stationary.

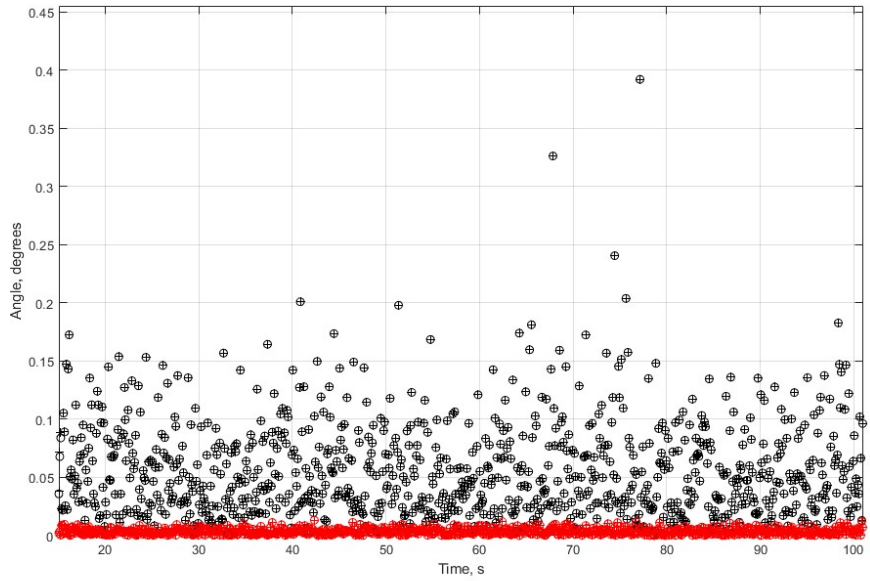

Figure 6.3: All the rotation angles calculated with IMU measurements when the vehicle is stationary.

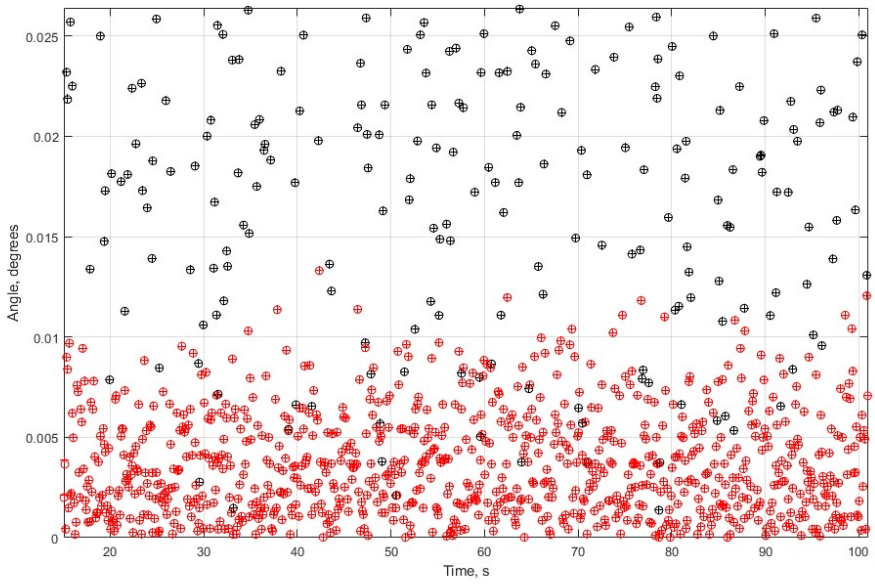

Figure 6.4: Part of the rotation angles calculated with IMU measurements when the vehicle is stationary shown in a smaller angle range.

6.2.3 Integration of IMU Angular Rate for Moving Vehicle

This section shows the results of accumulating IMU angular measurements when the vehicle is moving on a pile. It is mainly rotating around the z-axis of the IMU, but also has pitch and roll rotations as it translates across the uneven pile. The same approaches and symbols introduced in Section 6.2.2 are used with IMU data collected when the vehicle moves. The rotation angles calculated with vehicle moving measurements are shown in Figure 6.5.

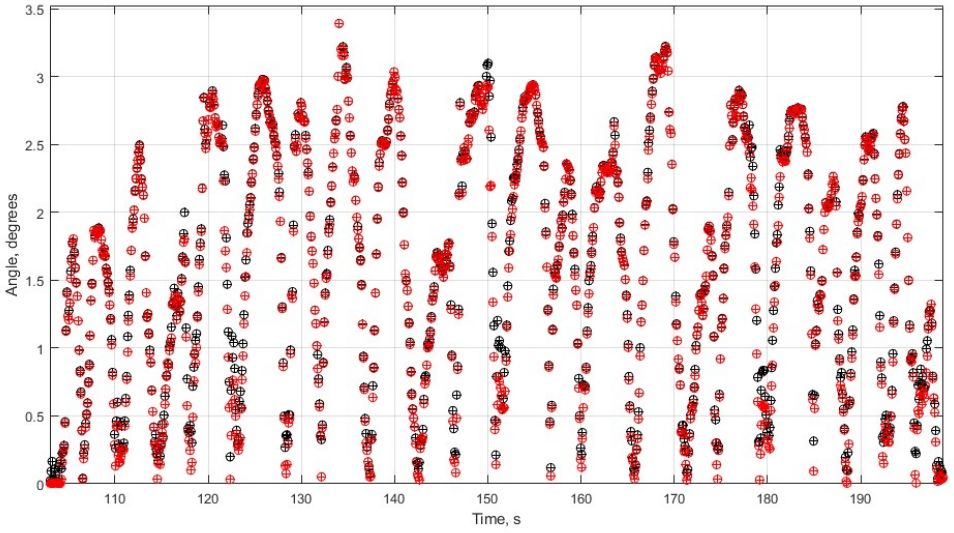

Figure 6.5: The rotation angle calculated with IMU measurements when the vehicle is moving.

The rotation angle calculated with $\mathbf{w}$ and the rotation angle calculated with w_z do not show significant difference because the rotation is mainly around the z-axis. For each calculated angle by integrating $\mathbf{w}$ or w_z, the difference of them are calculated and shown in Figure 6.6. The average (maximum) difference is 0.0440 (0.6343) degrees, which are effects of the w_x and w_y components on the rotation angle. Because the difference is not large, choosing either one of the approaches to integrate the IMU angular measurements makes little difference on the job of roughly aligning the

scan to the basemap. The performance of using the IMU calculated rotation matrix to align the scan will be compared with using the last estimated pose and velocity in Section 6.3 to determine the better approach for the rough-alignment.

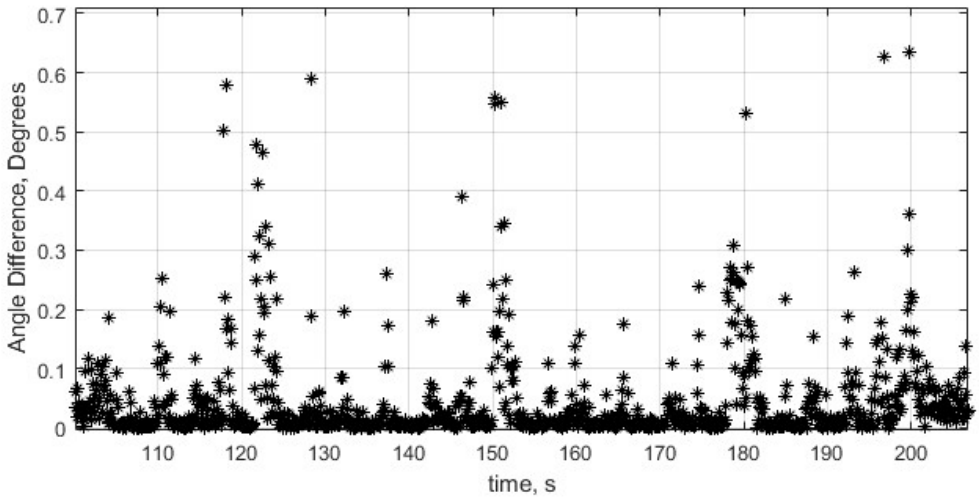

Figure 6.6: The rotation angle difference between integrating $\mathbf{w}$ and integrating w_z. The vehicle is moving in the approximate range of 100 to 200 seconds.

6.3 Experiment Results: Real-Time Localization

This section shows the results of real-time localization experiments to compare the performance of different approaches in Section 5.3 for calculating the prior and roughly aligning the scan to the basemap. The three approaches of P-ICP, PV-ICP, and PI-ICP are compared while the PI-ICP approach runs twice with $\mathbf{w}$ and $\mathbf{w}_z$, which will be denoted as PI-ICP-$\mathbf{w}$ and PI-ICP-w_z in this section.

6.3.1 Localization Results of P-ICP

The localization results using P-ICP is shown in Figure 6.7. As a visual comparison, the trajectory matches the vehicle motion in the experiment video record, which cannot be shown in the dissertation. In addition, it does not show significant discontinuity. If the P-ICP approach allowed significant errors to develop, then large corrections would be required, resulting in discontinuities. The lack of discontinuities is a first indicator that the method is working well.

To numerically evaluate the localization performance, after the alignment of ICP algorithm, the distance between each point p_s in the transformed scan and its closest point p_b in the basemap is calculated as: $d_{bs} = \|p_s - p_b\|$. The averaged distance $\underline{d_{bs}}$ of those point pairs (p_s, p_b) are calculated for this scan:

$$\underline{d_{bs}} = \frac{\sum d_{bs}}{N_s}, \tag{6.4}$$

where N_s is the number of points in the transformed scan. This average distance per scan allows accuracy to be assessed versus time. The results of $\underline{d_{bs}}$ are shown in Figure 6.8 for the localization results of the P-ICP approach. The results show nearly constant values at the beginning and end because the vehicle stayed unmoved at the beginning and end, and received points reflected from the same surfaces. The maximum averaged distance is less than 10 cm, which is enough for the vehicle localization in cooperative transloading.

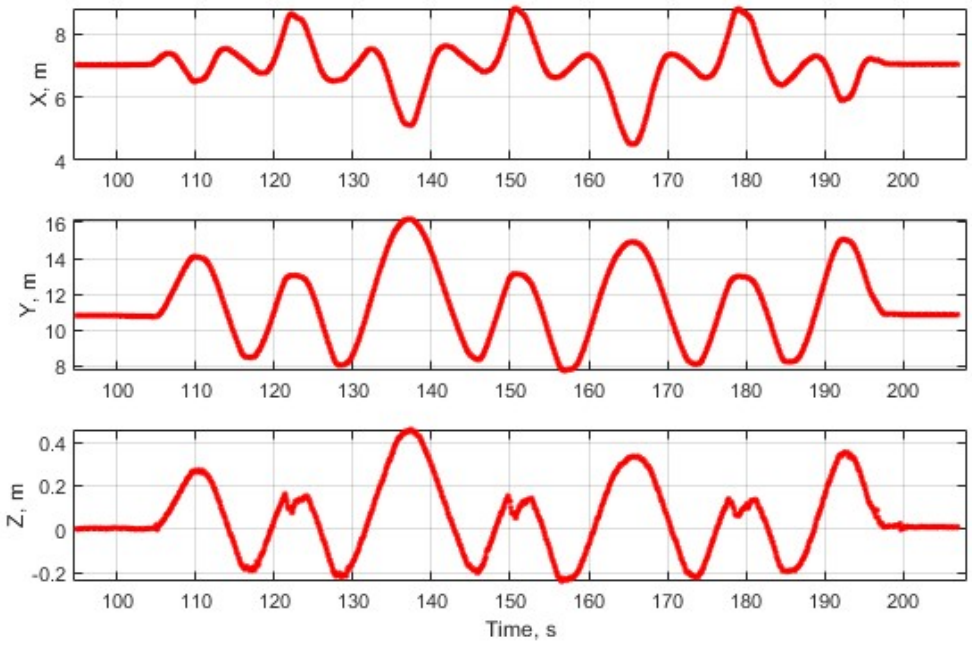

Figure 6.7: The localization results using last estimated pose to calculate the prior pose.

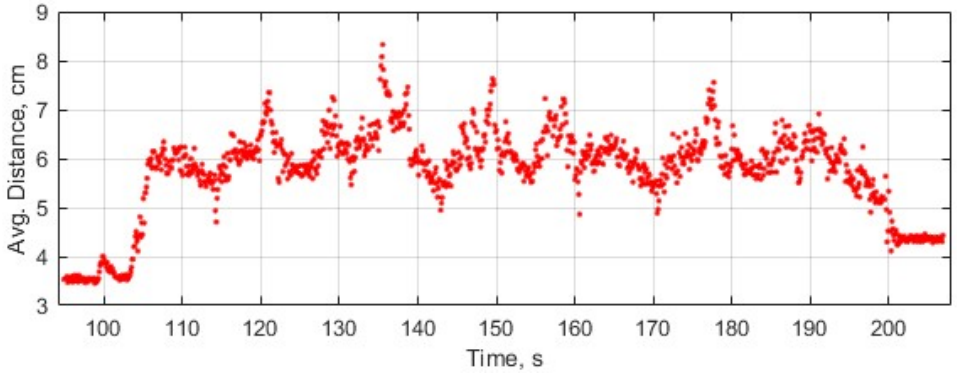

Figure 6.8: The average value of the distance from each point of the aligned scan to its closest point in the basemap.

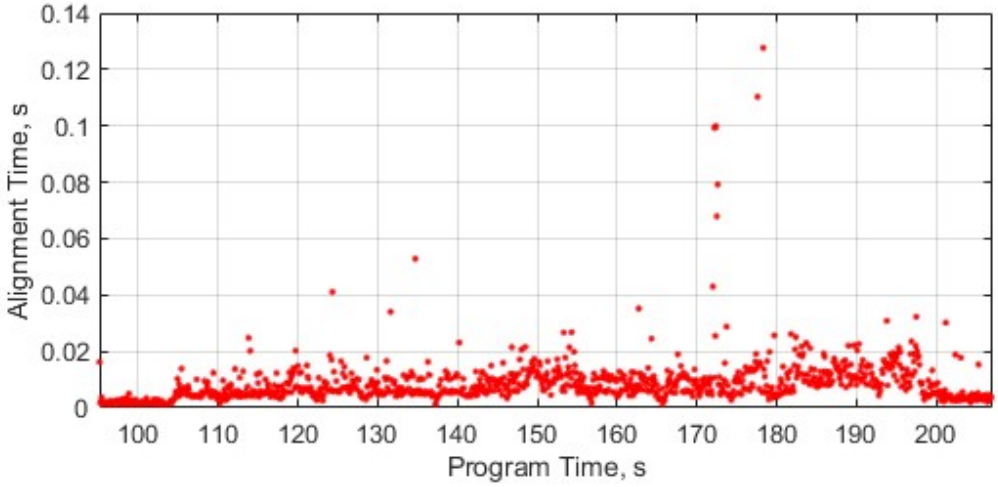

Figure 6.9: The time for processing each scan to update the vehicle pose.

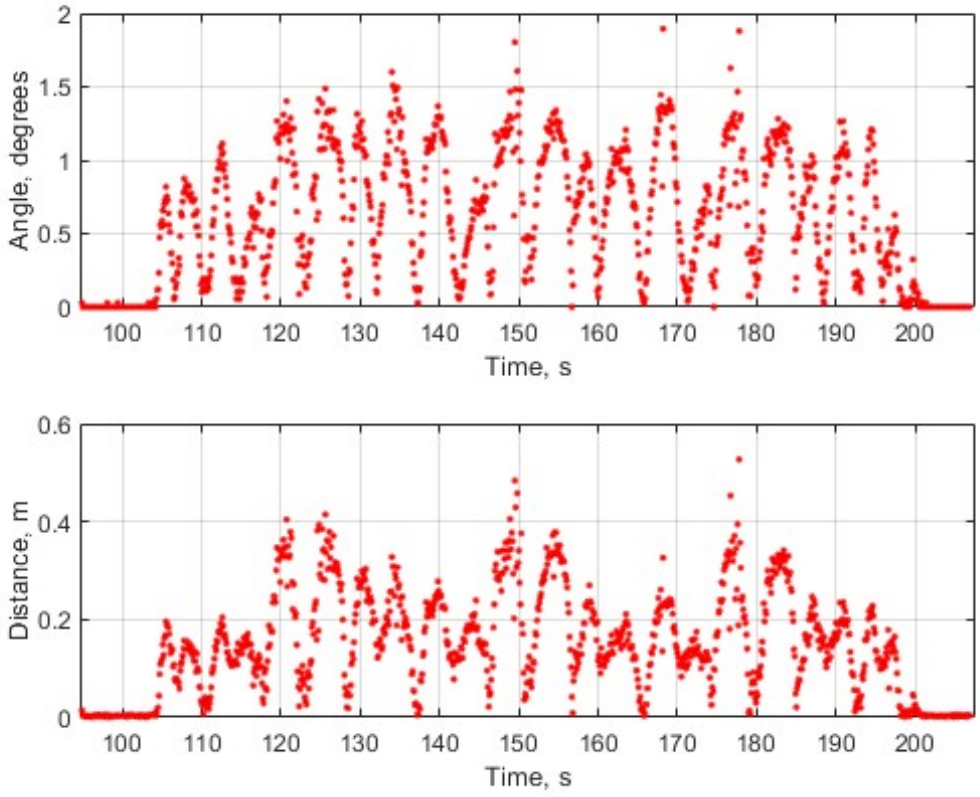

Figure 6.10: The ICP correction rotation angle and translation for each scan.

Another important metric is the time consumed for ICP to align each scan. This is important because each scan has to be processed immediately (i.e., in less than 0.1 seconds before the next scan arrives) such that the vehicle pose can be updated in real-time. The ICP computation time for each scan using this approach is shown in Figure 6.9. The average (maximum) alignment time in this experiment is 0.0087 (0.1277) seconds. A new scan arrives every 0.1 second. Therefore, in most time, the pose can be updated before the new scan arrives. If the vehicle was translating at its maximum in-container speed of 10 m/s, then the average (maximum) position error due to these computational delays would be 0.087 (1.277) m. The maximum position error of 1.277 meter is risky and should be avoided.

Figure 6.10 shows the correction rotation angle calculated from the ICP output rotation parameter ${}_pR_r$ using eqn. (6.3), and the correction norm of the ICP output translation parameter ${}_pT_r$. As the average ICP point matching distance shown in Figure 6.8 is always less than 10 *cm*, the angles and distances in Figure 6.10 indicate the magnitude of the vehicle rotation and translation in the 0.1 second between each two scans: the maximum angle is 1.895 degrees and the maximum distance is 0.527 meters in Figure 6.10, which gives a reference that the maximum angular rate and velocity of the vehicle in this experiment are around 18.95 $degrees/s$ and 5.27 m/s, respectively.

6.3.2 Localization Results Comparison

The results of different approaches are compared to the results shown in Figure 6.7. The differences between the results are shown in Figure 6.11. Each magenta point represents the localization result difference between PV-ICP and P-ICP. Each green point represents the localization result difference between PI-ICP-w_z and P-ICP. Each blue point represents the localization result difference between PI-ICP-**w** and P-ICP. As shown in the figure, the magnitude of the difference are

all less than 10 *cm*. Therefore, the localization results are similar and any of the four approaches successfully calculated the vehicle pose in the experiment. Note that the magenta points show larger difference from the other points. However, this only shows the magenta points are more different from the red point results of Figure 6.7 and the quality of aligning the scan should refer to the next analysis of the average distance after aligning the scan.

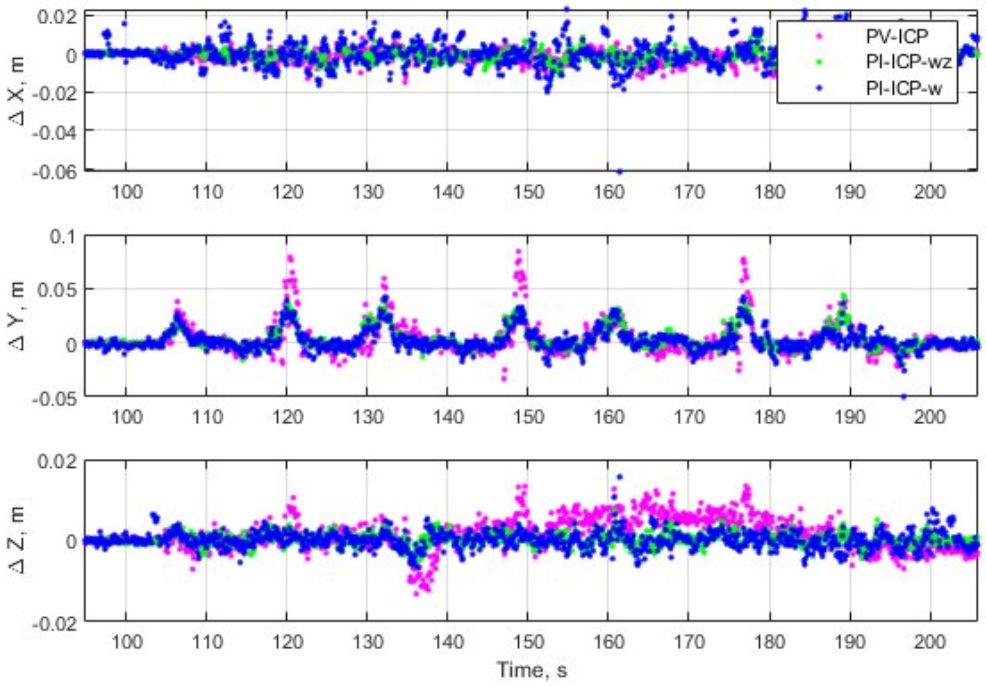

Figure 6.11: The difference of localization results between each other approach and the results shown in Figure 6.7.

The results of the average distance $\underline{d_{bs}}$ of each approach are shown in Figure 6.12. The average (median, maximum) $\underline{d_{bs}}$ for the P-ICP is 5.7568 (5.9273, 8.3277) cm. The average (median, maximum) $\underline{d_{bs}}$ for the PV-ICP is 5.5997 (5.7581, 7.9058) cm. The average (median, maximum) $\underline{d_{bs}}$ for the PI-ICP-w_z is 5.7425 (5.9459, 7.8760) cm. The average (median, maximum) $\underline{d_{bs}}$ for the PI-ICP-**w** is 5.8011 (5.9562, 7.8073) cm. The difference of each statistic between each two

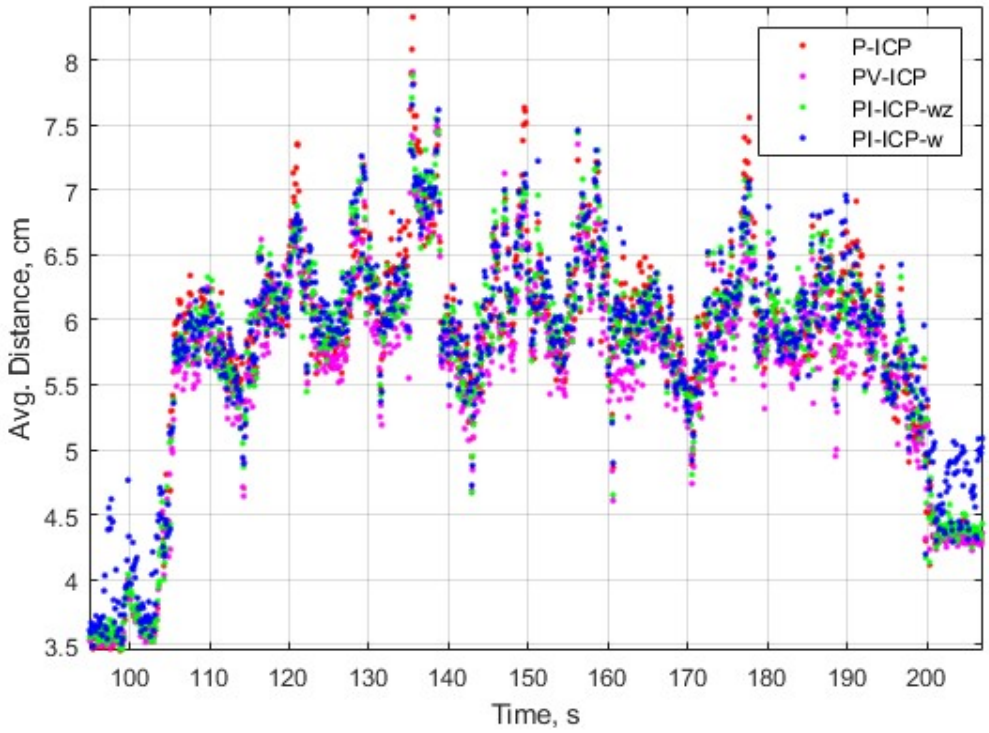

Figure 6.12: The average distance results $\underline{d_{bs}}$ of the four approaches.

approaches are too small to state any one of them is the best approach for calculating the prior pose for localization.

The comparison of the alignment time is shown in Figure 6.13. The average (maximum) alignment time of the P-ICP is 0.0087 (0.1277) seconds. The average (maximum) alignment time of the PV-ICP is 0.0054 (0.0631) seconds. The average (maximum) alignment time of the PI-ICP-w_z is 0.0157 (0.2159) seconds. The average (maximum) alignment time of the PI-ICP-**w** is 0.0180 (0.0889) seconds. The PV-ICP approach uses the shortest average and maximum time to finish the ICP alignment. The maximum time consumed is less than 0.1 second. Therefore, each scan was aligned with PV-ICP before the next scan arrives.

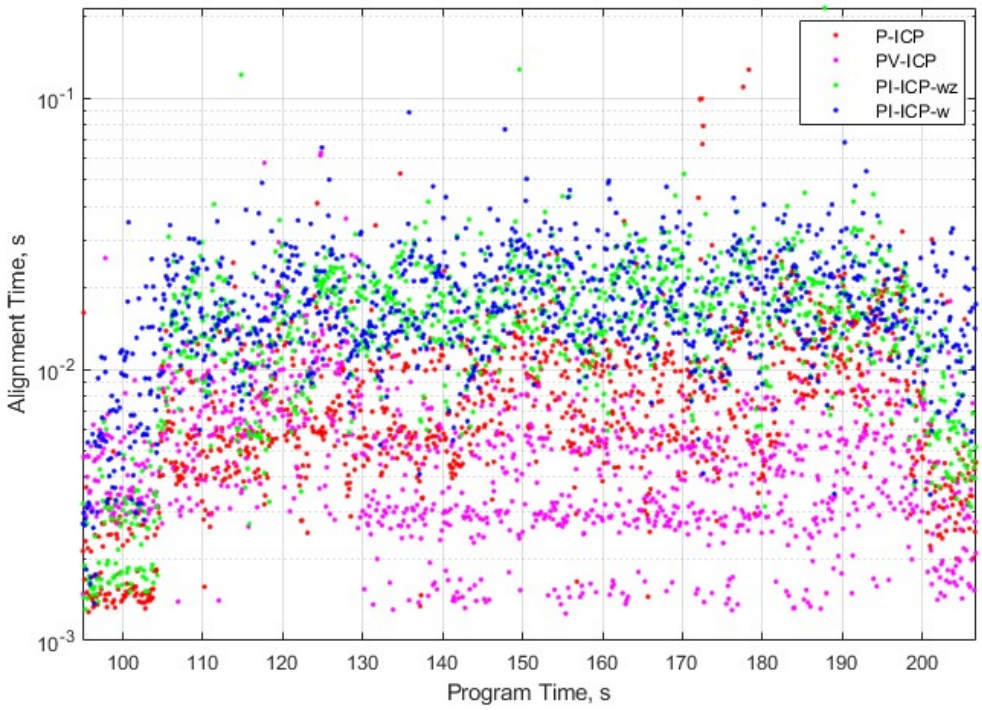

Figure 6.13: The ICP alignment time for four approaches.

The rotation matrix used by each approach to calculate the prior can be converted to the axis-angle form. The angles are called the prior angle. After the ICP alignment, the output R_r can be also converted and its angle is called the correction angle. The prior angle and the correction angle results of each approach are shown in Figure 6.14. As discussed in Section 5.3, the scan is first adjusted with the calculated prior (i.e., rotated by the prior angle) and then aligned to the basemap by ICP algorithm (i.e., rotated by the correction angle). Figure 6.14 shows that PV-ICP adjusted the scans appropriately such that the ICP algorithm requires fewer iterations to align the scans to the basemap and the correction angles are small (i.e., average 0.0925 degrees and maximum 0.4942 degrees). After the rough alignment of PI-ICP-w_z and PI-ICP-**w**, the ICP algorithm needs to do a larger correction rotation than PV-ICP to match the scan to the basemap. An example

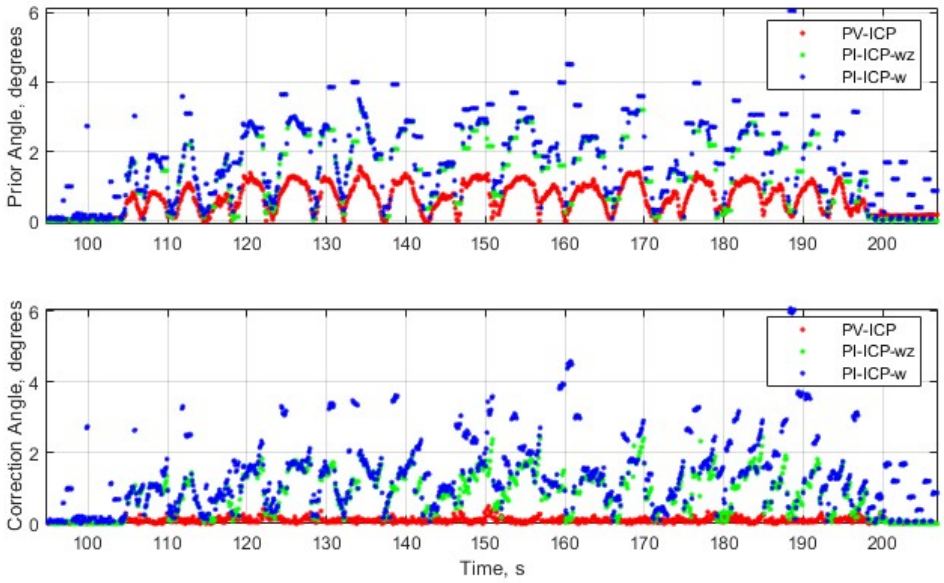

Figure 6.14: The calculated prior and the correction calculated from ICP outputs.

of this is shown with a partial point cloud of one scan, which is being aligned to the basemap, in Figure 6.15. The red points belong to the basemap. The green points belong to the roughly-aligned scan using PV-ICP and the blue points belong to the roughly-aligned scan using PI-ICP-$\mathbf{w}_z$. The blue points have a larger gap from the basemap than the green point. The larger gap results in more ICP iterations for PI-ICP-$\mathbf{w}$ to correct the gap than PV-ICP and more alignment time as shown in Figure 6.13. The result that the IMU does not give a better estimate of the rotation angle needs experiments with better IMU (i.e., higher bandwidth and sampling rate) to confirm. Future research will evaluate the performance when using a properly selected IMU for calculating the prior pose. With the current setup, the PV-ICP approach behaves the best in the real-time localization experiments and succeeds for the cooperative transloading task.

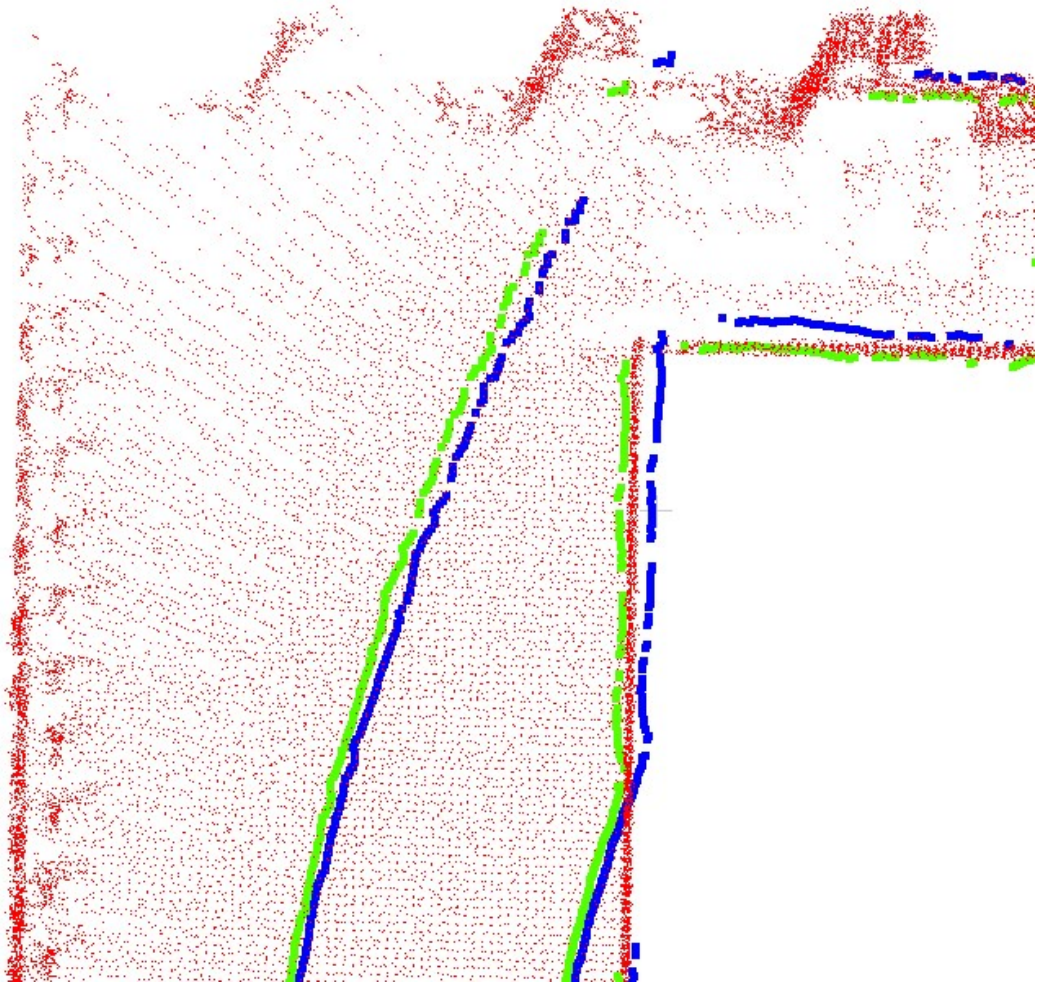

Figure 6.15: The calculated prior and the correction calculated from ICP outputs.

Chapter 7

Conclusions

This section summarizes the contributions of this work and the future research to improve the methods discussed in this dissertation.

7.1 Contribution

The main contributions are:

1. Development of a novel approach combining point pillar based processing with the HT algorithm and knowledge about the hatch to recognize and locate the hatch.

2. Development of a LiDAR-based approach to initialize a shared reference frame and the robot pose in a non-stationary closed-space environment.

3. Evaluation on several approaches based on LiDAR and IMU to calculate the priors for the real-time localization in the non-stationary frame.

4. Development and implementation of a real-time software for localization in cooperative transloading.

7.2 Future Work

In the course of this research, which resulted in a working prototype system, various additional ideas occurred that may be worthy of additional research.

1. The current basemap is built in the initialization stage and unchanged during operations. For large containers or containers with complicated structures, scanning the container at the initial position can be inadequate due to laser dispersion at long distance and occlusion of the structures. A future work is determining eligible points to add to and update the basemap point cloud when the vehicle maneuvers in the container.

2. The current IMU for the experiments is not properly selected. On the one hand, this resulted herein with a demonstration that the system could be implemented without an IMU. On the other hand, it leaves open the opportunity to study further with a properly selected IMU. Therefore, an IMU with higher bandwidth is planned for future work to further test its benefits on calculating an accurate prior pose for localization.

3. The current localization strategy for cooperative transloading depends on the initial hatch extraction results to build the common reference frame. Inaccurate determination of the common reference frame will cause incorrect reporting of the vehicle pose to the crane. One future work is updating the hatch position with the scans collected when the vehicle maneuvers in the container.

4. The bulk cargo surface information is necessary for planning the transloading work. One future research is building the cargo surface with scan inputs from both the LiDAR's on the vehicle and on the crane. Observing the cargo from difference directions can reduce the effects of occlusion.

5. For cooperative transloading between a crane and a robot, only the robot horizontal position is required for the purpose of avoiding collisions. Therefore, a 2D localization approach might be enough while requiring lower computation.

6. The initial pose determination still requires a human to check the vehicle heading direction when placing it in the container such that the reference frame is defined identically for the crane and the vehicle. Using GPS to determine the vehicle heading before it's placed into the container can cancel the human check and further automating the process of initialization.

Bibliography

[1] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(6):641–647, June 1994.

[2] Waqas Ali, Peilin Liu, Rendong Ying, and Zheng Gong. A Feature Based Laser SLAM Using Rasterized Images of 3D Point Cloud. *IEEE Sensors Journal*, 21(21):24422–24430, November 2021.

[3] Dana H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[4] Mitra Basu. Gaussian-based edge-detection methods-a survey. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 32(3):252–260, August 2002.

[5] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, Rudi Penne, and Maarten Weyn. A benchmark survey of rigid 3D point cloud registration algorithms. *International Journal on Advances in Intelligent Systems*, 8:118–127, 2015.

[6] Paul J. Besl and Neil D. McKay. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992.

[7] Johan Augusto Bocanegra, Davide Borelli, Tomaso Gaggero, Enrico Rizzuto, and Corrado Schenone. A novel approach to port noise characterization using an acoustic camera. *Science of The Total Environment*, 808:151903, February 2022.

[8] Andrea Censi. An ICP variant using a point-to-line metric. In *2008 IEEE International Conference on Robotics and Automation*, pages 19–25, May 2008.

[9] Min Young Chang, Suyong Yeon, Soohyun Ryu, and Donghwan Lee. SpoxelNet: Spherical Voxel-based Deep Place Recognition for 3D Point Clouds of Crowded Indoor Spaces. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8564–8570, Las Vegas, NV, USA, October 2020. IEEE.

[10] Yang Chen and Gérard Medioni. Object modeling by registration of multiple range images. In *1991 IEEE International Conference on Robotics and Automation Proceedings*, pages 2724–2729 volume 3, April 1991.

[11] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance Evaluation of RANSAC Family. In *British Machine Vision Conference*, pages 81.1–81.12, London, UK, 2009. British Machine Vision Association.

[12] Hanieh Deilamsalehy and Timothy C. Havens. Sensor fused three-dimensional localization using IMU, camera and LiDAR. In *2016 IEEE SENSORS*, pages 1–3. IEEE, 2016.

[13] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, January 1972.

[14] James H. Elder and StevenW. Zucker. Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):699–716, July 1998.

[15] Jay A. Farrell. *Aided Navigation: GPS with High Rate Sensors*. Electronic engineering. McGraw-Hill.

[16] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[17] Liang Gong, Yongsheng Zhang, Zhengguo Li, and Quanfu Bao. Automated road extraction from LiDAR data based on intensity and aerial photo. In *2010 3rd International Congress on Image and Signal Processing*, pages 2130–2133, Yantai, China, October 2010. IEEE.

[18] Wei Guan, WenTao Li, and Yan Ren. Point cloud registration based on improved ICP algorithm. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1461–1465, Shenyang, June 2018. IEEE.

[19] Florent Guiotte, Minh-Tan Pham, Romain Dambreville, Thomas Corpetti, and Sebastien Lefevre. Semantic Segmentation of LiDAR Points Clouds: Rasterization Beyond Digital Elevation Models. *IEEE Geoscience and Remote Sensing Letters*, 17(11):2016–2019, November 2020.

[20] Timo Hackel, Jan D. Wegner, and Konrad Schindler. FAST SEMANTIC SEGMENTATION OF 3D POINT CLOUDS WITH STRONGLY VARYING DENSITY. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, III-3:177–184, June 2016.

[21] Marcus Hammer, Marcus Hebel, Björn Borgmann, Martin Laurenzis, and Michael Arens. Potential of LiDAR sensors for the detection of UAVs. In *Laser Radar Technology and Applitions XXIII*, page 4, Orlando, USA, May 2018. SPIE.

[22] Rostislav Hulik, Michal Spanel, Pavel Smrz, and Zdenek Materna. Continuous plane detection in point-cloud data based on 3D Hough Transform. *Journal of Visual Communication and Image Representation*, 25(1):86–97, January 2014.

[23] Du Q. Huynh. Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[24] Zeyi Jiang, Xuqing Liu, Mike Ma, Guanlin Wu, and Jay A. Farrell. LiDAR-Based Hatch Localization. *Remote Sensing*, 14(20):5069, October 2022.

[25] G. Ajay Kumar, Ashok Kumar Patil, Rekha Patil, Seong Sill Park, and Young Ho Chai. A LiDAR and IMU integrated indoor navigation system for UAVs and its application in real-time pipeline classification. *Sensors*, 17(6):1268, 2017.

[26] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, Long Beach, USA, June 2019. IEEE.

[27] Menggang Li, Hua Zhu, Shaoze You, Lei Wang, and Chaoquan Tang. Efficient Laser-Based 3D SLAM for Coal Mine Rescue Robots. *IEEE Access*, 7:14124–14138, 2019.

[28] Shihua Li, Jingxian Wang, Zuqin Liang, and Lian Su. Tree point clouds registration using an improved ICP algorithm based on kd-tree. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 4545–4548. IEEE, 2016.

[29] Yangming Li and Edwin B. Olson. A General Purpose Feature Extractor for Light Detection and Ranging Data. *Sensors*, 10(11):10356–10375, November 2010.

[30] Haibo Liu, Tianran Liu, Yapeng Li, Mengmeng Xi, Te Li, and Yongqing Wang. Point cloud registration based on MCMC-SA ICP algorithm. *IEEE Access*, 7:73637–73648, 2019.

[31] Kok-Lim Low. Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration. page 3.

[32] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, Hamburg, Germany, September 2015. IEEE.

[33] Hao Men, Biruk Gebre, and Kishore Pochiraju. Color point cloud registration with 4D ICP algorithm. In *2011 IEEE International Conference on Robotics and Automation*, pages 1511–1516. IEEE, 2011.

[34] Xiaoli Meng, Heng Wang, and Bingbing Liu. A robust vehicle localization approach based on gnss/imu/dmi/lidar sensor fusion for autonomous vehicles. *Sensors*, 17(9):2140, 2017.

[35] Chao Mi, Youfang Huang, Haiwei Liu, Yang Shen, and Weijian Mi. Study on Target Detection & Recognition Using Laser 3D Vision Systems for Automatic Ship Loader. *Sensors & Transducers*, 158(11):436–442, November 2013.

[36] Chao Mi, Yang Shen, Weijian Mi, and Youfang Huang. Ship Identification Algorithm Based on 3D Point Cloud for Automated Ship Loaders. *Journal of Coastal Research*, 73:28–34, March 2015.

[37] Chao Mi, Zhi-Wei Zhang, You-Fang Huang, and Yang Shen. A fast automated vision system for container corner casting recognition. *Journal of Marine Science and Technology*, 24(1):54–60, February 2016.

[38] Yang Miao, Changan Li, Zhan Li, Yipeng Yang, and Xinghu Yu. A novel algorithm of ship structure modeling and target identification based on point cloud for automation in bulk cargo terminals. *Measurement and Control*, 54(3-4):155–163, March 2021.

[39] Andreas Pfrunder, Paulo V. K. Borges, Adrian R. Romero, Gavin Catt, and Alberto Elfes. Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2601–2608, September 2017.

[40] François Pomerleau, Francis Colas, and Roland Siegwart. A Review of Point Cloud Registration Algorithms for Mobile Robotics. *FNT in Robotics*, 4(1):1–104, 2015.

[41] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP variants on real-world data sets: Open-source library and experimental protocol. *Autonomous Robots*, 34(3):133–148, April 2013.

[42] Arya Senna Abdul Rachman. 3D-LIDAR Multi Object Tracking for Autonomous Driving: Multi-target Detection and Tracking under Urban Road Uncertainties. Master's thesis, TU Delft Mechanical, Maritime and Materials Engineering, November 2017.

[43] Farzana S. Rahman. *Real-time Sub-meter Vehicle Positioning: Low-cost GNSS-Aided INS*. Ph.D., University of California, Riverside, United States – California.

[44] Thinal Raj, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain. A Survey on LiDAR Scanning Mechanisms. *Electronics*, 9(5):741, April 2020.

[45] Walid Remmas, Ahmed Chemori, and Maarja Kruusmaa. Diver tracking in open waters: A low-cost approach based on visual and acoustic sensor fusion. *Journal of Field Robotics*, 38(3):494–508, May 2021.

[46] Zhuli Ren and Liguan Wang. Accurate Real-Time Localization Estimation in Underground Mine Environments Based on a Distance-Weight Map (DWM). *Sensors*, 22(4):1463, February 2022.

[47] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Robotics: Science and Systems*, volume 2, page 435. Seattle, WA.

[48] Yang Shen, Weijian Mi, and Zhiwei Zhang. A Positioning Lockholes of Container Corner Castings Method Based on Image Recognition. *Polish Maritime Research*, 24(s3):95–101, November 2017.

[49] Xiaojing Shi, Tao Liu, and Xie Han. Improved Iterative Closest Point (ICP) 3D point cloud registration algorithm based on point cloud filtering and adaptive fireworks for coarse registration. *International Journal of Remote Sensing*, 41(8):3197–3220, 2020.

[50] Jingxuan Sun, Boyang Li, Yifan Jiang, and Chih-yung Wen. A Camera-Based Target Detection and Positioning UAV System for Search and Rescue (SAR) Purposes. *Sensors*, 16(11):1778, October 2016.

[51] Fredrik Svanstrom, Cristofer Englund, and Fernando Alonso-Fernandez. Real-Time Drone Detection and Tracking With Visible, Thermal and Acoustic Sensors. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 7265–7272, Milan, Italy, January 2021. IEEE.

[52] Andrew Symington, Sonia Waharte, Simon Julier, and Niki Trigoni. Probabilistic target detection by camera-equipped UAVs. In *2010 IEEE International Conference on Robotics and Automation*, pages 4076–4081, Anchorage, AK, May 2010. IEEE.

[53] Fayez Tarsha-Kurdi, Zahra Gharineiat, Glenn Campbell, Mohammad Awrangjeb, and Emon Kumar Dey. Automatic Filtering of Lidar Building Point Cloud in Case of Trees Associated to Building Roof. *Remote Sensing*, 14(2):430, January 2022.

[54] Fayez Tarsha-Kurdi, Tania Landes, and Pierre Grussenmeyer. Hough-transform and extended ransac algorithms for automatic detection of 3D building roof planes from Lidar data. In *ISPRS Workshop on Laser Scanning and SilviLaser*, volume 36, pages 407–412, 2007.

[55] Peter JG Teunissen and Oliver Montenbruck. *Springer handbook of global navigation satellite systems*, volume 10. Springer, 2017.

[56] Paweł Trybała. LiDAR-based Simultaneous Localization and Mapping in an underground mine in Złoty Stok, Poland. *IOP Conference Series: Earth and Environmental Science*, 942(1):012035, November 2021.

[57] Victor Vaquero, Ely Repiso, and Alberto Sanfeliu. Robust and Real-Time Detection and Tracking of Moving Objects with Minimum 2D LiDAR Information to Advance Autonomous Cargo Handling in Ports. *Sensors*, 19(1):107, December 2018.

[58] Anh-Vu Vo, Linh Truong-Hong, Debra F. Laefer, and Michela Bertolotto. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:88–100, June 2015.

[59] Wei Wang, Jun Liu, Chenjie Wang, Bin Luo, and Cheng Zhang. DV-LOAM: Direct Visual LiDAR Odometry and Mapping. *Remote Sensing*, 13(16):3340, August 2021.

[60] Wei Xin and Jiexin Pu. An improved ICP algorithm for point cloud registration. In *2010 International Conference on Computational and Information Sciences*, pages 565–568. IEEE, 2010.

[61] Yuquan Xu, Vijay John, Seiichi Mita, Hossein Tehrani, Kazuhisa Ishimaru, and Sakiko Nishino. 3D point cloud map based vehicle localization using stereo camera. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 487–492, June 2017.

[62] Yusheng Xu, Xiaohua Tong, and Uwe Stilla. Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, 126:103675, 2021.

[63] Guanghui Xue, Jinbo Wei, Ruixue Li, and Jian Cheng. LeGO-LOAM-SC: An Improved Simultaneous Localization and Mapping Method Fusing LeGO-LOAM and Scan Context for Underground Coalmine. *Sensors*, 22(2):520, January 2022.

[64] Lina Yang, Yuchen Li, Xichun Li, Zuqiang Meng, and Huiwu Luo. Efficient plane extraction using normal estimation and RANSAC from 3D point cloud. *Computer Standards & Interfaces*, 82:103608, August 2022.

[65] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly Coupled 3D Lidar Inertial Odometry and Mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3144–3150, May 2019.

[66] Hee-Joo Yoon, Young-Chul Hwang, and Eui-Young Cha. Real-time container position estimation method using stereo vision for container auto-landing system. In *2010 International Conference on Control, Automation and Systems (ICCAS 2010)*, pages 872–876, Gyeonggi-do, South Korea, Oct. 2010. IEEE.

[67] Qingming Zhan, Yubin Liang, and Yinghui Xiao. Color-based segmentation of point clouds. *Laser scanning*, 38(3):155–161, 2009.

[68] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41(2):401–416, February 2017.

[69] Shaojiang Zhang, Yanning Guo, Qiang Zhu, and Zhiyuan Liu. Lidar-IMU and Wheel Odometer Based Autonomous Vehicle Localization System. In *2019 Chinese Control And Decision Conference (CCDC)*, pages 4950–4955, June 2019.

[70] Xin Zheng and Jianke Zhu. Efficient LiDAR Odometry for Autonomous Driving. *IEEE Robotics and Automation Letters*, 6(4):8458–8465, October 2021.