Mobile Robot Manipulator System Design for Localization and Mapping in Cluttered Environments

by

Chia-Sung Liu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2018

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In this thesis, a compact mobile robot has been developed to build real-time 3D maps of hazards and cluttered environments inside damaged buildings for rescue tasks using visual Simultaneous Localization And Mapping (SLAM) algorithms. In order to maximize the survey area in such environments, this mobile robot is designed with four omni-wheels and equipped with a 6 Degree of Freedom (DOF) robotic arm carrying a stereo camera mounted on its end-effector. The aim of using this mobile articulated robotic system is monitor different types of regions within the area of interest, ranging from wide open spaces to smaller and irregular regions behind narrow gaps.

In the first part of the thesis, the robot system design is presented in detail, including the kinematic systems of the omni-wheeled mobile platform and the 6-DOF robotic arm, estimation of the biases in parameters of these kinematic systems, the sensors and calibration of their parameters. These parameters are important for the sensor fusion utilized in the next part of the thesis, where two operation modes are proposed to retain the camera pose when the visual SLAM algorithms fail due to variety of the region types. In the second part, an integrated sensor data fusion, odometry and SLAM scheme is developed, where the camera poses are estimated using forward kinematic equations of the robotic arm and fused to the visual SLAM and odometry algorithms. A modified wavefront algorithm with reduced computational complexity is used to find the shortest path to reach the identified goal points. Finally, a dynamic control scheme is developed for path tracking and motion control of the mobile platform and the robot arm, with sub-systems in the form of PD controllers and extended Kalman filters. The overall system design is physically implemented on a prototype integrated mobile robot platform and successfully tested in real-time.

Acknowledgements

I would like to thank my supervisor, Dr. Baris Fidan, for giving me the opportunity to study and work with a great team at the University of Waterloo. During these years, his helpful guidance and patient support have taught me a great deal about the implementation of cutting edge robotic research.

I would like to thank the members of my committee, Dr. William Melek and Dr. Stephen Smith, for their recommendations to improve this thesis. On my coursework, I also want to thank Prof. Baris Fidan, Prof. Behrad Khamesee, Prof. John Zelek, Prof. Soo Jeon, Prof. Steven Waslander (in alphabetical order). I was joyfully immersed in the knowledge they taught in their lectures. These courses are also the foundation of this research.

I would like to thank my teammates, Mr. and Mrs. Zengin and Assylbek Dakibay, who gave me constructive feedback and support on the development of this robot. Lastly, I want to thank Elisabeth, Dave ,and Grant for providing comments to improve the clarity of this thesis. Last but not least, I would like to sincerely thank all the teachers and friends who had ever encouraged me when I was in the low tide of my life.

Dedication

I would like to dedicate this thesis to my parents, wife and kids.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AR Augmented Reality

DBoW an open source C++ library for indexing and converting images into a bag-of-word representation

D-H parameters Denavit-Hartenberg parameters

DOF Degrees of Freedom

EF End Effector

IMU Inertial Measurement Unit

ORB Oriented FAST and Rotated BRIEF

PID Proportional-Integral-Derivative

SIFT Scale Invariant Feature Transform

SLAM Simultaneous Localization And Mapping

SURF Speeded-Up Robust Features

VR Virtual Reality

Chapter 1

Introduction

Autonomous robots play an important role in exploring unknown or dangerous environments such as collecting rock samples on Mars and surveying collapsed buildings for rescue purposes [7, 89]. In such tasks, perceptive sensors on the robots reconstruct the 3D environment they travese, called the 'map' of an environment. A map produced by state-of-the-art SLAM algorithms contains precious information such as textures, shapes, positions, and geometries. The map can be contained in the robot's local memory or shared with others over a wireless network. A regional map done by a mobile robot is like a piece of puzzle for the unknown environment.

The map can be used for further applications, including the Augmented Reality (AR), Virtual Reality (VR) virtual reality tours, and path planning. For the rescue applications, real-time mapping is exceptionally beneficial to the crew for preparing the safety protective equipment and planning the safest route to reach wounded survivors.

Autonomous navigation is contingent on the availability of an accurate map. However, mapping requires precise posture estimates of the mobile robot. This challenge of solving mapping and localization iteratively has lead to development of Simultaneous Localization And Mapping (SLAM) in the field of robotics.

In the past decade, there has been a significant amount of research on developing novel SLAM algorithms and numerical methods to efficiently implement these SLAM algorithms in real-time on affordable hardware platforms (sensors, CPU, and GPU). Another large contribution to the field of SLAM comes from active open source communities, filling the gap between theoretical studies and coding for real-time implementation.

This thesis proposes a methodology and develops the algorithms, mechanical design, and instrumentation, to implement real-time SLAM in GPS-denied and cluttered envi-

ronments with a mobile robotic system composed of an omni-mobile platform, a low-cost camera, and Inertial Measurement Unit (IMU).

In this thesis, the prototype of a compact mobile robot is built based on the kinematic models, SLAM algorithms, and control schemes presented. This mobile robot is equipped a robotic arm with a stereo camera fixed on the end-effector. This eye-in-hand setup can drive the camera into gaps and holes to observe scenes occluded by obstacles. Meanwhile, the camera on the end-effector is streaming the video to SLAM algorithms that can reconstruct a 3D map in real-time. At the end, a modified wavefront algorithm is presented to find the shortest route on the gathered map.

1.1 Motivation

On March 11, 2011, the Fukushima nuclear disaster resulted in three nuclear meltdowns and hydrogen-air chemical explosions, from a tsunami following the Thoku earthquake in Japan. Due to high levels of the radiation, rescuers could not reach the damaged reactor buildings and many surrounding areas of the nuclear power plant. If autonomous mobile robots were available at the time, that could have entered and mapped the real-time scene inside the building. This would help first responses plan and minimize human casualties and nuclear pollution. Furthermore, mobile robots equipped with manipulators could also take some actions to mitigate the disaster such as closing control valves. That were rendered inoperable from the control room.

In Canada, a coal mine exploded in Nova Scotia on May 9, 1992. The explosion killed every person in the mine and tore off the metal roof at the pit entrance. The tunnel inside the coal mine contained hazardous gas and loose rocks. That made it dangerous to send rescue crews without knowing the best path inside the tunnel. If a robot could explore the terrain ahead, it would be helpful to prevent the crew from entering any dangerous zones.

1.2 The Mobile Platform Mapping and Surveillance Problem

In a collapsed building, the scene is not ordinary. Fallen walls or heavy shelves might block pathways. In a robotic exploration and rescue scenario in such an environment, the mobile robot cannot move further to survey the other side of the blocking obstacle. This thesis mainly focuses on mapping in this kind of environments with narrow gaps using

a robot manipulator mounted on a wheeled robot. This kind of setting can enhance the reachability of the perception sensor mounted on the End Effector (EF) to provide the more in-depth search of the clutter building environment. Due to perception sensors being fragile, collision avoidance needs to be considered.

A mobile robot is designed as compact as possible to travel inside the cluttered area. When a mobile robot reaches to the targets, e.g. disaster survivors, other robots can rely on the map to be shared by the reaching robot, and then carry foods and tools to the destination where peers need supports.

1.3 Contributions

The main contribution of this thesis, which is novel to the best of our knowledge, is to integrate and implement the eye-in-hand manipulator SLAM in GPS denied and cluttered environments for mobile robots. In damaged or contaminated buildings, there are many obstacles scattered throughout the floor. The motion of robots is limited by these obstacles. The proposed system can go around obstacles to achieve gap mapping and maximize the surveillance area without lost tracking.

The other contribution is on the path planner. A modified wavefront algorithm is proposed aiming to increase the efficiency to compute the effective and reachable destination where people or robots need supports on a given map.

1.4 Organization

This thesis is organized as follows: The background literature review on SLAM and path planning, presented in Chapter 2. Chapter 3 defines the cluttered environment surveillance mapping problem of interest with detailed specifications, and provides the general description of the mobile robot manipulator system proposed for this task, including the mechanical design and sensor instrumentation. SLAM algorithm development, motion planning, and low-level control design, are presented in Chapters 4, 5, 6, respectively. The system architecture is outlined in the Fig.1.1. The experimental results and conclusions are provided in the last two chapters.

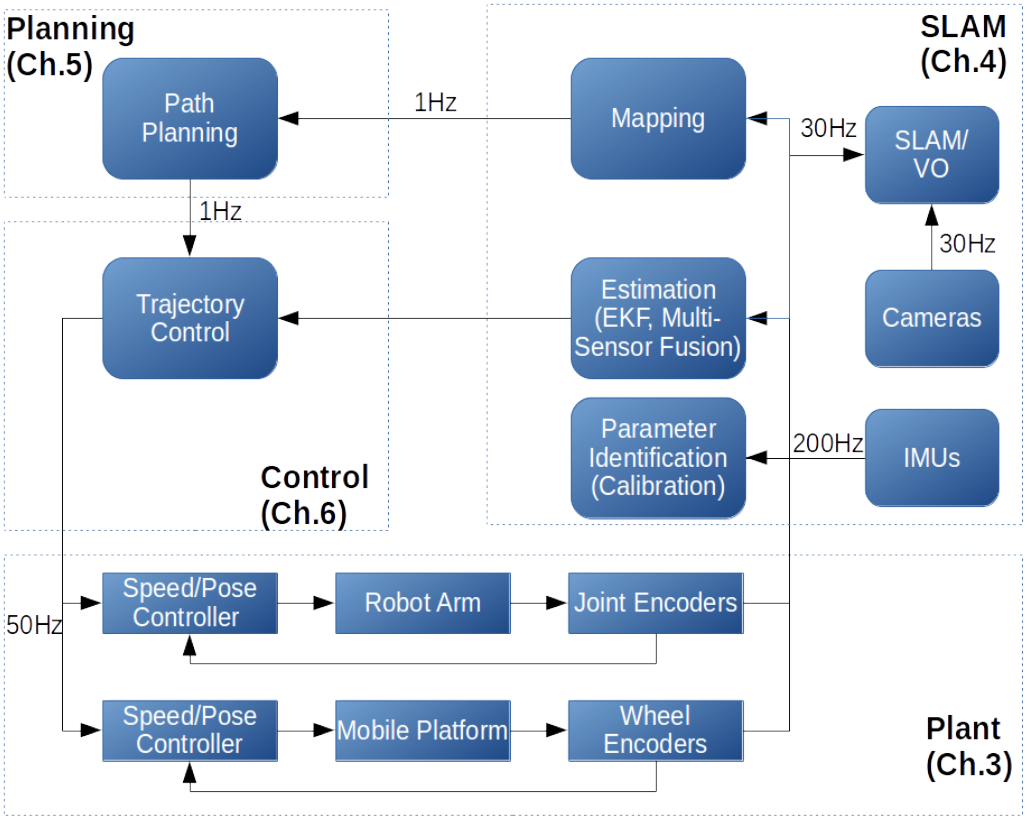

Figure 1.1: System roadmap.

Chapter 2

Background and Literature Review

In [18, 34], SLAM is defined as the problem, for an autonomous mobile robot, of building the consistent map of an unknown environment and localizing itself within this map. Therefore, the task of a rescue robot to explore an unknown environment can be formulated as SLAM. In late 1980's, a number of researchers including Peter Cheeseman, Jim Crowley, and Hugh Durrant-Whyte, Raja Chatila, Oliver Faugeras, Randal Smith, and others concluded that consistent probabilistic mapping is a fundamental problem in SLAM. Since then, SLAM process is generally modelled as the standard Bayesian formulation in which estimates the relative positions to landmarks and a mobile robot pose recursively.

In the early stage, the key papers produced by Smith and Cheesman [81] and Durrant-Whyte [35] have shown that the correlations between estimates of the location of different landmarks in a map grow with successive observations. In increasing number of observations, computational complexity is a crucial issue. Later on, researchers focused on a series of problems on building the consistent map and estimating the robot state. Thrun [92] proposed the Kalman-filter-based SLAM method which achieved the convergence between the probabilistic localization and mapping. It is briefly introduced in the next section.

The research in SLAM has grown gradually in recent years. Some researchers are generous to share their implementations as open source code [11] with public. That gets more attention than ever. Recently, the researches have been mainly focusing on four key areas:

1. Real-time implementation (computational complexity).
2. Map representation.

3. Data association (loop-closure detection and feature matching).

4. Measurement sensors, such as GPS, IMU, Wheel Odometer, LiDar, RGB-D camera, Stereo camera, and radar, etc. They can measure the robot state and landmark directly or indirectly. (See Fig. 2.1)

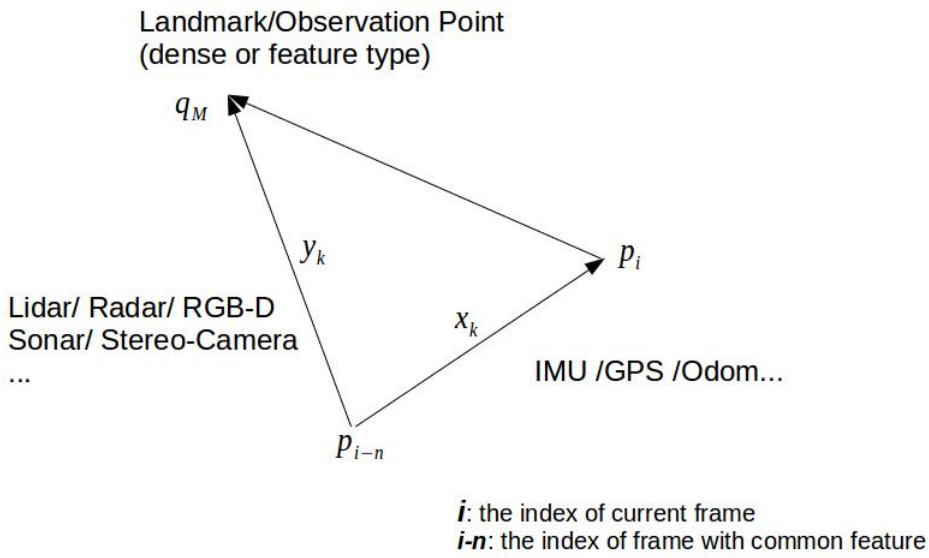

Figure 2.1: The major sensors used in SLAM process and their associated metric measurements.

2.1 Probabilistic SLAM

In the probabilistic SLAM approach [34], following the theory in [91, 92], a robot state and its observing map can be described in terms of the following conditional probability:

$$P(\mathbf{x}_t, \mathbf{m}|\mathbf{Y}_{0:t}, \mathbf{U}_{0:t}, \mathbf{x}_0). \tag{2.1}$$

where $\mathbf{x}_t$ is the state vector of mobile robot at time t, $\mathbf{m}$ is the vector of time-invariant landmarks in the map, $\mathbf{Y}_{0:t} = [\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_t]$, $\mathbf{U}_{0:t} = [\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_t]$, $\mathbf{x}_0$ is the initial state of the robot.

The motion model of mobile robot can be described by another conditional probability,

$$P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \tag{2.2}$$

The observation model of the mobile robot can be described in the form

$$P(\mathbf{y}_t|\mathbf{x}_t, \mathbf{m}). \tag{2.3}$$

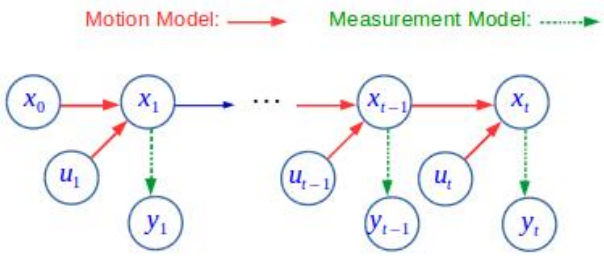

Figure 2.2: Markov assumptions of the Bayes filter in [91].

The SLAM algorithm in [91, 92] is implemented as a recursion of the prediction-update

$$P(\mathbf{x}_t, \mathbf{m}|\mathbf{Y}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0) = \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) P(\mathbf{x}_{t-1}, \mathbf{m}|\mathbf{y}_{0:t-1}, \mathbf{u}_{0:t-1}, \mathbf{x}_0) \mathrm{d}\mathbf{x}_{t-1} \tag{2.4}$$

and the measurement-update

$$P(\mathbf{x}_t, \mathbf{m}|\mathbf{Y}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0) = \frac{P(\mathbf{y}_t|\mathbf{x}_t, \mathbf{m}) P(\mathbf{x}_t, \mathbf{m}|\mathbf{Y}_{0:t-1}, \mathbf{u}_{0:t}, \mathbf{x}_0)}{P(\mathbf{Y}_t|\mathbf{Y}_{0:t-1}, \mathbf{u}_{0:t})} \tag{2.5}$$

Based on Markov assumptions as shown in Fig.2.2, the motion model is described in the form of a state-space model with additive Gaussian noise $\mathbf{w}_t$, as follows:

$$P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \Longleftrightarrow \mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t. \tag{2.6}$$

The measurement model is described, considering an additive Gaussian noise $\mathbf{v}_t$,

$$P(\mathbf{Y}_t|\mathbf{x}_t, \mathbf{m}) \Longleftrightarrow \mathbf{Y}_t = \mathbf{h}(\mathbf{x}_t, \mathbf{m}) + \mathbf{v}_t. \tag{2.7}$$

$\mathbf{f}(\mathbf{x}_{t-1}, \mathbf{u}_t)$ in Eq.2.6 and $\mathbf{h}(\mathbf{x}_t, \mathbf{m})$ in Eq.2.7 are nonlinear functions. The most common algorithms used to solve the SLAM problem defined above include EKF-SLAM [19] and Rao-Blackwellized particle filter, or FastSLAM algorithm [93].

In vision-based SLAM implementations, the digital camera is not only a cost-effective sensor but also comes with the richest information, such as colours, textures, intensity and a huge number of pixels. It can adapt to indoor and outdoor environments. Meanwhile, there are many profound researches in computer vision area transferable to solve SLAM issues. In this thesis, a monochrome stereo camera is chosen to be the main sensor to observe the unknown environment. Wheel odometer and IMU are utilized to measure the robot states. An articulated robot arm is used to maximize the explore area.

2.1.1 Visual SLAM Algorithms

Following are two the state-of-art approaches to data association in the passive vision-based SLAM algorithms. In this thesis, these two SLAM algorithms are utilized in our experiments.

1. **Feature-based SLAM:** The features are described by the unchanged characters at certain specific locations on the map. There are many descriptors from the computer vision category, such as Oriented FAST and Rotated BRIEF (ORB) [77] , Scale Invariant Feature Transform (SIFT) [64] and Speeded-Up Robust Features (SURF) [20]. used to be the signatures of features. The descriptor, a set of numbers, can be used to match the corresponding points in different images. For instance, in Fig. 2.3, the matching points in the left image and the right image are linked by lines in light blue colour. The adjustable threshold of the feature number depends on the system computation capability. The 7-point/8-point correspondences algorithm[50] or optimization-on-manifold can be applied to solve the transformation of one image to another. On the contrary, the mismatch feature points may increase the error dramatically. Concerning the risk of the mismatch, RANSAC [39] is one kind of algorithm to reject the outliers. The number of feature points depends on the texture of an image. Overall, this number is still less than all the pixels in an image. The feature points are also called 'sparse' feature points. The advantage of this type of SLAM is sufficiency to compute the transformation matrix from an image frame to

another frame due to sparse points. The disadvantage is its sensitivity to the blurred images. Using a high-speed camera which takes a picture in fast frame rate can overcome this issue. However, the cost is higher than standard cameras under 30 frames per second(fps). ORB-SLAM2[74] is an efficient algorithm for this domain.

Figure 2.3: Matching the corresponding feature points in different images. These temporal images are taken by Tara camera in our proposed system.

2. **Dense SLAM:** Dense SLAM algorithm uses the photometric value of each pixel directly without preprocessing. The benefit is strong tolerance to blur issues. However, all photometric errors of the pixels in the image are in account. Hence, higher image resolution becomes a burden on the CPU. The semi-dense SLAM is an alternative solution to reduce the computation. It aligns the small size of patch in the image and minimizes the photometric error in the selected area of image. The patches are applied to the region with sufficiently photometric gradient, such as corners, edges and high texture areas. LSD-SLAM [38], SVO [41], and DSO [37], are widely studied in the recent papers. The benchmark of feature-based and dense SLAM algorithms is shown in Fig.2.4.

	SVO	SVO (edgelets)	ORB-SLAM (no loop-closure)	ORB-SLAM (no loop, real-time)	DSO	DSO (real-time)	LSD-SLAM (no loop-closure)
Living Room 0	0.04	0.02	0.01	0.02	**0.01**	0.02	0.12
Living Room 1	0.07	0.07	0.02	0.03	**0.02**	0.03	0.05
Living Room 2	0.09	0.10	0.07	0.37	0.06	0.33	**0.03**
Living Room 3	×	0.07	**0.03**	0.07	0.03	0.06	0.12
Office Room 0	0.57	0.34	**0.20**	0.29	0.21	0.29	0.26
Office Room 1	×	0.28	0.89	0.60	0.83	0.64	**0.08**
Office Room 2	×	**0.14**	0.30	0.30	0.36	0.23	0.31
Office Room 3	0.08	**0.08**	0.64	0.46	0.64	0.46	0.56

Figure 2.4: Benchmark of two major visual odometry and SLAM, taken from [42]. The absolute translational errors (RMSE) is in meters.

3. Robotic Arm SLAM: In [57], Matthew *et al.* proposed the first paper regarding the robotic arm SLAM using a robotic arm. This articulated manipulator is mounted on a fixed base. A RGB-D camera is equipped on its EF. They used TSDF SLAM algorithm to estimate the EF poses and build a 3D map. In this paper, the forward kinematics is assumed invalid due to overload, backlash or disturbance.

2.1.2 Camera Model

In this thesis, the stereo camera is the only perceptional sensor that can measure the distances of the obstacles in front of the camera. Furthermore, the images from the camera can be sent to visual SLAM algorithms which can estimate the camera position as well. In order to achieve the precise measurements, the accurate camera parameters are required. Following subsections are the background of the camera models and parameters.

Pinhole Camera Model

The origin of a pixel matrix of a digital image is starting from the top-left corner. The X-axis and Y-axis are in the horizontal direction and vertical direction respectively. Based on the right-hand rule, the Z-axis is normal to the image plane and passing the center of camera lens. If there are n feature points in 3D space observed by a camera; these n points can be projected through the centre of the camera lens to the vision sensor. The perfect pinhole camera model is assumed that a projection point, q_1, on the image is laid on the

ray from the centre point of lens, X_i, to the point, Q_1, in 3D and its projection on the image of vision sensor, are laid on a ray and formed in Eq.2.8. A camera intrinsic matrix, $\mathbf{K}$ in 2.8, includes the focal lengths and the centre of the image. By triangulation in the temporal stereo analysis, the actual depth of a feature point along two projected rays from the camera posture, X_i, to the posture, X_j, can be computed if two extrinsic matrices in 2.8 are given (see Fig. 2.5).

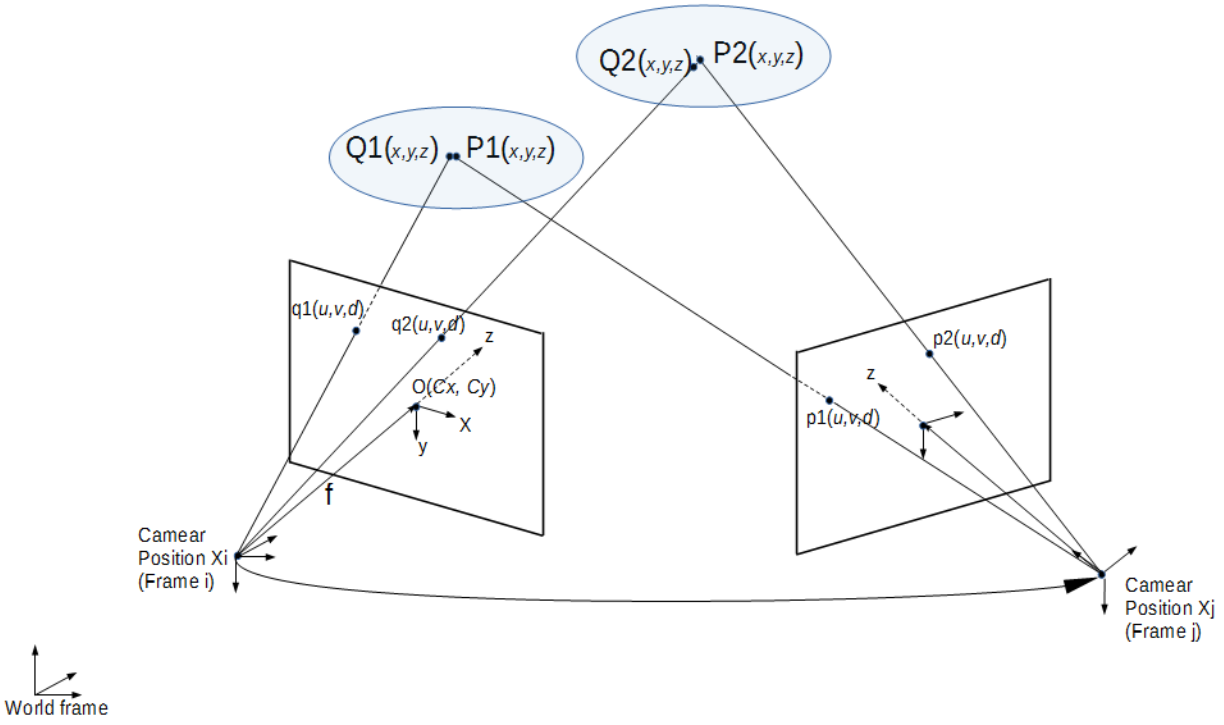

Figure 2.5: Observation of the same feature points from different camera poses, (z: forward, x: right, and y: down).

$$
\begin{aligned}
s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \mathbf{K}\,[\mathbf{R}|t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\
&= \underbrace{\underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{K:\ intrinsic\ \ matrix} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}}_{extrinsic\ \ matrix}}_{P:\ projection\ \ matrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\end{aligned}
\tag{2.8}
$$

where:

- extrinsic matrix is the transformation from the world frame to the camera frame,
- intrinsic matrix, K, is related to the lens parameters regarding 3D objects projected to the pixel matrix,
- $Q(X, Y, Z)$ are the coordinates of a 3D point in the world coordinate space,
- $q(u, v)$ are the coordinates of the distortion-free projection point in the pixel matrix,
- (c_x, c_y) is a principal point that is usually at the image centre,
- f_x,f_y are the focal lengths expressed in pixel units.

The lens curvature leads to the image distorted from the centre. In order to meet the pinhole model in Eq.2.8, following two models [10, 98, 100] are commonly utilized to rectify the distorted images before applying to visual SLAM algorithms:

- **Radial Distortion Model**
  A pixel in a distorted image is governed by following polynomial equation:

$$
\begin{aligned}
x_d &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + ...) \\
y_d &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + ...)
\end{aligned}
\tag{2.9}
$$

  where (x, y) is a pixel on the distance $r = \sqrt{x^2 + y^2}$ from the principal point of the correction image.

(2.9) can be simplified as

$$\begin{aligned}(x_d, y_d) &= (x, y) f(r, \mathbf{k}). \\ (x, y) &= f^{-1}(r, \mathbf{k})(x_d, y_d).\end{aligned} \tag{2.10}$$

where $\mathbf{k} = [k_1, k_2, k_3, ...]$.

- **Tangential Distortion Model**
  This is used for the lens not being parallel to the image sensor.

$$\begin{aligned}\mathrm{x}_d &= x + [2p_1xy + p_2(r^2 + 2x^2)] \\ \mathrm{y}_d &= y + [p_1(r^2 + 2y^2) + 2p_2xy]\end{aligned}$$

  where $\mathbf{p} = [p_1, p_2]$ are the parameters of this model.

Stereo Camera Model

The stereo camera is normally composed of two cameras with the same orientation. The camera sensors are aligned in the same plane apart with a fixed baseline along the x-axis of the camera frame. The projection matrix becomes the intrinsic matrix and camera translation:

$$P = \begin{bmatrix} f_x & 0 & c_x & t_x \\ 0 & f_y & c_y & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix}. \tag{2.11}$$

Its left 3x3 portion is the intrinsic matrix used for the rectified image. The fourth column $[t_x \;\; t_y \;\; t_z]^T$ is the translation from the position of the optical centre of a camera to the left camera's frame. Thus the fourth column of the projection matrix of the left camera can be simply set as $t_x = t_y = t_z = 0$. Then ideally the right camera can be simplified to $t_y = t_z = 0$ and $t_x = -f_x \times B$, where B is the baseline from the left camera to the right camera.

Given a 3D point $Q(X, Y, Z)$, the projection $q(u, v, d)$ of the point onto the rectified image is given by $[u \;\; v]^T = P[X \;\; Y \;\; Z \;\; 1]^T$.

Depth and inverse depth comparison

The inverse depth representation, $\frac{1}{\rho_0}$, is widely used in the mapping applications [25]. The advantage of this representation can be easy to extended to infinity, when ρ_0 is close to

zero. The only limitation is that the depth closed to zero cannot be accounted. The additive Gaussian noise of the depth measurement can be propagated to the new image by linearization of transformation function as shown in Fig. 2.6.

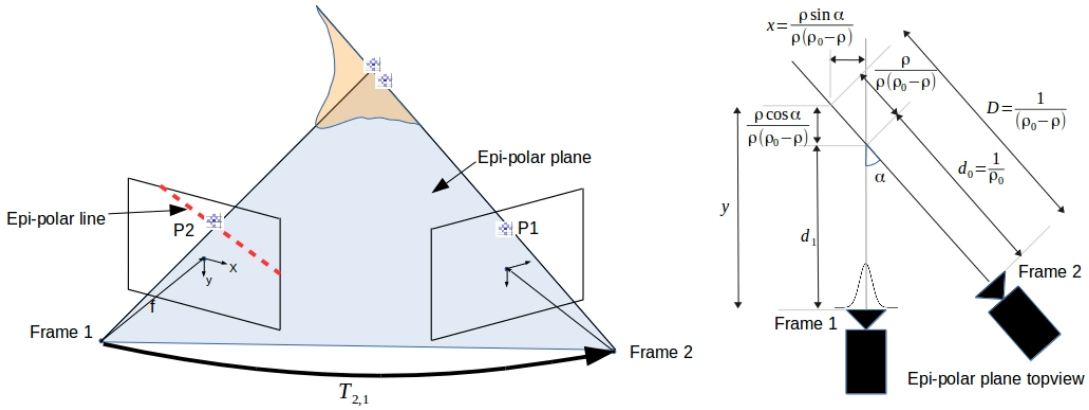

Figure 2.6: Depth and inverse depth coding[25].

2.1.3 Temporal Camera Position

The camera in our system is mounted on the end-effector of a robotic arm. The trajectory of camera can be represented by a 4×4 homogenous transformation matrix which contains a 3×3 rotation matrix and 3×1 translation vector in the rigid-body transformation $T_{j,i} \in SE(3)$ where the temporal footnotes i and j denote the transformation taken from the i-th frame to the j-th frame. The rotation matrix can be derived from Eular angles in order of roll-pitch-yaw in $SO(3)$ or unit quaternion [33].

$$\begin{bmatrix} \mathbf{X}_j \\ 1 \end{bmatrix} = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{X}_i \\ 1 \end{bmatrix}. \tag{2.12}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the measurement positions of the i-th frame and j-th frame, and the rotation matrix:

$$R(\phi,\theta,\psi) = R_r(\phi)R_p(\theta)R_y(\psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi c_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\theta s_\psi \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\theta c_\psi \end{bmatrix}. \tag{2.13}$$

or quaternion:

$$R(\mathbf{q}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 + 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \tag{2.14}$$

where $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$.

Reprojection Errors (2D points on images to 3D points in the space)

When the camera moves a short distance in the consecutive N images. The corresponding feature points on these images can be reprojected to 3D space using the projection matrix. Ideally, the static feature points from different images reprojected to 3D space will be aligned together. Alternatively, the 3D points can be transformed from the i-th frame to the j-th frame.

$$\mathbf{X}_j = T_{j,i}\mathbf{X}_i \tag{2.15}$$

where $\mathbf{X}_i$ are the 3D points projected from the i-th image frame, $\mathbf{X}_j$ are the 3D points projected from the j-th image frame, $T_{j,i} = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ O_{1\times3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}$, $T_{j,i}^{-1} = \begin{bmatrix} R_{3\times3}^T & -R^T t_{3\times1} \\ O_{1\times3} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}$. Thereby, the residual of the reprojection error of the observed common **static** points to 3D space can be formulated as the following cost function:

$$E = \sum_{i,j} \|\mathbf{X}_j^* - T_{j,i}\mathbf{X}_i^*\|^2. \tag{2.16}$$

$T_{j,i}$ comprises two factors: one is from the translation $\Delta t \in \mathbb{R}^3$, the other is from the rotation angles $\mathbf{r} \in \mathbb{R}^{\mathbf{3}}$.

$$\|\Delta t\| + \min(2\pi - \|\mathbf{r}\|, \|\mathbf{r}\|) \tag{2.17}$$

To solve the camera motion, $T_{j,i}$, through minimizing the reprojection error is called 'Perspective-n-Point' [12].

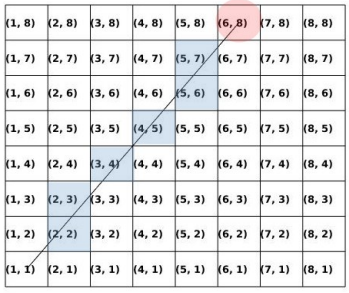

Figure 2.7: Bresenham's ray tracing.

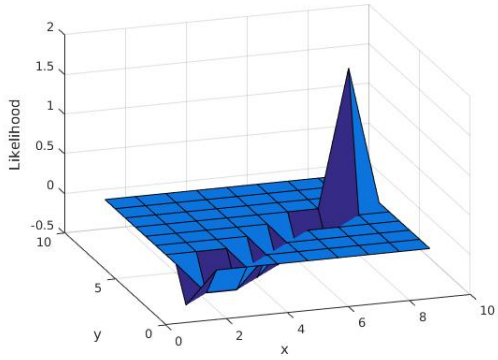

Figure 2.8: Likelihood of occupancy grid cells.

2.2 Map Representation

There are two types of maps commonly used in the robotics field:

1. **Point cloud map** is a collection of the measurements defined by a given coordinates, a set of 3D points, to represent non-transparent objects.

2. **Volumetric map** is defined by 2D grid cells or 3D cubic volumes with a given metric unit and probability. When a measurement point falls in a 2D grid cell, the likelihood of the cell occupied will increase.

   Bresenham's line tracing algorithm (Algorithm 1) can be used to determine the grid cells is occupied or not. The algorithm searches the discrete cells passed by a straight line from a robot's sensor to the measured point. The cells passed by this straight line are assumed to be free, and then the algorithm decreases the likelihood of these cells. On the contrary, the probability of an observed point is increased. Thus, its likelihood of occupancy is increased. Once a cell's likelihood of occupancy reaches a predefined threshold, this cell is considered occupied by an obstacle. For example, in Fig. 2.7, the robot's sensor is on the cell, $(1, 1)$, and the measured point is on the cell, $(6, 8)$. The likelihood of occupied is as shown in the Fig. 2.8. The algorithm can be extended to 3D mapping in the same scheme.

In this thesis, the map is exploited for mobile robot navigation and path planning with collision avoidance. The volumetric map is used to determine obstacles. In [53], Hornung

et al. proposed the Octo Map, which is an efficient probabilistic 3D volumetric mapping represented by Octree. Each node, a cubic volume called a voxel, in an Octree tree has eight descendant nodes. In Octree, the 3D space is recursively subdivided into eight octants until reaching the demanded metric size. This approach has the following advantages:

1. Full 3D model: The map is able to model arbitrary environments without prior assumptions. The occupied area represents the obstacles. Thereby, to compute the distance between the robot geometry and occupied space is essential for collision avoidance detection.

2. Updatable: Each cubic region contains all the prior information. It is possible to add new information or sensor readings at any time. Furthermore, multiple robots are able to contribute to the same map and a previously recorded map is expendable when the robot can be localized itself in the map.

3. Flexible: The map can be dynamically expanded as needed.

The depth of the octree impacts on the resolution and memory consumption. For instance, if the depth of the tree is 16 and each cubic is $1[cm^2]$, the OC tree represents a volume of $2^{16}0.01[m^3] = 655.36[m^3]$. $2^{32}0.01[m^3] = 42949672.96[m^3]$.

Each class node contains a probability value and a pointer to the eight child nodes. Each occupancy probability of volume is initialized to the uniform probability of $P(n) = 0.5$, as shown in Fig. 2.8.

Large-Scale Mapping

The number of measurements will keep growing when a robot explores a large-scale area. That becomes the burden of the computer resources on the mobile robot. The scale of the complexity is $\mathcal{O}(n^2)$ time, where n is the sum of poses and measurements [1]. Thereby, to manage the memory becomes another critical issue.

In [75], Reid *et al.* proposed a multi-robot SLAM algorithm to build the map in the large-scale area. The map building was implemented by a centralized ground control station (GCS) to fuse the submaps from different unmanned ground vehicles (UGVs). In [61], Labbe *et al.* exploited multi-session approach to solve the memory shortage issue

[1] In computer science, big $\mathcal{O}$ notation is used to classify algorithms according to how their running time or space requirements grow as the input size grows. [2]

Algorithm 1 Bresenham's ray tracing algorithm in 2D map

See the correspondent points in Fig. 2.1
function RAY TRACING(p_{ix},p_{iy}, q_{ix},q_{iy}, T):
    point(q_{ix}, q_{iy}) is falling in the cell n
    $p(n|z_{1:T}) = \left[1 + \frac{1-P(n|z_T)}{P(n|z_T)} \frac{1-P(n|z_{1:T-1})}{P(n|z_{1:T-1})} \frac{P(n)}{1-P(n)}\right]^{-1}$
    $\alpha = \text{logit}(p) = \log\left(\frac{p}{1-p}\right)$
    $p = \text{logit}^{-1}(\alpha) = \frac{1}{1+\exp(-\alpha)}$
    $L(n|z_{1:T}) = L(n|z_{1:T-1}) + L(n|z_T)$
    if $L(n) > l_{occupied}$ **then**
        the cell is an obstacle
    end if
    if $L(n) < l_{free}$ **then**
        the cell is free
    end if
    $dx = q_{ix} - p_{ix}$
    $dy = q_{iy} - p_{iy}$
    $D = dy - dx$
    $y = q_{iy}$
    for x from p_{ix} to $q_{ix} - 1$ **do**
        set cell(x,y) free
        if $D \geq 0$ **then**
            $y = y + 1$
            $D = D - dx$
        end if
        $D = D + dy$
    end for
end function

where T is the time. For example, if the likelihood of occupied is $l_{occupied} = 0.85$ and the likelihood of free is $l_{free} = 0.4$, the corresponding to probabilities of $p_{occupied} = \log^{-1}(0.85) = \frac{1}{1+\exp(-0.85)} = 0.700567142$ and $p_{occupied} = \text{logit}^{-1}(0.4) = \frac{1}{1+\exp(0.4)} = 0.40131234$ for occupied and free volumes .

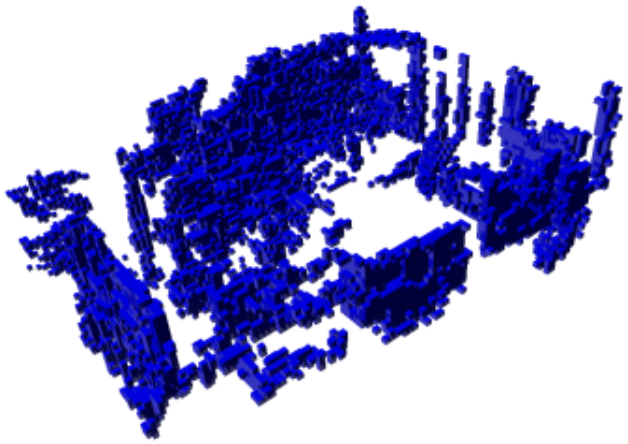

Figure 2.9: This is an example of Octomap built from the point clouds that are captured using Tara stereo camera. The room dimension is 5.8m (L) x 4.1m (W) x 2.3m (H).

with large-scale mapping. The zone of the map around the robot saves in the short-term memory. If a given zone is further away than a predefined distance from current robot position, the zone will be moved to long-term memory, e.g. local hard drive or remote GCS, for later use. Thus, remote GCS can stitch the maps from multiple robots in the field.

2.3 Place Recognition and Loop Closure Detection

When a vision-based robot travels to where it had been before and recognizes the place,"Loop Closure Detection" takes place. For example, a mobile robot at the centre of a room changes its heading angle (yaw) from starting point 0 degree to 360 degrees, it will face the same scene again. If the robot recognizes that this scene is the same as the one it observed at 0 degree, the first node of the pose-graph is equal to the last node of it. This means the no more new node is required to be added and these nodes become a closed loop. Through the iterative optimization algorithms, the first node and last node may be aligned more closely and corrected the accumulated drift errors from node to node. All nodes inside the loop will be refined after the iterative optimization.

For larger loops, a robot needs to search a large number of images in the memory to find a similar scene. So the loop closure detection is another challenging topic in the SLAM problem. Bag of Words [46] is an efficient algorithm to store and search the feature points in the historical images, like the an open source C++ library for indexing and converting images into a bag-of-word representation (DBoW) [3].

A place recognition algorithm can be applied to solve the recovery issue. For example, if a visual SLAM algorithm is lost tracking or a robot power is reset, a robot can use a place recognition algorithm to localize itself on the existing map.

The detected loop comprises nodes which have robot state and measurements in the following form.

$$\mathbf{P} = \{p_1, p_2, \ldots, p_N\} \in \mathbf{X}. \tag{2.18}$$

$$\mathbf{Q} = \{q_1, q_2, \ldots, q_M\} \in \mathbf{Y}. \tag{2.19}$$

$$\forall i, p_i = Rq_i + t. \tag{2.20}$$

where R is rotation matrix, t is the translation vector.

The map can be refined and the drift error can be minimized to minimize.

$$\min_{R,t} \sum_{i=1}^{N} \|p_i - (Rq_i + t)\|_2 \tag{2.21}$$

In graph-based SLAM[48], the trajectory, states and parameters of robot moved in a period of time are described by nodes $\mathbf{x} = \{x_1, \ldots, x_N\}$ on the graph. The measurement done by odometry and perception sensors are represented by $\mathbf{y} = \{y_1, \ldots, y_k\}$. See Fig. 2.10. There are several methods commonly used in the graph optimization to solve (2.21), such as Gauss-Newton method, Levenberg-Marquardt, gradient descent on a manifold to find the adjacent links on the graph. g2o [59], GTSAM [32] and Google Ceres-Solver [17] are popular open source C++ algorithms to solve non-linear least square problems in real-time.

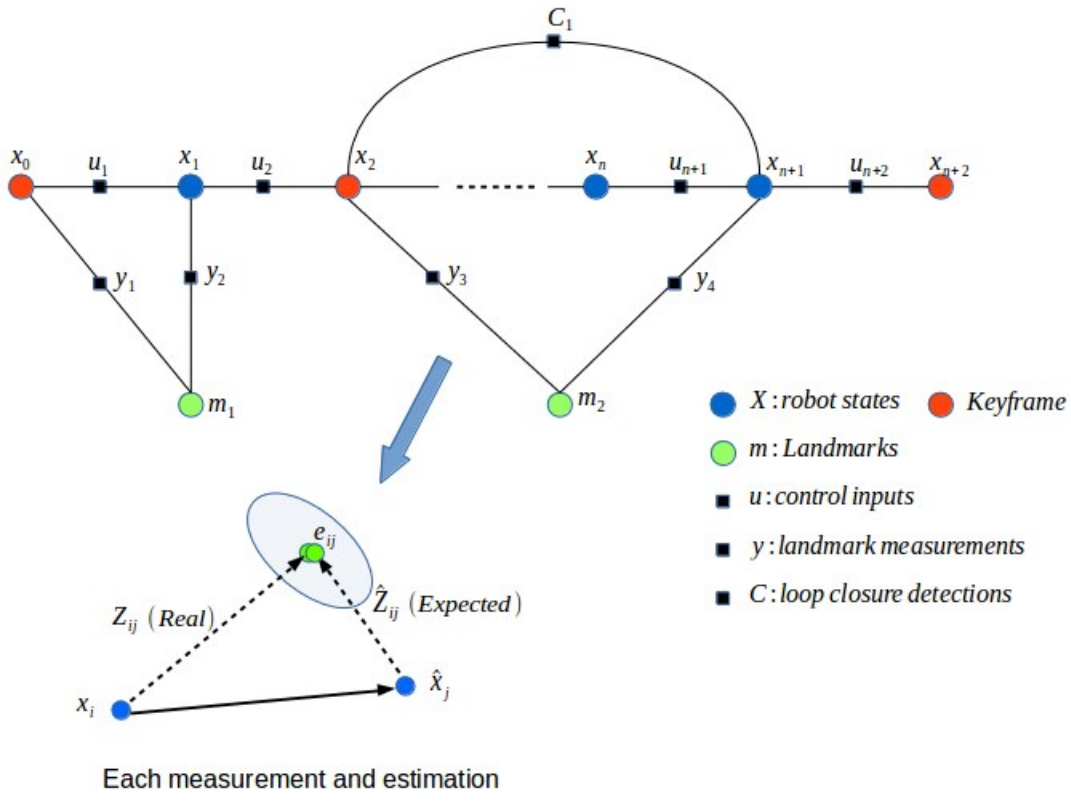

Figure 2.10: Factor graph-based SLAM is formulated by parameter nodes and measurements. When the robot travels to the location where it passed before, the loop closure detection will bridge the nodes and minimize the error residuals.

2.3.1 Graph-Based Optimization

Graph-Based SLAM is formulated by the nodes of a robot state and each node is connected by the control input in the sequence of time steps as shown in Fig.2.10. Each node contains the timestamped states and measurements taken at that states, such as the feature points extracted from the 2D image. The feature points can be reprojected to 3D space with a metric depth using a stereo camera or a scaled depth in monocular camera.

The log-likelihood l_{ij} of measurements, residual of reprojecting feature points from the i-th frame and the j-th frame to 3D space, can be formulated in the following form [48]

$$\begin{aligned} \mathrm{l}_{ij} \propto \mathbf{E_{ij}} &= [\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j)]^T \mathbf{\Omega}_{ij} [\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j)] \\ &= \quad \mathbf{e}_{ij}^T(\mathbf{x}_i, \mathbf{x}_j) \mathbf{\Omega}_{ij} \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ &= \quad \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij} \end{aligned}$$

where $\mathbf{\Omega}_{ij}$ is w.r.t. the zero mean and information matrix of the measurements taken from the i-th frame and the j-th frame.

The SLAM algorithms minimize the sequence of residuals, $\mathbf{e}_{ij}$, of the measurements to derive the approximated camera poses.

$$\begin{aligned} \mathbf{x}^* &= \arg\min_{\mathbf{x}} \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{E}_{ij} \\ &= \arg\min_{\mathbf{x}} \ \mathbf{E}(\mathbf{x}) \end{aligned} \tag{2.22}$$

where $\mathcal{C} = \{\langle j_1, i_1 \rangle, ..., \langle j_n, i_n \rangle\}$ is set of pairs of nodes $\in [1..n]$for which $\hat{\mathbf{z}}$ exists.

Based on the current solution $\mathbf{x}^*$, we let $\breve{\mathbf{x}} = \mathbf{x}^*$ to compute the approximation of next solution $\mathbf{x}^* = \breve{\mathbf{x}} + \mathbf{\Delta x}^*$.

$$\begin{aligned} \mathbf{E}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x}) &= \mathbf{e}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x})^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}(\breve{\mathbf{x}} + \mathbf{\Delta x}) \\ &\simeq (\mathbf{e}_{ij} + \mathbf{J}_{ij} \mathbf{\Delta x})^T \mathbf{\Omega}_{ij} (\mathbf{e}_{ij} + \mathbf{J}_{ij} \mathbf{\Delta x}) \\ &= \underbrace{\mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}}_{\mathrm{c}_{ij}} + 2 \underbrace{\mathrm{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{b}_{ij}} \mathbf{\Delta x} + \mathbf{\Delta x}^T \underbrace{\mathbf{J}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}}_{\mathbf{H}_{ij}} \mathbf{\Delta x} \\ &= \mathrm{c}_{ij} + 2\mathbf{b}_{ij} \mathbf{\Delta x} + \mathbf{\Delta x}^T \mathbf{H}_{ij} \mathbf{\Delta x} \end{aligned} \tag{2.23}$$

Eq. (2.23) is the least squares optimization problem.

$$\mathbf{H}\boldsymbol{\Delta}\mathbf{x}^{*} = -\mathbf{b} \tag{2.24}$$

where

$$\mathbf{H}_{ij} = \begin{bmatrix} \ddots & & & & \\ & \mathbf{A}_{ij}^{T}\boldsymbol{\Omega}_{ij}\mathbf{A}_{ij} & \cdots & \mathbf{A}_{ij}^{T}\boldsymbol{\Omega}_{ij}\mathbf{B}_{ij} & \\ & \vdots & \ddots & \vdots & \\ & \mathbf{B}_{ij}^{T}\boldsymbol{\Omega}_{ij}\mathbf{A}_{ij} & \cdots & \mathbf{B}_{ij}^{T}\boldsymbol{\Omega}_{ij}\mathbf{B}_{ij} & \\ & & & & \ddots \end{bmatrix}, \tag{2.25}$$

$$\mathbf{b}_{ij} = \begin{bmatrix} \vdots \\ \mathbf{A}_{ij}^{T}\boldsymbol{\Omega}_{ij}\mathbf{e}_{ij} \\ \vdots \\ \mathbf{B}_{ij}^{T}\boldsymbol{\Omega}_{ij}\mathbf{e}_{ij} \\ \vdots \end{bmatrix}, \tag{2.26}$$

and $\mathbf{A}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_i}$ and $\mathbf{B}_{ij} = \frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}_j}$. In [69, 74], the optimization solver is based on the library proposed by Rainer Kummerle [59].

2.4 Path Planning

The main idea of path planning for visual servoing is to plan and generate a feasible trajectory while satisfying for the robot's kinematic constraints, and then to guide the robot to follow this collision-free path. A planned trajectory can be decomposed into discrete segments and saved to a linked list beginning with the current robot position and ending up with a goal.

2D Planner

Wavefront algorithm is a methodology to find the shortest path from one point to the other in a map [45]. In the occupancy grid map, we use in this thesis, is a 2D plane segmented by equal spacing grid cells. The size of each grid cell is represented as an absolute scale on the map. From the starting position, a grid cell in Fig. 2.11, is set to zero cost/distance. The

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Figure 2.11: The cost increment mask of 8-connected grid graph starts at zero cost. It comprises the current position cost with the mask and applies to the cell not being calculated, and then keeps exploring until the wave mask reaches the goal.

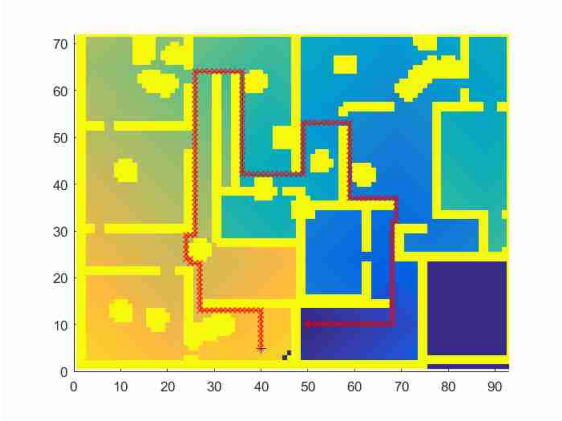

Figure 2.12: Wavefront path planning algorithm.

adjacent cells are set at an incremented cost value based on distance. Discrete grid cells around a starting point have the same cost. Therefore, the cost of each cell is represented as the distance from the starting point. After all reachable cells on the map are computed, the breadth first search algorithm can be applied to find the shortest path to the goal reversely due to monotonic. However, to compute entire grid cells on a map is very expensive and the complexity is proportional to the number, n, of grid cells, $\mathcal{O}(n^2)$. In order to, leverage the computation, the modified method is proposed in the chapter 5 to reduce the computation the issue.

Path Following

In [70], Derek R Nelson *et al.* proposed a method based on the vector fields, which are exploited to generate desired inputs to the control loop. Two path following algorithms are presented in this paper. One is for straight-line paths and the other is circular arcs. This algorithm can be used for cruise control effectively from one waypoint to another. This algorithm is integrated into our control system.

Collision Avoidance

In [72], Jia Pan presented a library to integrate and perform collision checking. This library can take articulated models to create a set of queries to perform penetration depth estimation, discrete collision detection and continuous collision detection with a unified

interface. Moreover, it can perform the probabilistic collision checking. In [62], Leeper proposed a Collision-Free Arm Teleoperation (CAT) method using sequential quadratic programming and a bi-directional sampling-based planner RRT in OMPL to plan the EF trajectory. Meanwhile, it computes a collision-free trajectory by the Flexible Collision Library (FCL) [72].

Chapter 3

The Surveillance Mapping Task and the Proposed Robotic System

In this chapter, the kinematic models of the omni-wheeled mobile platform and an equipped robotic arm are presented. The stereo camera is rigidly mounted on the EF of the robotic arm to allow the camera to be travelled across a gap where the mobile platform cannot traverse to maximize the mapping and searching area. Accordingly, to satisfy this demand requires driving the robotic arm base close to a gap that permits the work envelope of the robotic arm to cover the observed gap. So the omni-wheeled mobile platform which can move the car body in the lateral direction without changing its heading direction, without nonholonomic motion constraints, is chosen in our system. Next, all the major sensors in the mobile platform and the robotic arm are presented. At the end of this chapter, a proposed methodology successfully adjusts the bias in the Denavit-Hartenberg parameters (D-H parameters) which are not provided by the producer of 7Bot©.

3.1 General Structure of the Proposed System

The proposed surveillance robot system is composed of four modules: Mechanical robot system, perception sensors, motion control system, path planning and SLAM scheme. The robot hardware comprises two major components: omni-wheeled mobile platform and an attached 6-Degrees of Freedom (DOF) serial robotic arm. The only perception sensor which measures the distances of surrounding obstacles is a stereo camera in our system.

In the system block diagram Fig. 3.1, the main controller is the embedded PC in the middle block. The PC has Intel© Core^{TM} i5-3337U CPU @ 1.80GHz, 16GB RAM and

60 GB Solid State hard drive. It is running Ubuntu 14.04 and ROS Indigo packages. Two measurement sensors, IMU and stereo camera, are connected to embedded PC via USB interface. The four individual omni-wheels are driven by four gear motors controlled by an Arduino Mega 2560 board. Each gear motor has an optical rotary encoder mounted on the end of the motor shaft. The Arduino Mega 2560 reads the encoder pulse counts through the I^2C interface and outputs control signal to the motor drive through the built-in PWM pins. The serial robotic arm, 7Bot, is controlled by an Arduino Duo. The control scheme is similar. The controller reads joint angles through six encoders on the revolute joints and outputs voltage signal to PWM pins. The remote computer, Intel© Core^{TM} i7-4710MQ CPU @ 2.50GHz × 8, 16GB RAM and 120 GB Solid State hard drive, is used for data logging and path planning. It is running Ubuntu 16.04 and ROS Kinetic packages.

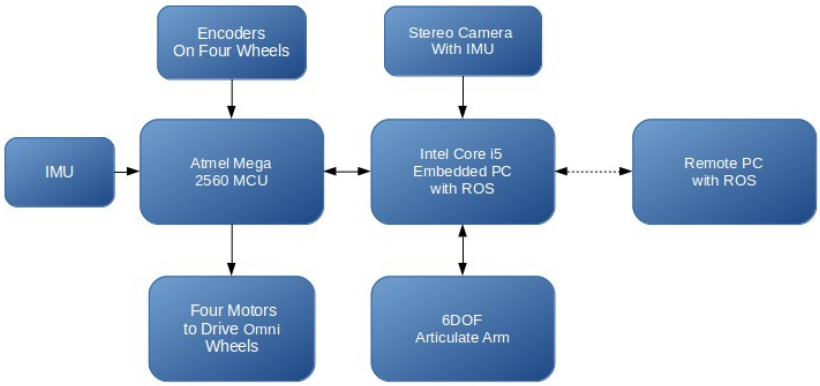

Figure 3.1: Integration layout of the sensors and processors of the proposed surveillance robot system.

3.2 Mechanical Robot System

The proposed mechanical robot system is composed of an omni-wheel mobile platform and a 6-DOF serial robot manipulator equipped on the front edge of the mobile platform, as shown in Fig. 3.3. The robot manipulator holds the stereo camera and is utilized to control the camera poses. Next, we present the details and kinematic models of the robot manipulator and the mobile platform, which are used in the motion control and visual SLAM of our prototype robot.

3.2.1 Robot Manipulator

7Bot© is a low cost six DOF articulated robot manipulator. The six revolute joints are driven by six servo motors which have been retrofit and controlled by a PID implemented on an Arduino DUE board. The command angle of each joint is sent from the mini PC through the ROS serial interface. The original design of its end-effector is equipped with a gripper. In order to mount our stereo camera, the gripper is replaced by a bracket instead.

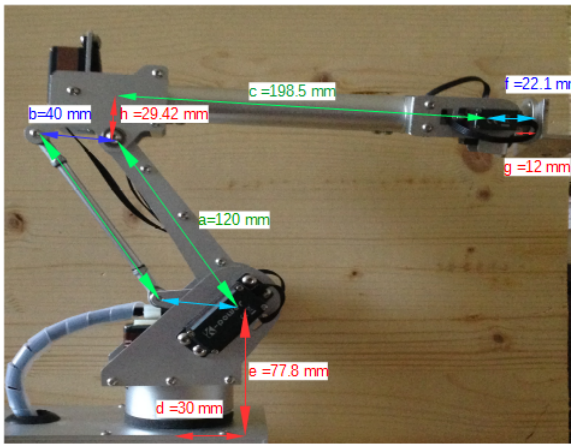

Figure 3.2: The robot arm geometry.

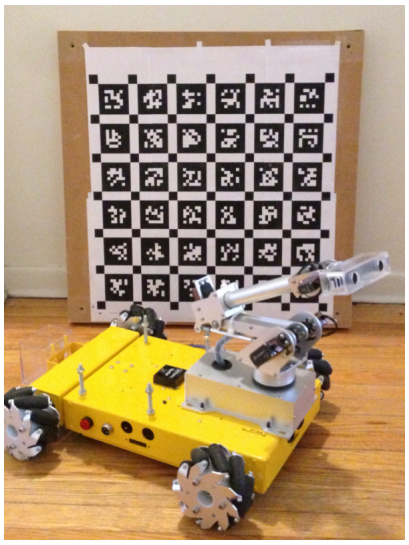

Figure 3.3: Robotic arm, mobile platform and april tag checkerboard.

Forward Kinematics

Based on the D-H table 3.1 from the measurement by hand, the homogeneous transformation matrix, $T_{i,i+1}$, from a i-th joint to the $(i+1)$-th joint through the following sequence of individual transformations can be derived. The joint rotation angle θ_i $(i = 1, 2, ..., 6)$ is measured by a joint encoder about the z-axis associated with the given joint using the right-hand rule[27].

Substituting the D-H parameters in Table 3.1 into the generic homogeneous transfor-

Link i	d_i(mm)	a_i(mm)	α_i	$\theta_i(t)$
1	77.8	30	$\frac{\pi}{2}$	$\theta_1(t)$
2	0	120	0	$\theta_2(t)$
2_v	0	0	0	$\theta_{2v}(t)$
3	0	29.42	$\frac{\pi}{2}$	$\theta_3(t)$
4	198.5	0	$\frac{\pi}{2}$	$\theta_4(t)$
5	0	0	$-\frac{\pi}{2}$	$\theta_5(t)$
6	0	0	0	$\theta_6(t)$

Table 3.1: D-H table of the articulated robot arm

mation matrix formula

$$T_{i,i+1} = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_i \\ sin\theta_i cos\alpha_i & cos\theta_i cos\alpha_i & -sin\alpha_i & -d_i sin\alpha_i \\ sin\theta_i sin\alpha_i & cos\theta_i sin\alpha_i & cos\alpha_i & d_i cos\alpha_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

the forward kinematics of each link is derived as follows:

$$T_{0,6} = T_{0,1}T_{1,2}T_{2,2_v}T_{2_v,3}T_{3,4}T_{4,5}T_{5,6} \tag{3.2}$$

Inverse Kinematics

All the joints of 7Bot robot arm are on the same plane. Thus, we can use the closed-form to describe the inverse kinematic solution and find the desired angle of each joint individually. The wrist transformation from joint 4 to joint 6 will not be affected by joint 1 to joint 3. Therefore, we use the geometric method to decompose entire inverse kinematics into the inverse position kinematics and the inverse orientation kinematics instead.

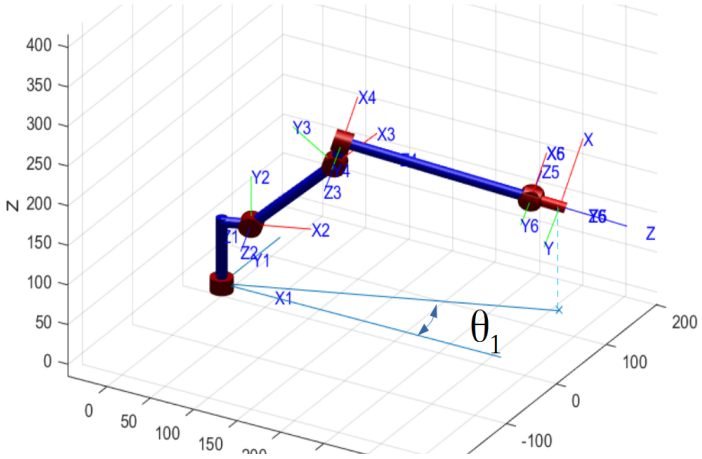

Figure 3.4: Due to joint 2 to 6 are on the same plane, the end-effector position projected to the robot base plane is the function of the first joint angle θ_1.

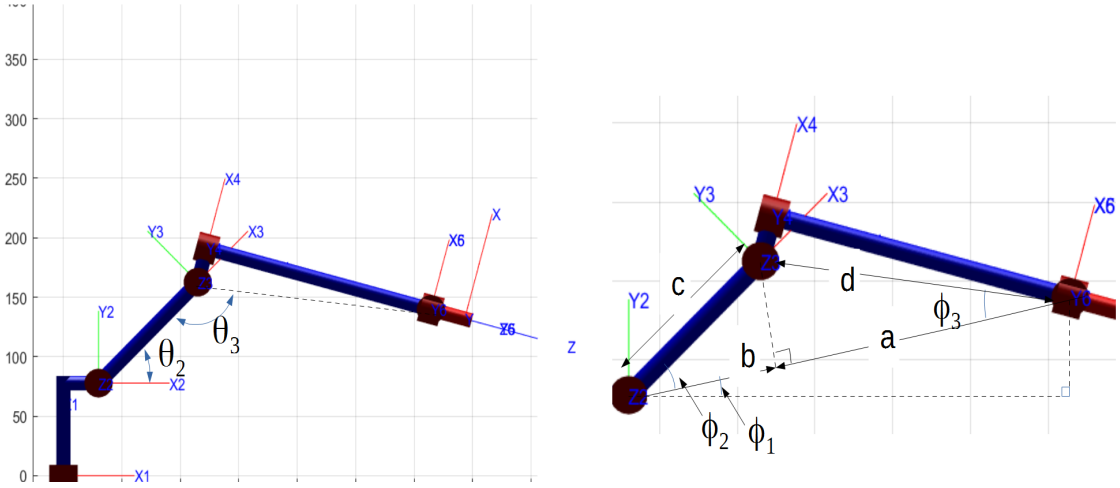

Figure 3.5: Joint 2, 3, and 5, on the same plane can be formed a triangle.

$$\theta_1 = Atan({}^0P_5^y, {}^0P_5^x), \tag{3.3}$$

where ${}^0P_5^y$ and ${}^0P_5^x$ is the point where the 5th joint is projected to on the arm base plane with respect to the joint 1 as the origin.

$$\theta_2 = \phi_1 + \phi_2, \tag{3.4}$$

$$\theta_3 = \pi - (\phi_2 + \phi_3), \tag{3.5}$$

where $L = a + b = \sqrt{({}^2P_5^x)^2 + ({}^2P_5^y)^2}$, ${}^2P_5^x$ and ${}^2P_5^z$ are the translation vector from joint 2 to the joint 5 w.r.t. the frame of joint 2 , $\phi_1 = Atan2({}^2P_5^z, {}^2P_5^x)$, $\phi_2 = Acos(b/c)$, $\phi_2 = Acos(a/c)$, $c = a_2$, $d = \sqrt{(a_3)^2 + (d_4)^2}$; a_2, a_3 and d_4 are parameters in the D-H table. See Fig. 3.5.

$$c^2 - d^2 = a^2 - b^2 = const., \tag{3.6}$$

where $a = \frac{L}{2} + \frac{c^2 - d^2}{2L}$.

The role of the robotic arm is to increase the visual observation area. Thereby, the dynamic control which needs to consider the various payload and speed variations is not a focus of this thesis.

3.2.2 Mobile Platform

In Figure 3.9, the mobile platform is actuated by four omni-wheels which are composed of several rollers around the circumference of the wheel hub. When the wheel rotates and the roller contacts on the ground surface, the axial direction of the roll gets the reaction. On the contrary, the opposite direction is free to move. The type of wheeled mobile platforms is able to move in any direction without changing the heading direction. This allows the mobile platform to travel in narrow paths without collision.

Nexus© Mobile Base in Fig. 3.6 is the platform used in the experiment. The geometry of this platform is compact. So it is suitable to move in narrow paths or tunnels. The base comprises four individual motors equipped with omni-wheels. Each motor has an upgrade encoder with 300 pulses per revolution. The wheel hubs and the motors are coupled by reduction gearboxes with 1:64 ratio to generate more torque. Four DC brush motors are equipped and controlled by Arduino Mega 2560 board. Four individual PID controllers are implemented on an Arduino Mega 2560 board in 20Hz update rate. The teleoperation is done by keyboard or the commercial joystick via the Bluetooth interface, as shown in Fig. 3.6.

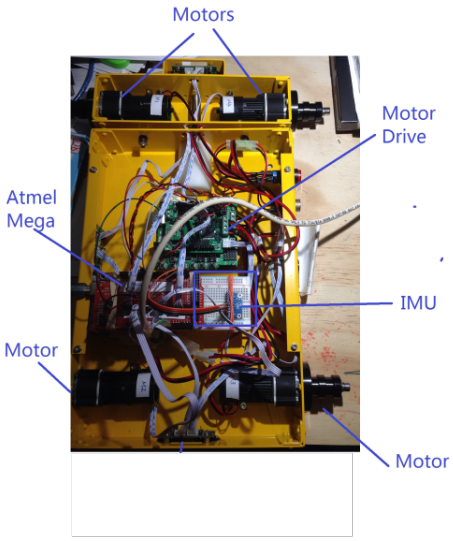

Figure 3.6: Mobile platform.

Figure 3.7: SONY©PS3 joystick for the teleoperation.

Posture Kinematic Model

In this subsection, the posture kinematic model of the omni-wheeled platform [54] is introduced. The compact form [79] is adapted to describe the pure rolling constraints for the omni-wheels.

The position and heading of the geometric centre of the wheeled mobile robot is denoted by

$$\xi(t) = (x_I(t) \quad y_I(t) \quad \theta_I(t))^T. \tag{3.7}$$

where $(x_I, y_I) \in \mathbb{R}$ are the coordinates of the centre of mass(C.M.) with respect to the inertial frame in X and Y directions. $\theta_I \in (-\pi, \pi]$ is the heading angle about z-axis in Fig. 3.8.

Following is the general posture kinematic model for the omni-wheeled platform,

$$J_{1_{sw}} R(\theta)\dot{\xi} + J_2\dot{\varphi} = 0. \tag{3.8}$$

where $\dot{\varphi} = \begin{bmatrix}\dot{\varphi}_1 & \dot{\varphi}_2 & \dot{\varphi}_3 & \dot{\varphi}_4\end{bmatrix}^T$ is the vector of wheel angles, $J_{1_{sw}} \in \mathbb{R}^{4\times 3}$, $J_2 = diag[r_w cos\delta_1, \quad r_w cos\delta_2, \quad r_w cos\delta_3, \quad r_w cos\delta_4]$, r_w is the uniform radius of each omni-wheel,

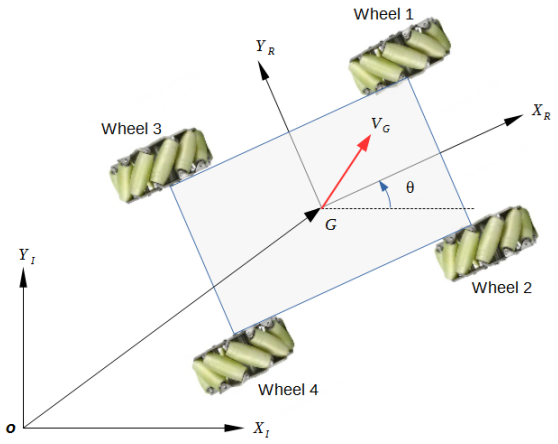

Figure 3.8: The posture definition of the mobile platform on the inertial frame.

, δ_i $(i = 1, 2, 3, 4)$ is the angle between the small rollers' axial and the wheel plane in Fig 3.9.

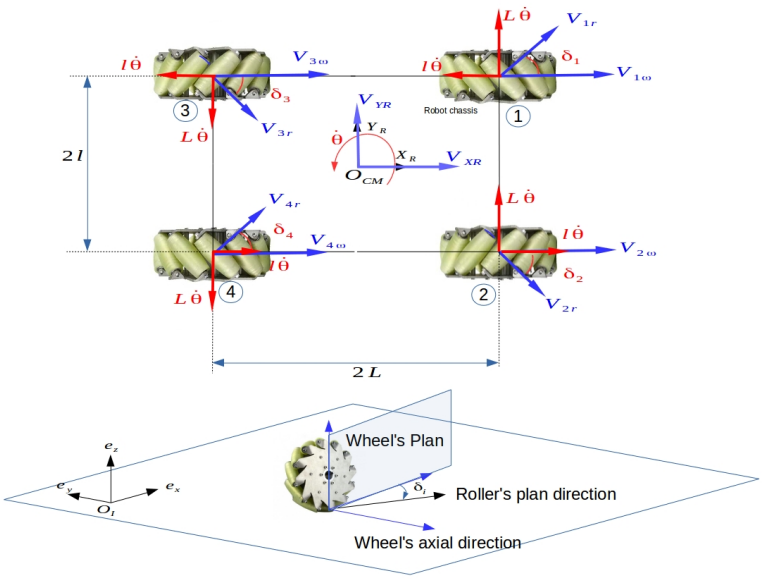

Figure 3.9: Roller orientation of each omni-wheel.

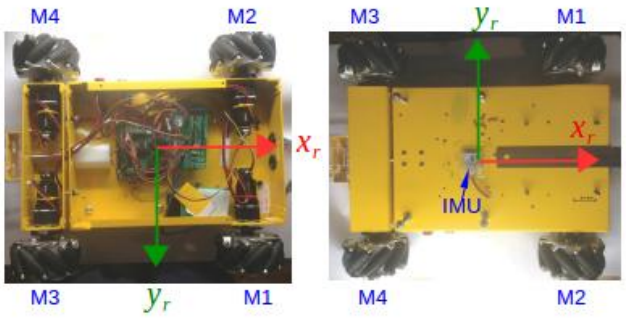

Figure 3.10: Left: bottom-view of the mobile base; Right: top-view of the mobile base.

Following are the steps to derive $J_{1_{sw}}$ [96]:

$$V_{1X_R} = V_{1w} + V_{1r}cos(\delta_1) \quad , \quad V_{1Y_R} = V_{1r}sin(\delta_1). \tag{3.9}$$

$$V_{2X_R} = V_{2w} + V_{2r}cos(\delta_2) \quad , \quad V_{2Y_R} = -V_{2r}sin(\delta_2). \tag{3.10}$$

$$V_{3X_R} = V_{3w} + V_{3r}cos(\delta_3) \quad , \quad V_{3Y_R} = -V_{3r}sin(\delta_3). \tag{3.11}$$

$$V_{4X_R} = V_{4w} + V_{4r}cos(\delta_4) \quad , \quad V_{4Y_R} = V_{4r}sin(\delta_4). \tag{3.12}$$

$$V_{1X_R} = V_{X_R} - l\dot{\theta} \quad , \quad V_{1Y_R} = V_{Y_R} + L\dot{\theta}. \tag{3.13}$$

$$V_{2X_R} = V_{X_R} + l\dot{\theta} \quad , \quad V_{2Y_R} = V_{Y_R} + L\dot{\theta}. \tag{3.14}$$

$$V_{3X_R} = V_{X_R} - l\dot{\theta} \quad , \quad V_{3Y_R} = V_{Y_R} - L\dot{\theta}. \tag{3.15}$$

$$V_{4X_R} = V_{X_R} + l\dot{\theta} \quad , \quad V_{4Y_R} = V_{Y_R} - L\dot{\theta}. \tag{3.16}$$

where $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 45°$. By comparison with Eq.3.9-3.16, the following equations are derived.

$$V_{1w} = r_w\dot{\varphi_1} = V_X - V_Y - (L+l)\dot{\theta}. \tag{3.17}$$

$$V_{2w} = r_w\dot{\varphi_2} = V_X + V_Y + (L+l)\dot{\theta}. \tag{3.18}$$

$$V_{3w} = r_w\dot{\varphi_3} = V_X + V_Y - (L+l)\dot{\theta}. \tag{3.19}$$

$$V_{4w} = r_w\dot{\varphi_4} = V_X - V_Y + (L+l)\dot{\theta}. \tag{3.20}$$

Combining (3.8) and (3.17) - (3.20) into matrix form (3.21).

$$J_{1_{sw}} R(\theta)\dot{\xi} = -J_2\dot{\varphi}. \tag{3.21}$$

In our experiment $L = l$, hence, we have

$$J_{1_{sw}} = \begin{bmatrix} 1 & -1 & -(L+l) \\ 1 & 1 & (L+l) \\ 1 & 1 & -(L+l) \\ 1 & -1 & (L+l) \end{bmatrix} = \begin{bmatrix} 1 & -1 & -2l \\ 1 & 1 & 2l \\ 1 & 1 & -2l \\ 1 & -1 & 2l \end{bmatrix}, \tag{3.22}$$

$$J_2 = \begin{bmatrix} r_w cos(\delta_1) & 0 & 0 & 0 \\ 0 & r_w cos(\delta_2) & 0 & 0 \\ 0 & 0 & r_w cos(\delta_3) & 0 \\ 0 & 0 & 0 & r_w cos(\delta_4) \end{bmatrix}. \tag{3.23}$$

where $\dot{\xi} = \begin{bmatrix} \dot{x_I} & \dot{y_I} & \dot{\theta} \end{bmatrix}^T$, .

The platform velocity can be obtained from the wheel's angular speed by a pseudo inverse

$$\dot{\xi} = -R(\theta)^T J_{1_{sw}}^{\dagger} J_2\dot{\varphi}, \tag{3.24}$$

$$J_{1_{sw}}^{\dagger} = (J_{1_{sw}}^T J_{1_{sw}})^{-1} J_{1_{sw}}^T = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{2l} & \frac{1}{2l} & -\frac{1}{2l} & \frac{1}{2l} \end{bmatrix}. \tag{3.25}$$

The nonlinear kinematic model (3.24) can be put in compact form

$$\dot{\xi}(t) = g(\varphi_t, \xi_t) = R(\theta_t)^T \Sigma \varphi_t + \epsilon_t, \qquad \epsilon_t \sim N(0, R_t), \tag{3.26}$$

$$\Sigma = -J_{1_{sw}}^{\dagger} J_2 = \frac{r_w}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{2l} & \frac{1}{2l} & -\frac{1}{2l} & \frac{1}{2l} \end{bmatrix}. \tag{3.27}$$

Linearizing the non-linear equation (3.26) w.r.t. the mean of robot states, μ_{t-1}, of the prior state by the first order Taylor series expansion.

$$\begin{aligned} g(\varphi_t, \xi_{t-1}) &\approx g(\varphi_t, \mu_{t-1}) + \frac{\partial}{\partial \xi_{t-1}} g(\varphi_t, \xi_{t-1})|_{\xi_{t-1}=\mu_{t-1}} (\xi_{t-1} - \mu_{t-1}) \\ &= g(\varphi_t, \mu_{t-1}) + G_t(\xi_{t-1} - \mu_{t-1}). \end{aligned} \tag{3.28}$$

The mobile platform motion system contains two sensors with different rates. One is IMU which is able to measure bearing in 50Hz. The other one is the stereo camera with different visual odometries to measure the position and bearing.

The measurement model is

$$y_t = C_t \xi_t + \sigma_t, \qquad \sigma_t \sim N(0, Q_t). \tag{3.29}$$

Extended Kalman Filter (EKF) algorithm [91] for (3.33)-(3.36) is designed as follows:

The prediction update is

$$\begin{aligned} \overline{\mu_t} &= g(\mu_{t-1}, u_t) \\ \overline{\Sigma_t} &= G_t \Sigma_{t-1} G_t^T + R_t. \end{aligned} \tag{3.30}$$

The measurement update is

$$
\begin{aligned}
K_t &= \overline{\Sigma_t} C_t^T (C_t \overline{\Sigma_t} C_t^T + Q_t)^{-1} \\
\mu_t &= \overline{\mu_t} + K_t (y_t - C_t \overline{\mu_t}) \\
\overline{\Sigma_t} &= (I - K_t C_t) \overline{\Sigma_t}
\end{aligned}
\tag{3.31}
$$

A simulation of the linearized motion model together with the EKF design above for the mobile platform moving on a dual circle trajectory is shown in Fig. 3.11. The dynamic equation is not used in this thesis and moved to the Appendix B for future work, such as cooperative control of multiple-agent systems.

3.3 Robot Sensors

Following are the sensors mounted on the surveillance robotic platform to measure the robot states:

1. High-speed counter, LS7366, 32-bit quadrature counter module reading the pulses from each encoder attached to the end shaft of the motor. The Arduino Mega 2560 is cycling through four modules to read the motor rotation angles via the serial interface.

2. IMU, BNO055 board integrate 3-axis 12bit accelerometer, 3-axis magnetometer and 3-axis 16-bit gyroscope together. This module includes a Cortex-M0 ARM processor to process the fusion data and export quaternions, Euler angle, rotation vector, linear acceleration, angular velocity and magnetic field. These data are transmitted through I2C bus to MCU.

3. PhidgetSpatial Precision 3/3/3 IMU has the functionality of a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis compass. It stays precision in the accelerometer when measuring less than 2g, and gyroscope precision at angular velocities less than 100/s in 16-bit resolution. It is controlled by mini-USB interface.

4. Joint encoders: magnetic type, resolution 0.35, range 0-360, PWM(deadband 1-2s) output.

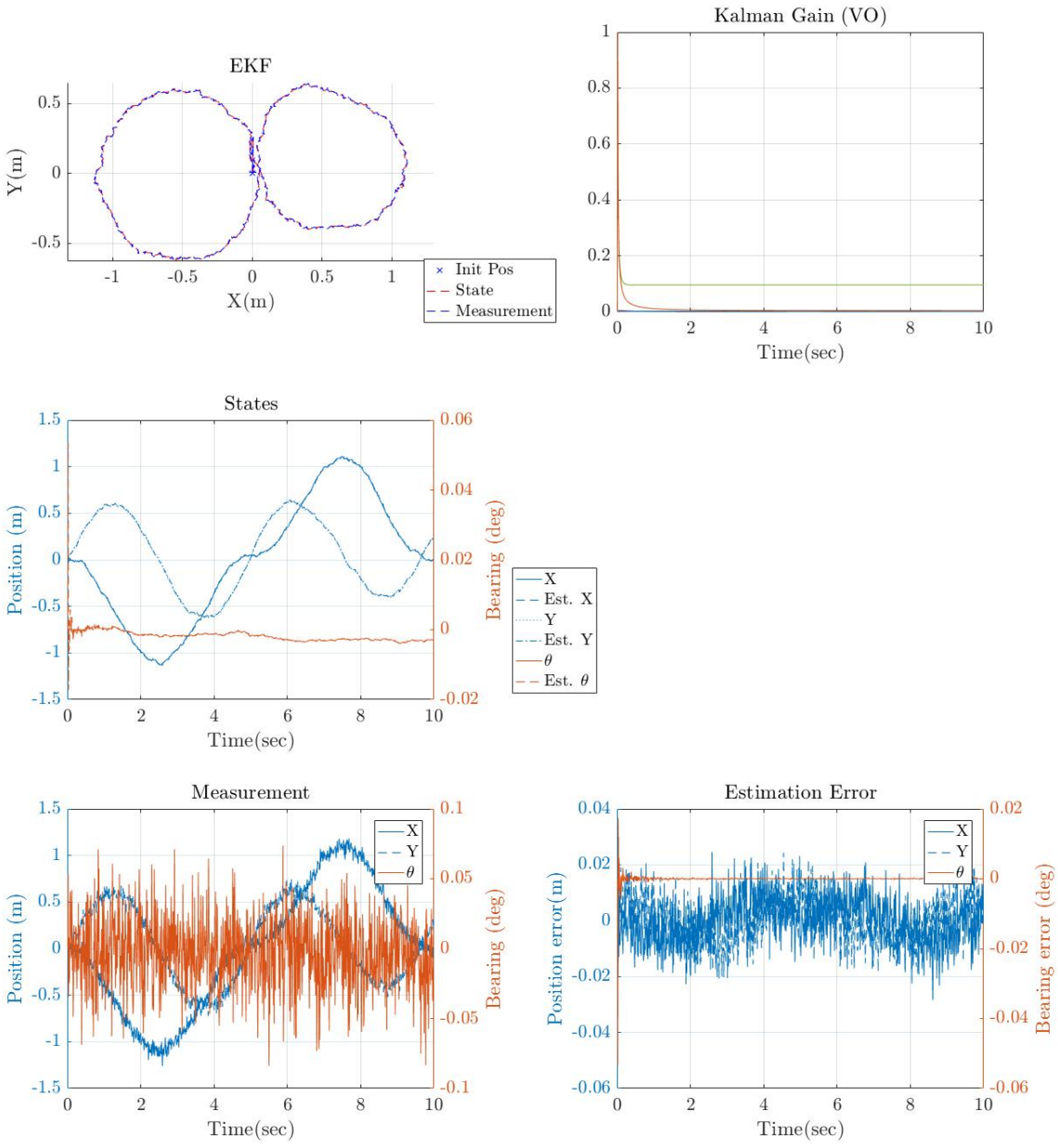

Figure 3.11: Simulation of the mobile platform and the EKF design.

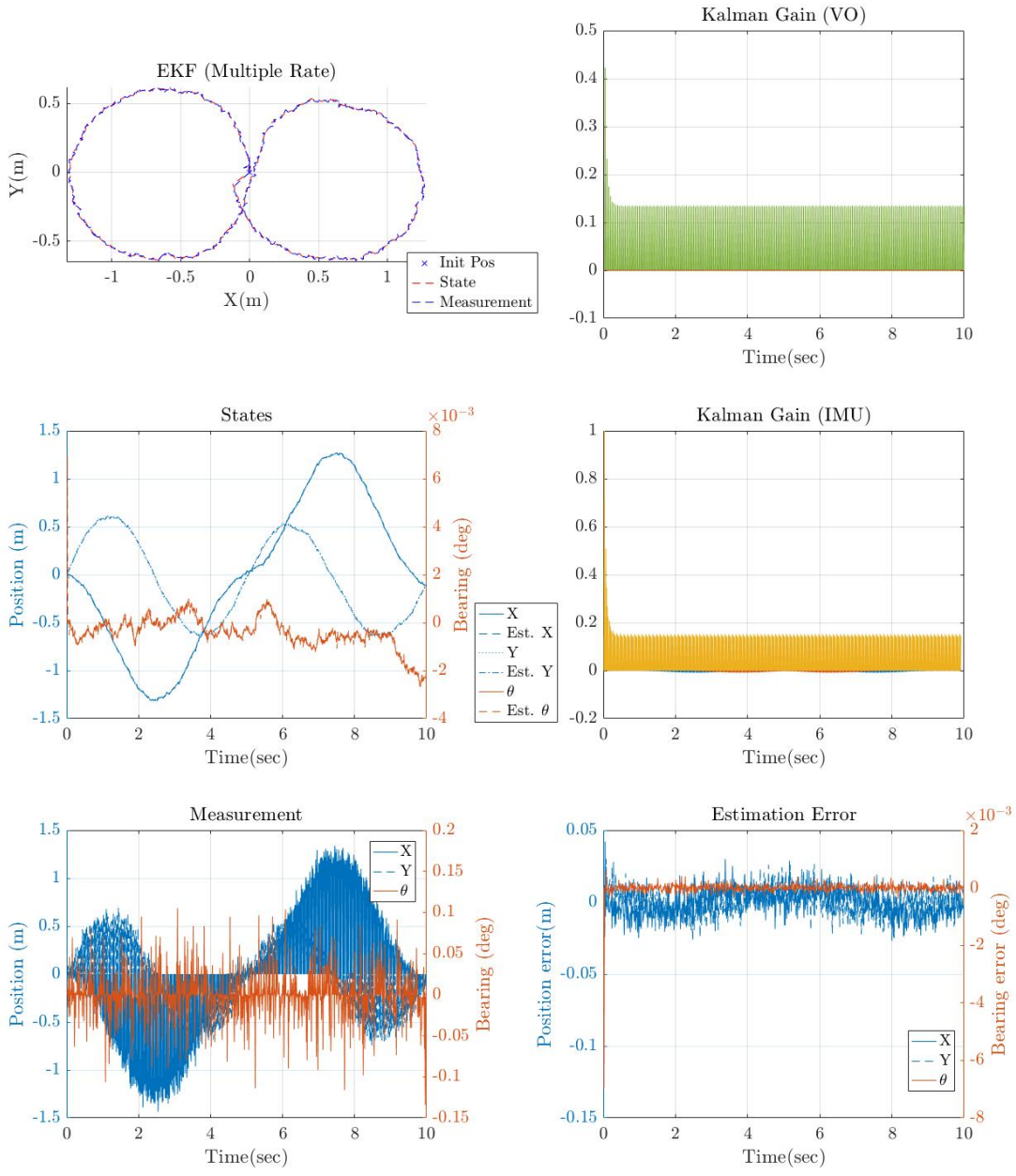

Figure 3.12: Simulation of the mobile platform and the EKF with multiple sensors design.

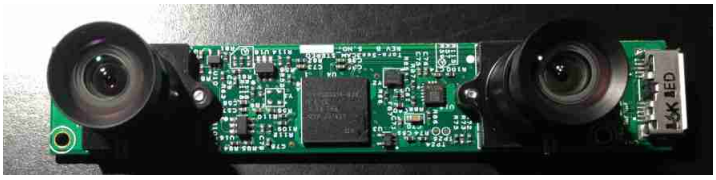

Figure 3.13: Tara stereo camera with built-in IMU.

3.3.1 Inertial Sensors

MEMS gyroscopes and linear accelerometers [99] have a list of the advantageous properties, such as compact size, low weight, rugged construction, low power consumption, short start-up time, high reliability, low maintenance and low cost. IMU used in this thesis is comprised by MEMS gyroscopes and linear accelerometers. Orientation, rotation matrix, linear acceleration and position of the mobile platform can be calculated by measurements of IMU. There are estimation approaches such as the gradient descent algorithm [66] to calculate the orientation using the acceleration and angular velocity from IMU. These are the states used for the multiple sensor fusion.

IMU Parameters

IMU sensor measures the angular velocities and linear accelerations w.r.t. three axes of a three-dimensional Cartesian coordinate system. So the noise can be defined as a function in the frequency domain, such as power spectral density (PSD) or Fast Fourier Transform (FFT). In the frequency domain, the noise becomes a function of the bandwidth of the sensor. Furthermore, the rate of angle can be integrated over time to get the angle as a function of time. The linear acceleration can also be double integrated over time to get the distance function of time.

Fig. 3.15, 3.16, 3.17, and 3.18 are the results of Allan deviation of two IMUs sitting standstill for 3 hours. In Table 3.2 and 3.3 are the parameters calculated from the sensor measurement. **Random Walk** (RW) is defined as $(\frac{unit}{\sqrt{time}})$ to describe the average deviation or error that occurs when we integrate the signal. When the sensor is rest on the table, the integration of the measured angular velocity should be zero degree. However, the noise is also integrated that causes the angle drifting by time.

$$\Sigma_{LSM6DS0 \quad Gyro} = \begin{bmatrix} 0.0097 & 0.0009 & 0.0004 \\ 0.0009 & 0.0059 & 0.0003 \\ 0.0004 & 0.0003 & 0.0060 \end{bmatrix}. \tag{3.32}$$

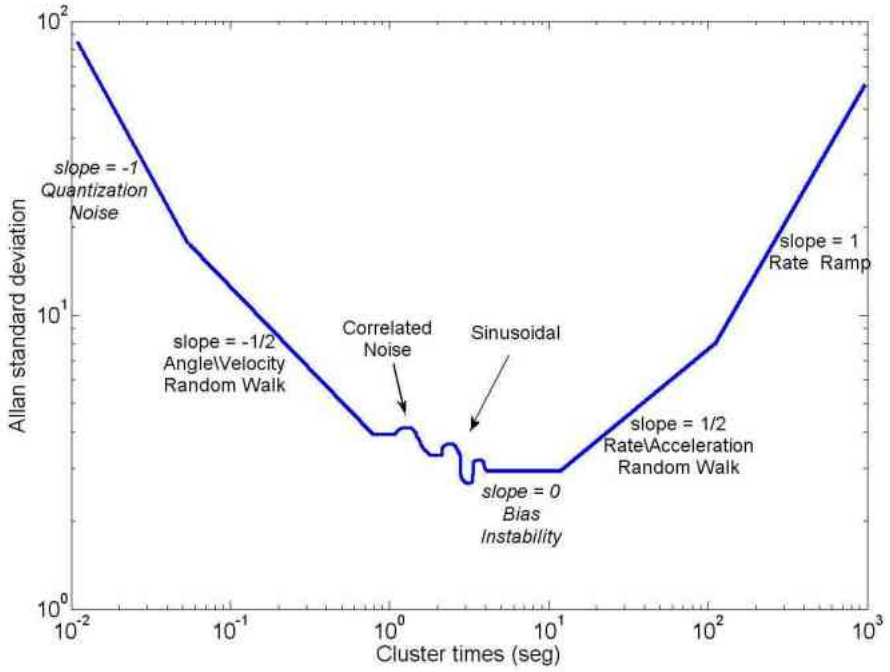

Figure 3.14: Allan deviation of an inertial sensor, cf[21].

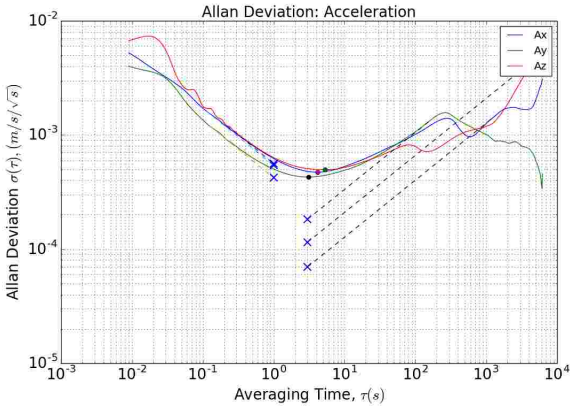

Figure 3.15: LSM6DS0 Accelerometer.

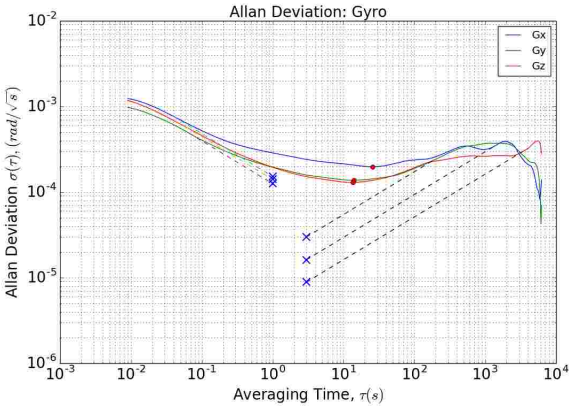

Figure 3.16: LSM6DS0 Gyro.

Table 3.2: LSM6DS0 IMU parameters.

Axis	White Noise	Bias Instability	Rate Random Walk
Ax	5.4319e-04	4.7165e-04	6.9490e-05
Ay	4.2224e-04	4.2760e-04	**1.8210e-04**
Az	**5.6288e-04**	**4.9614e-04**	1.1453e-04
Gx	**1.5252e-04**	**1.9684e-04**	1.6154e-05
Gy	1.2514e-04	1.3722e-04	**2.9837e-05**
Gz	1.4034e-04	1.2923e-04	8.8494e-06

Table 3.3: Phidget IMU parameters.

Axis	White Noise	Bias Instability	Rate Random Walk
Ax	9.8209e-04	**2.3518e-04**	**1.6970e-05**
Ay	8.9154e-04	2.2659e-04	1.5818e-05
Az	**1.1036e-03**	2.2425e-04	1.4733e-05
Gx	**4.3330e-04**	**1.1008e-04**	**7.6734e-06**
Gy	3.9599e-04	8.7191e-05	7.5833e-06
Gz	3.1963e-04	1.0348e-04	1.0230e-06

$$\Sigma_{LSM6DS0 \quad Acc} = \begin{bmatrix} 0.0000777 & 0.0000070 & 0.0000424 \\ 0.0000070 & 0.0000764 & -0.0000185 \\ 0.0000424 & -0.0000185 & 0.0001578 \end{bmatrix}. \tag{3.33}$$

$$\Sigma_{PhidgetSpatial \quad Gyro} = \begin{bmatrix} 0.0223 & -0.0003 & 0.0004 \\ -0.0003 & 0.0176 & -0.0004 \\ 0.0004 & -0.0004 & 0.0162 \end{bmatrix}. \tag{3.34}$$

$$\Sigma_{PhidgetSpatial \quad Acc} = \begin{bmatrix} 0.00001827 & -0.00000408 & -0.00000046 \\ -0.00000408 & 0.00001370 & 0.00000159 \\ -0.00000046 & 0.00000159 & 0.00000908 \end{bmatrix}. \tag{3.35}$$

3.3.2 Passive Vision Sensors

We exploit the stereo camera to estimate the depth of obstacles using two cameras with a fixed baseline (6cm). The point clouds we use to detect obstacles are coming from stereo

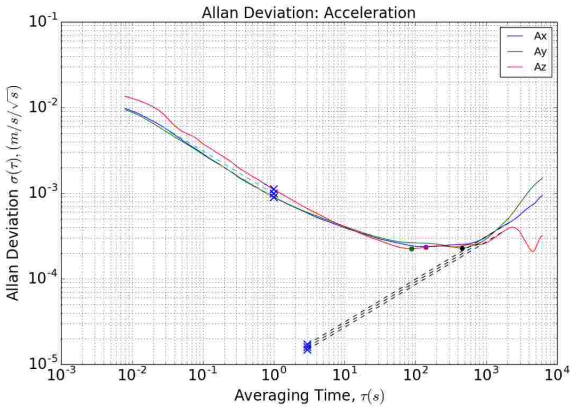

Figure 3.17: PhidgetSpatial Accelerometer.

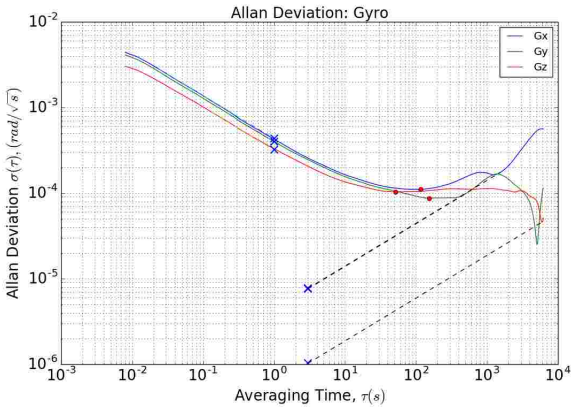

Figure 3.18: PhidgetSpatial Gyro.

block matching algorithm [23] which processes the undistorted images. Then, the metric depth of each point cloud can be used to update the occupancy map.

Camera Parameters

Distortion is mainly caused by a characteristic of the lens. In order to detect the features in different position in an image, we do need to do the camera calibration to get distortion parameters. Then we can undistort the images and use the result to rectify images before generating the point clouds. There is a ROS package [15, 67] available.

Table 3.4: Tara stereo camera calibration, pinhole model.

Left Camera	k_1	k_2	k_3	k_4
Pinhole model	8.982447e-2	-1.07281e-1	1.005296e-3	-1.59310e-3
Radtan model	753.530413	752.6850	305.7520	259.7882
Right Camera				
Pinhole model	8.49042e-2	-0.107633	-5.36353e-4	-1.4541e-3
Radtan model	756.7709	756.05422	304.31874	252.66928

$$T_{C_0,C_1} = \begin{bmatrix} \mathbf{0.9999973} & -1.663546e-05 & 2.282859e-3 & \mathbf{-5.9340e-2} \\ 1.39334e-05 & \mathbf{0.9999992} & 1.18363e-3 & -4.4084889e-4 \\ -0.002282 & -0.0011836 & \mathbf{0.99999} & 7.9437e-4 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (3.36)$$

3.3.3 Stereo-Cameras and IMU Calibration

In order to derive the intrinsic and extrinsic parameters of the IMU and cameras, Kalibr [44, 67], a well-known offline toolbox, can be used to measure the transformation among multiple cameras and IMUs. The first step is using the stereo camera from different poses to capture images of a static April grid, which comprises different coded April tags in grid cells and separated by equal space. Since the tags' size are given, thereby the metric distance from the camera to each tag is measurable. Therefore, the camera poses in the April tag frame can be estimated For IMU trajectory, it can be derived from Newton's law. Thus, the static transformations between camera(s) and IMU(s) can be estimated through the nonlinear optimizers.

The stereo camera has two cameras, the left camera, cam_0, is the main camera used in the SLAM algorithm. The right camera, cam_1, is provided as the reference image to calculate disparity and point clouds.

Table 3.5: Tara stereo camera and IMU calibration.

Reprojection error	mean	median	std
cam_0 [px]	0.217889164512	0.203345692995	0.118197230139
cam_1 [px]	0.225464454778	0.207969892634	0.12640559052
Gyro [rad/s]	0.509795240203	0.438614100972	0.392775070394
Acc $[m/s^2]$	0.0305871166197	0.0229676499537	0.0391722635506

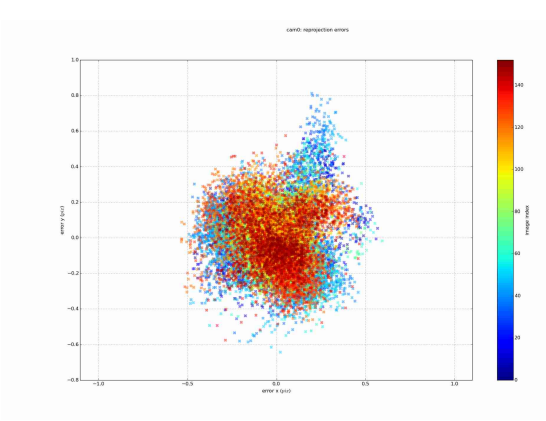

Figure 3.19: Left camera reprojection errors.

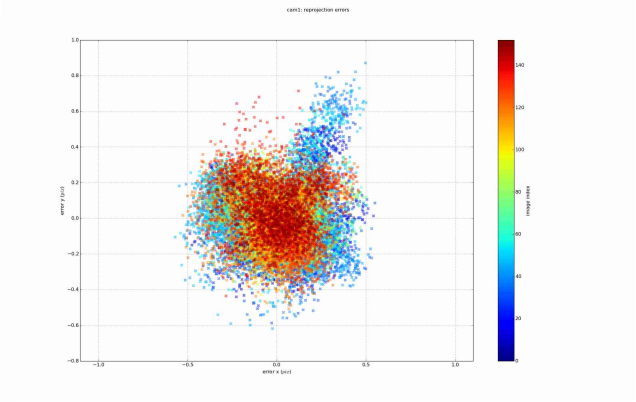

Figure 3.20: Right camera reprojection errors.

Following are the derived static transformation matrices from the IMU to the left camera ,$T_{C_0,I}$, and the IMU to the right camera,$T_{C_1,I}$.

$$T_{C_0,I} = \begin{bmatrix} 0.08028368 & 0.99662157 & -0.01731996 & \mathbf{0.04867814} \\ -0.99643282 & 0.08069753 & 0.02468892 & -0.00194908 \\ 0.02600319 & 0.01527606 & 0.99954513 & 0.06839485 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.37}$$

$$T_{C_1,I} = \begin{bmatrix} 0.08035941 & 0.9966525 & -0.01503851 & \mathbf{-0.01050617} \\ -0.99640023 & 0.08072944 & 0.02587176 & -0.00230829 \\ 0.02699921 & 0.01290533 & 0.99955215 & 0.06908017 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.38}$$

3.3.4 Vision-Inertial Model

Through vision-inertial SLAM algorithms [94], the position and orientation of the perception camera can be derived as following discrete motion model:

$$p_{k+1} = p_k + v_k \Delta t. \tag{3.39}$$
$$v_{k+1} = v_k + R_k(a_z - b_a)\Delta t. \tag{3.40}$$
$$R_{k+1} = R_k exp([\omega_z - b_\omega]_\times \Delta t). \tag{3.41}$$

where $[\cdot]_\times$ is the skew-symmetric matrix for the cross product. The b_a is the bias of the linear accelerometer, b_ω is the bias of the gyroscope.

3.4 Articulated Arm Calibration

The 7Bot robotic arm is designed for education and entertainment. Therefore, its accuracy is not as good as industrial manipulators. In order to convert the measurements from EF frame to mobile base frame, the offsets of each link and each joint angle are required. Thus, we formulate a cost function to estimate the offsets through an experiment to gather the end-effector poses from the forward-kinematics and the ground truth by motion capture system, OptiTrack. The setup is shown in Fig. 3.21. Four markers formed a rigid body are placed on the base of the robotic arm. Similarly setting, five markers are mounted on the camera case. Driving each joint motor, the motion capture system is streaming out the trajectory of end-effector $\mathbf{p}_(t)$to the PC. Meanwhile, all the joint angles $\theta_i(t)$ are also recorded to compute the end-effector position using forward-kinematics. The result is on the left graph in Fig. 7.3.

Figure 3.21: Measure the end-effector trajectory using motion capture system.

In order to aligned the trajectory which is derived from forward-kinematics with the ground truth, following cost function is defined with joint angle bias and link offsets:

$$\min_{\boldsymbol{a_b},\boldsymbol{\alpha_b},\boldsymbol{\theta_b},\boldsymbol{d_b}} \sum_{i=1}^{N} \|\boldsymbol{p}_i - \mathbf{x}_e(i, \boldsymbol{a_b}, \boldsymbol{\alpha_b}, \boldsymbol{\theta_b}, \boldsymbol{d_b})\| \tag{3.42}$$

where p_i is the position of the geometrical centre of several marks measured by OptiTrack at the i-th step in discrete time, $\mathbf{x}_e$ is the position of end-effector by forward-

Link i	d_i(mm)	a_i(mm)	α_i	$\theta_i(t)$
1	$77.8 + d_{1b}$	$30 + a_{1b}$	$\frac{\pi}{2} + \alpha_{1b}$	$\theta_1(t) + \theta_{1b}$
2	$0 + d_{2b}$	$120 + a_{2b}$	$0 + \alpha_{2b}$	$\theta_2(t) + \theta_{2b}$
2_v	$0 + d_{2vb}$	$120 + a_{2vb}$	$0 + \alpha_{2vb}$	$\theta_{2v}(t) + \theta_{2vb}$
3	$0 + d_{3b}$	$29.42 + a_{3b}$	$\frac{\pi}{2} + \alpha_{3b}$	$\theta_3(t) + \theta_{3b}$
4	$198.5 + d_{4b}$	$0 + a_{4b}$	$\frac{\pi}{2} + \alpha_{4b}$	$\theta_4(t) + \theta_{4b}$
5	$0 + d_{5b}$	$0 + a_{5b}$	$-\frac{\pi}{2} + \alpha_{5b}$	$\theta_5(t) + \theta_{5b}$
6	$0 + d_{6b}$	$0 + a_{6b}$	$0 + \alpha_{6b}$	$\theta_6(t) + \theta_{6b}$

Table 3.6: D-H table of the 7Bot robot arm with joint angle biases and linkage offsets.

kinematics with following undetermined biases in the D-H table, $\mathbf{a_b}, \alpha_\mathbf{b}, \theta_\mathbf{b}, \mathbf{d_b}$. Use Levenberg-Marquardt algorithm to compute this nonlinear least-squares function and derive $\mathbf{a_b}, \alpha_\mathbf{b}, \theta_\mathbf{b}, \mathbf{d_b}$. Later, we applied the offsets to the forward-kinematic function. The undetermined biases and offsets are defined in Table 3.6.

Chapter 4

Localization and Mapping

Current state-of-the-art visual SLAM algorithms can build a precise 3D map and export the camera poses in the specific conditions, such as proper illumination, static objects around and distinguishable textures in the observed scene. However, the visual SLAM algorithms will fail when the eye-in-hand camera traverses through a gap due to invalid to the above conditions. In this research, we propose the two operation modes to retain the camera pose when the visual SLAM algorithms fail in the gap mapping mode. The key point is using the forward kinematic equation of the robotic arm to estimate the camera poses. The estimated camera pose can be used to recover the failed visual SLAM/odometry algorithms. Thus, the visual SLAM algorithms can be reset and then use the computed camera pose to initialized visual SLAM algorithms. Thereby, the visual SLAM algorithms can continue on the gap mapping mode.

4.1 Sensor Transformation

In our mobile robot, the individual sensors are mounted on different locations. In order to well estimate the robot position, all the metric measurements from the stereo camera and IMU(s) on the robot can be fused to the mobile base coordinate frame, base link. The estimation of base link will be utilized for the position and velocity control in the next chapter. Thereby, the static transformations between fixed sensors should be determined before applying the sensor fusion scheme. Kalibre [1] is an open-source tool used to calibrate such static transformations, e.g. the left camera to the right camera, $\mathbf{T_{C_2,C_1}}$; the left

[1]This product includes software developed by the Autonomous Systems Lab and Skybotix AG. [43**?**]

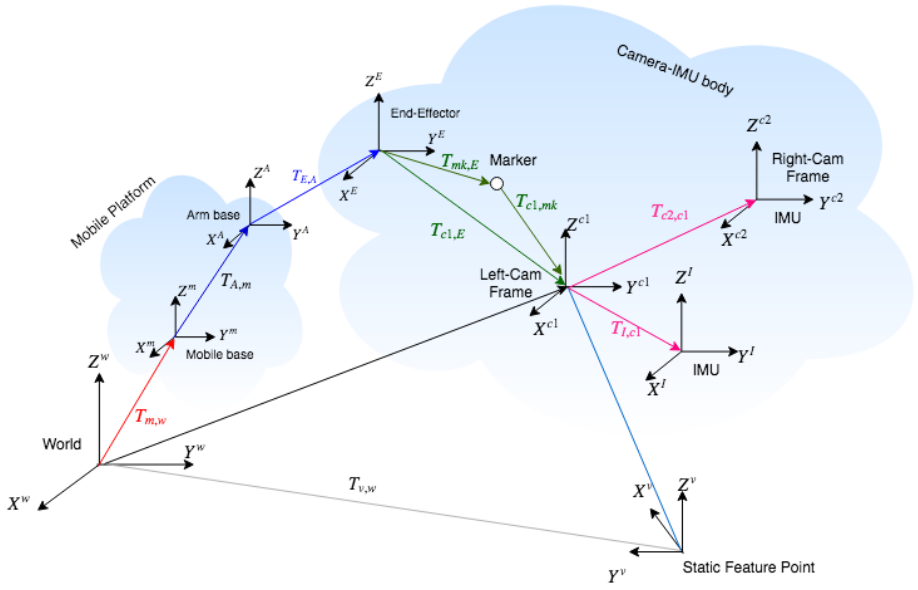

Figure 4.1: The transformation chain between all sensors in the mobile robot.

camera to IMU, $\mathbf{T_{I,C_1}}$. On the contrary, the dynamic transformation, $\mathbf{T_{E,A}}$, from the arm base to the EF is relied on the joint angles reading from joint encoders and the forward kinematic Eq. 3.2.

In our system, the weight of Tara stereo camera is 80.5 grams with an enclosure, and it is under the payload of the 7Bot robotic arm. Thus, we assume the forward kinematic equation is valid to convert from the mobile base frame to the EF.

4.2 Mapping

Two operating modes are proposed to implement the localization and mapping of cluttered environments. In the navigation mode, the mobile robot builds the **base map**. In the gap mapping mode, the robotic arm SLAM is performed to achieve **detail mapping**.

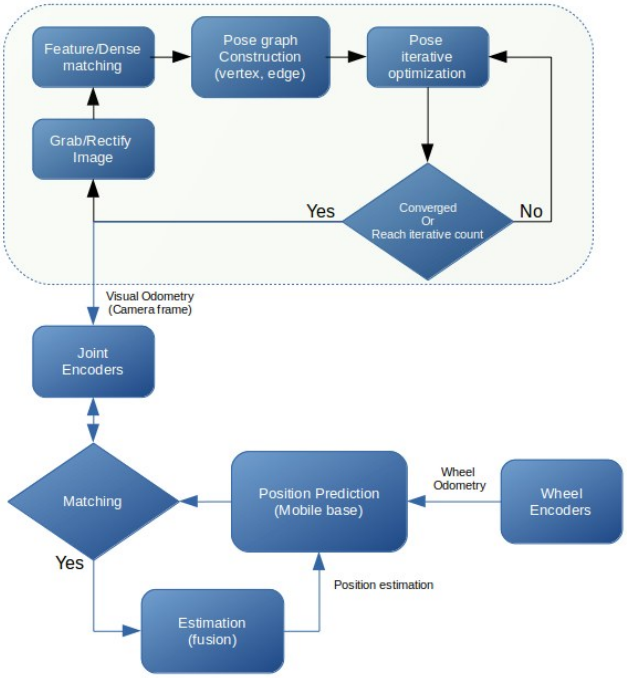

Figure 4.2: Sensor fusion pipeline.

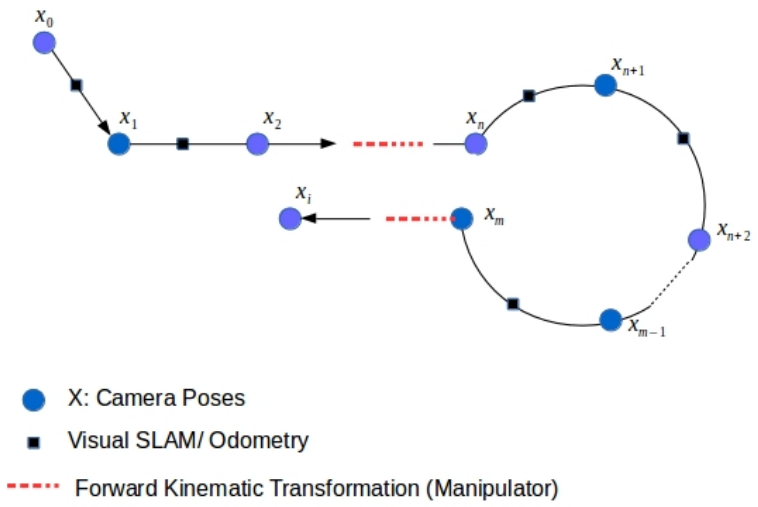

Figure 4.3: Pose graph in the detailed mapping mode.

4.2.1 Base Mapping

In the first operating mode, for navigation, the robotic arm holds the camera parallel to the x-axis of the mobile base frame shown in Fig. 3.10. The camera position, derived from the visual SLAM algorithms, can be fused to the base link of the mobile platform using the inverse transformation matrix, $\mathbf{T^{-1}_{c1,m}}$. The inverse transformation is from the left camera to the centre of the mobile base as shown in Fig. 4.1. The map built in this mode is named "base map". The mobile robot is also used the visual SLAM algorithms to localized itself in the base map.

The base mapping mode is used to build the large free space where the mobile can freely move around. A mobile robot can be controlled remotely either in the teleoperation mode or autonomously running "bug algorithm" on the PC in the mobile robot. The operators in the ground control station can watch the interior 3D map sending from the mobile robot. Furthermore, the operators have to determine all gaps in the map where may have wounded people behind and perform the gap mapping mode in the next section.

4.2.2 Detail Mapping

In the second operating mode, for gap mapping, we do need to confirm the visual SLAM algorithms are alive before we anchor our mobile platform. Thus, our mobile robot can build the consistent map. In this mode, the camera position is mainly maintained by the forward kinematic transformation matrix, $\mathbf{T_{c1,m}}$, the serial transformation from static mobile base to the camera frame, as shown in Fig. 4.1. The operators can look into the gaps because the obstacles may occlude survivors.

If a gap between obstacles is wider than the equipped camera on the EF, and the working envelope of the robotic arm can cover the gap, then the eye-in-hand camera can traverse through the gap in the detail mapping mode. However, the critical challenge in the gap mapping mode is the visual SLAM algorithms to deal with the blurred images caused by a camera too closed to the obstacles and out of focusing. Such blurred images without distinguishable textures will potentially lead to visual SLAM/odometry algorithms lost tracking. Therefore, the failed visual SLAM algorithms cannot output the camera position, and no more map can be built. In our system, the way to recover the camera's position is fusing the known position of the standstill mobile base. Thereby, the transformation, $\mathbf{T_{c_1,m}}$, from the mobile base to the left camera frame can be utilized to derive the camera poses. We can use this estimated camera pose to reinitialize the failed visual SLAM algorithms. This is the key point to keep the visual SLAM to retain the camera position consistently in the gap mapping mode. Then, the pose graph is linked by the forward kinematic transformation as the red edges shown in Fig. 4.3. The pipeline of transition between the base mapping and the detailed mapping is shown in Fig. 4.2.

Considering the transformation matrix from the mobile base, base link, to the left camera frame can be described as:

$$T_{c_1,m}(\theta_\mathbf{i}) = T_{A,m}T_{E,A}(\theta_\mathbf{i})T_{c_1,E}. \tag{4.1}$$

where $\mathbf{T_{c_1,E}}$ is the static transformation from the end-effector to the left camera, and $\mathbf{T_{A,m}}$ is the static transformation from the mobile base to the arm base. $\mathbf{T_{E,A}}(\theta_\mathbf{i})$ is the forward kinematic transformation, a function of joint rotation angles $\theta_\mathbf{i}$ $(i = 1, 2, ..., 6)$ measured by joint encoders. Thereby, the motion of mobile base can be derived from the visual odometry through the chain of transformation matrices, $\mathbf{T^{-1}_{c_1,m}}$.

4.3 Visualization

In order to exploit the map from the SLAM algorithm for path planning and collision avoidance, we choose the occupancy grid type map, OCtree map.

The Octree map algorithm uses log odds to update the likelihood of an occupancy grid cell:

$$l_{t,i} = \log \frac{p(m^i|y_t)}{1 - p(m^i|y_t)} \tag{4.2}$$

The probability of each grid cell can be converted from the log odds ratio:

$$p(m^i|y_t) = \frac{\exp(l_{t,i})}{1 + \exp(l_{t,i})} \tag{4.3}$$

The common assumption of a uniform prior probability is set to 0.5 on each cell on the initial map, so

$$l_0 = \log \frac{p(m^i)}{1 - p(m^i)} = \log \frac{0.5}{1 - 0.5} = 0. \tag{4.4}$$

The point clouds, the stereo camera output, in Fig. 4.4 illustrates the observed objects' surface in 3D space. The point clouds are refreshed and updated at each measurement. In order to convert the point clouds to a volumetric map, the occupancy of each voxel is carried out in line 3 through 13 in Algorithm 1. Thereby, a firm occupied voxel gradually becomes a solid cube as shown in Fig. 4.5. The distance from the mobile robot to an obstacle can be computed by using radius search algorithm in [30] or the Flexible Collision Library (FCL) [72].

All the voxels in 3D map can be projected into the ground plane to form a 2D occupancy grid map in Fig. 4.6. We can add a threshold of height to filter out the voxels higher than the mobile manipulator platform to create a map which represents the real obstacles and free space to the mobile robot. Thus, we can use this 2.5D map for the path planning in the next chapter.

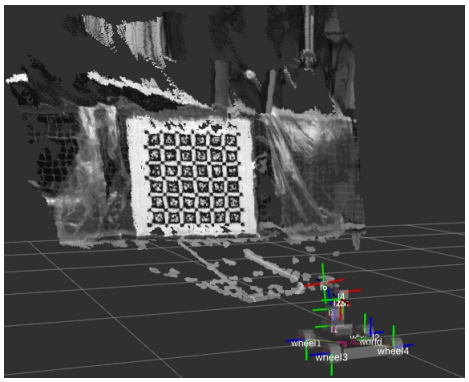

Figure 4.4: In this thesis, the point clouds is generated by SGBM algorithm implemented in OpenCV [10].

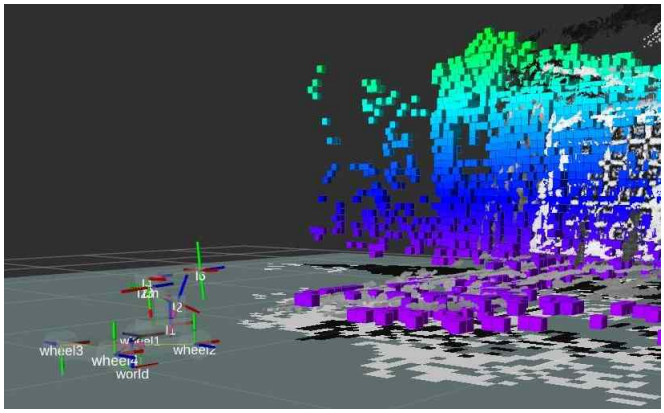

Figure 4.5: After each measurement, we can use the occupancy grid mapping algorithm to update the voxels in the 3D map.

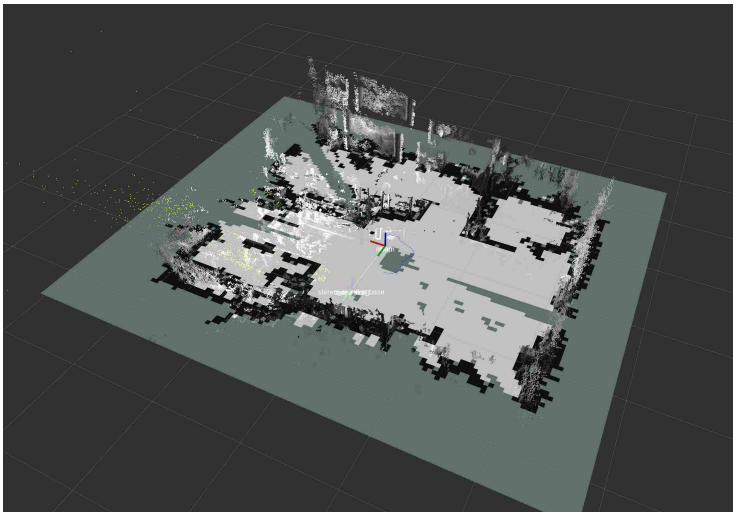

Figure 4.6: We can vertically project the occupied voxel in 3D map to the ground plane. Thus, we can convert the 3D map to 2D occupancy grid map [61].

Chapter 5

Motion Planning and Collision Avoidance

When the map has been built from a mobile robot in a hazard building, this map can be transmitted to the ground control station. Based on this real-time map, the wounded people can be identified on the map. Thus, the location can be set as a goal to the path planner to search the shortest route to get there. Consequently, the rescue crew or support robots can follow the shortest routes to arrive at the destinations as soon as possible. In this chapter, the modified wavefront, a path planning algorithm, is proposed. This algorithm can compute the shortest path more effective compared with the original wavefront algorithm.

5.1 Problem Description and Goal

The problem is set up as follows: The cluttered environment has limited accessible paths. The most popular path planner is the sample-based path planning algorithms, such as RRT, which leverage the complexity of time and calculation. However, the sample based planner may not get the shortest path if the samples are not enough on narrow paths. Another disadvantage of the sample-based planner is to produce jerky paths. Our research is focusing on the full search to find the shortest and smooth route.

5.2 Modified Wavefront for 2D Path Planning

The wavefront algorithm can guarantee to find the shortest path to a reachable goal. However, it is expensive to calculate the "cost to go" of all grid cells in the map. Thus, it is difficult for the onboard computer. The proposed methodology can reduce the computation of the local path planning.

Global Path Planning

Inspired by the coarse-to-fine approach in SLAM field, the hybrid method is proposed as shown in Algorithm 2. The First step is to reduce the resolution of the 2D occupancy map. In this process, the number of occupancy grid cells is decreasing. This low-resolution map allows the wavefront algorithm to finish the search with fewer cells. If the goal is still reachable on the low-resolution map, we can utilize the breadth first search algorithm to find the shortest path on this lower resolution map. Then, this path can be rescaled to the original map.

Local Path Planning

The path derived from the global path planner might lay on the obstacles on the original map. In this condition, the wavefront algorithm or sample based path planning algorithms can be applied to the collision area and search an accessible path to go around the obstacles in this local region. This local planner is carried out in line 8 through 13 in Algorithm 2. The other applicable scenario is when a mobile robot is on the halfway of the planned trajectory, the robot senses unexpected obstacles on its following path. The local path planning can also address this issue without recalculating the entire trajectory again.

5.2.1 Methodology

We introduce the approach with an example of point-to-point path planning within a given map as shown in Fig. 5.1 with $900(w) \times 700(h)$ cells where the red circle is the starting position and the green cross is the destination. The black cells represent occupied. The rest cells are free space as white. According to the global path planning, we reduce the original map to the scale, $\lambda = 0.1$ as shown in Fig. 5.2. Then, the low-resolution map becomes $90(w) \times 70(h)$ cells. We apply the wavefront algorithm to this downsize map as shown in Fig. 5.2. The shortest path can be derived as shown in Fig. 5.3. Thereby, we

rescale this shortest path found in the low-resolution map to the original map using the scale, λ^{-1}, as shown in Fig. 5.4. However, part of the path is in the obstacles. In these highlight areas, the path planning algorithm sets a bounding box covering the collision area and crops the box to form a local map. Consequently, the wavefront algorithm searches the shortest path on the local map, as shown in Fig. 5.5.

Algorithm 2 Modified wavefront algorithm.

function MODIFIED WAVEFRONT(Pos_G,Pos_R, λ, M):
    M_λ = Resize M to scale, λ
    $Pos_{G\lambda} = \lambda Pos_G$
    $Pos_{R\lambda} = \lambda Pos_R$
    $PathList_\lambda(Pos_{G\lambda}, Pos_{R\lambda})$ = Wavefront($Pos_{G\lambda}$,$Pos_{R\lambda}$,M_λ)
    $PathList(Pos_G, Pos_R) = \lambda^{-1} PathList_\lambda$
    for nodes in PathList **do**
        if $Path(m, n)$ in an obstacle **then**
            crop the region and save it to a submap, M_s
            $Path(m + 1, n - 1)$ = Wavefront$(m + 1, n - 1, M_s)$
            Translate $Path(m + 1, n - 1)$ to original map
            remove $Path(m, n)$
        end if
    end for
    return $Path(Pos_G, Pos_R)$
end function

The benchmark of running the modified wavefront in above example, Algorithm 2, is presented in Table 5.1, based on the Matlab© code running on a laptop with Intel© Core^{TM} i7-4710MQ CPU @ 2.50GHz × 8. The result shows that the time complexity becomes $\mathcal{O}(n \log n)$ when the map resolution is reduced.

Table 5.1: Modified wavefront algorithm benchmark.

Scale, λ	Time [sec]
0.10	**52.48732300**
0.30	89.92375400
0.50	447.71091500
0.70	2653.32454000
0.90	**5624.97035100**

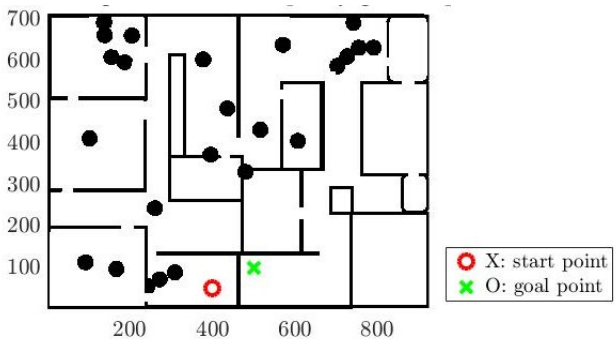

Figure 5.1: The original map.

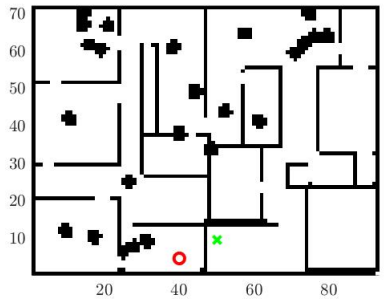

Figure 5.2: Resized map to 0.1 of original scale.

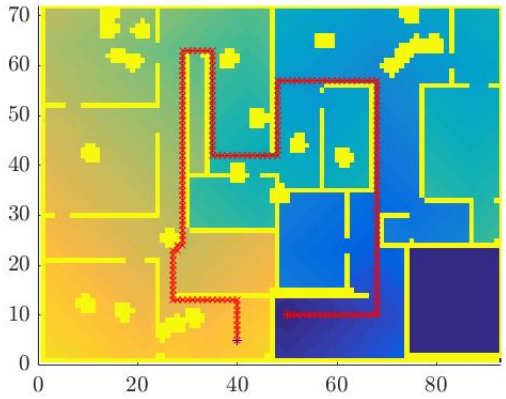

Figure 5.3: Apply wavefront path planning to the lower resolution map.

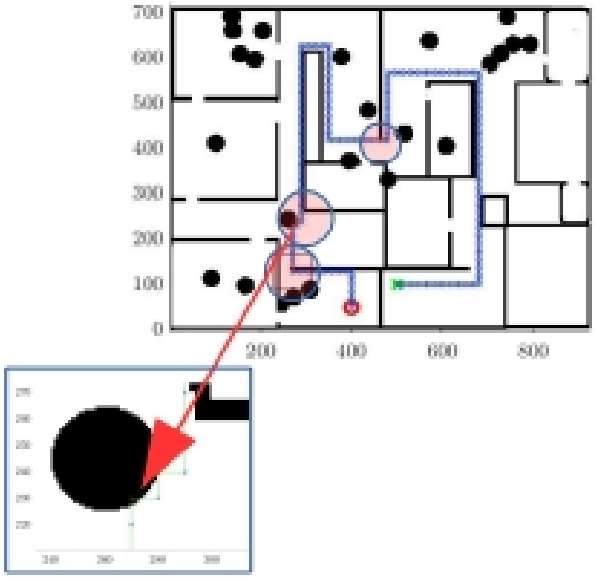

Figure 5.4: Restore the path found in the lower resolution to the original map. Some path nodes are in the obstacles highlighted by red circles.

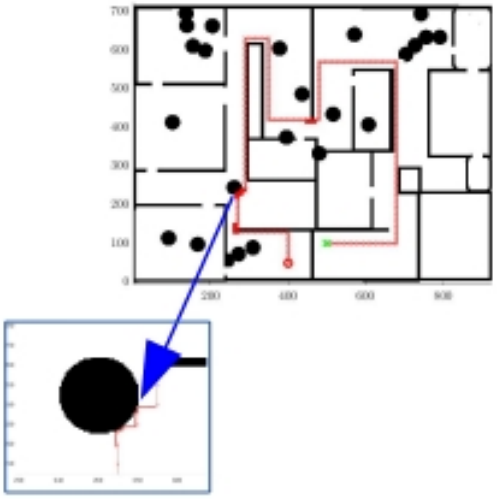

Figure 5.5: Refine the path falling on the obstacle areas.

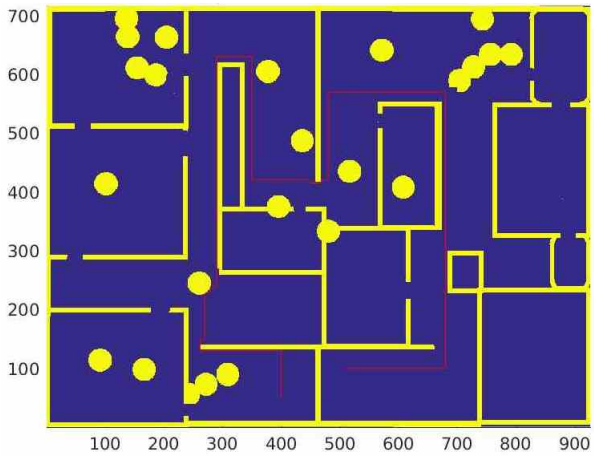

Figure 5.6: The shortest path derived by scale $\lambda = 0.1$.

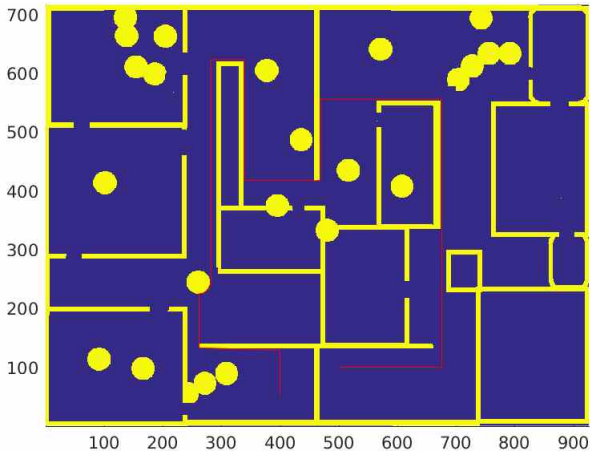

Figure 5.7: The resolution scale $\lambda = 0.7$.

Chapter 6

Planar Motion Control

For the omni-wheeled robot, its moving direction can be in any direction without nonholonomic motion constraints. We can define the trajectories without considering curvature constraints, and plan the robot to move on these trajectories without changing heading angle. From the state space model in Chapter 3, the mobile platform is a nonlinear system which is driven by inputs to the four individual wheels. Feedback control can be utilized to constrain the robot states to follow the path from the planning algorithm [55].

6.1 State Feedback

The control system of the mobile platform design is composed of two loops. One is the outer loop for path tracking, with details given in Section 6.2. The other is the inner loop controls with the wheel speed as the output and output of the outer loop as the input, as depicted in Fig. 6.1.

However, the path following with constant speed, $[V_I^T \quad \omega_I]^T = [v_{xI} \quad v_{yI} \quad \omega_I]^T$, is based on the twist velocity command, $\mathbf{u} = [v_{xId} \quad v_{yId} \quad \omega_{Id}]^T$. The motion kinematics is given by Eq. 3.24 , and the desired motion kinematics is given by

$$\dot{\xi}_d = \mathbf{u}. \tag{6.1}$$

The trajectory from path following algorithm is $\dot{\xi}_d$. Then the output, $\mathbf{u}$, of the outer loop which is the velocity command for desired kinematics becomes the input of the inner loop using PD (Proportional-derivative) control:

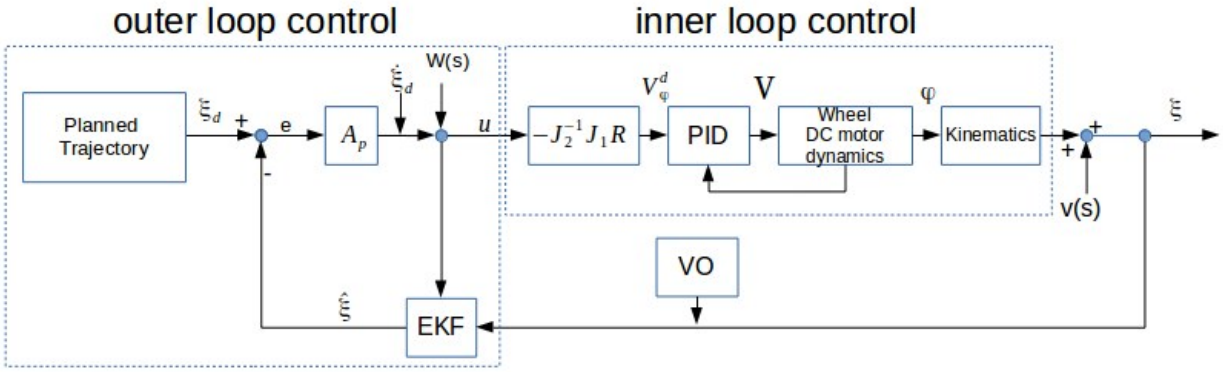

Figure 6.1: Trajectory control loop with path following algorithm.

$$\mathbf{u} = \mathbf{A}_p(\xi_d - \hat{\xi}) + \dot{\xi}_d. \tag{6.2}$$

The inner loop control maps u to the commands $V_{\varphi_i}^d$ for individual wheel speeds, $i = 1, 2, 3, 4$, i.e., $\mathbf{V}_\varphi^d = [v_{\varphi_1}^d \quad v_{\varphi_2}^d \quad v_{\varphi_3}^d \quad v_{\varphi_4}^d]$ is the desired value of $\dot{\varphi}$ in Eq. 3.24. Hence we assign

$$\mathbf{V}_\varphi^d = -J_2^{-1} J_{1sw} R(\theta)\mathbf{u}. \tag{6.3}$$

The command $\mathbf{V}_\varphi^d$ is mapped to the actual wheel DC motor voltages, $\mathbf{V} = [V_1 \ V_2 \ V_3 \ V_4]^T$ using a Proportional-Integral-Derivative (PID) controller implemented on an Arduino board. $\mathbf{V}$ through the DC motor dynamics produces the kinematic states of the platform.

6.2 Mobile Platform Path Tracking

The most of the guidance laws for the path following is using the kinematic constraint to control the vehicle with a constant speed on it. Typically, the path can be decomposed into many small sections of the straight line and circular orbit paths. In our planning method, Wavefront, which is using grid cell to express the map. So the path can be expressed by straight lines from one grid to the other. In reset of this thesis, we propose the PD controller mainly based on the straight line path following algorithm.

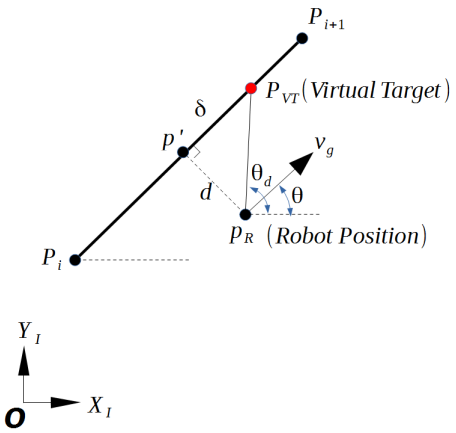

Figure 6.2: Path following a straight line.

By reference [87], the idea of this algorithm is to set a Virtual Target Point(VTP), P_V, on the path and this point is always ahead from the closest point (P') on the path to the current robot position with a fixed distance (δ). The robot is chasing the P_V till reaching the target goal (See Figure 6.2). The desired heading angle θ_d points to P_V. This algorithm is also named **Carrot Chasing Algorithm**.

Algorithm 3 Virtual Target Point Algorithm

Straight Line following

1: Initialize: $P_i = [x_i, y_i]$, $P_{i+1} = [x_{i+1}, y_{i+1}]$, $P_R = [x, y]$
2: $\overrightarrow{P_iP'} = [P_R - P_i][P_{i+1} - P_i]^T$ inner product
3: $R = \| \overrightarrow{P_iP'} \|$
4: $\overrightarrow{P_iP_V} = (\frac{R+\delta}{R})\overrightarrow{P_iP'}$
5: $P_V = [x'_t, y'_t] = P_i + \overrightarrow{P_iP_V}$
6: $\overrightarrow{P_RP_V} = [x'_t - x, y'_t - y]$
7: $\theta_d = atan2(x'_t - x, y'_t - y)$

Following is the simulation of tracking a square trajectory. The mobile platform started with prior state, $X = 0, Y = 0, \theta = 30°$ and carrot distance, $\delta = 0.05$ (m). The control speed is 1.3 (m/s).

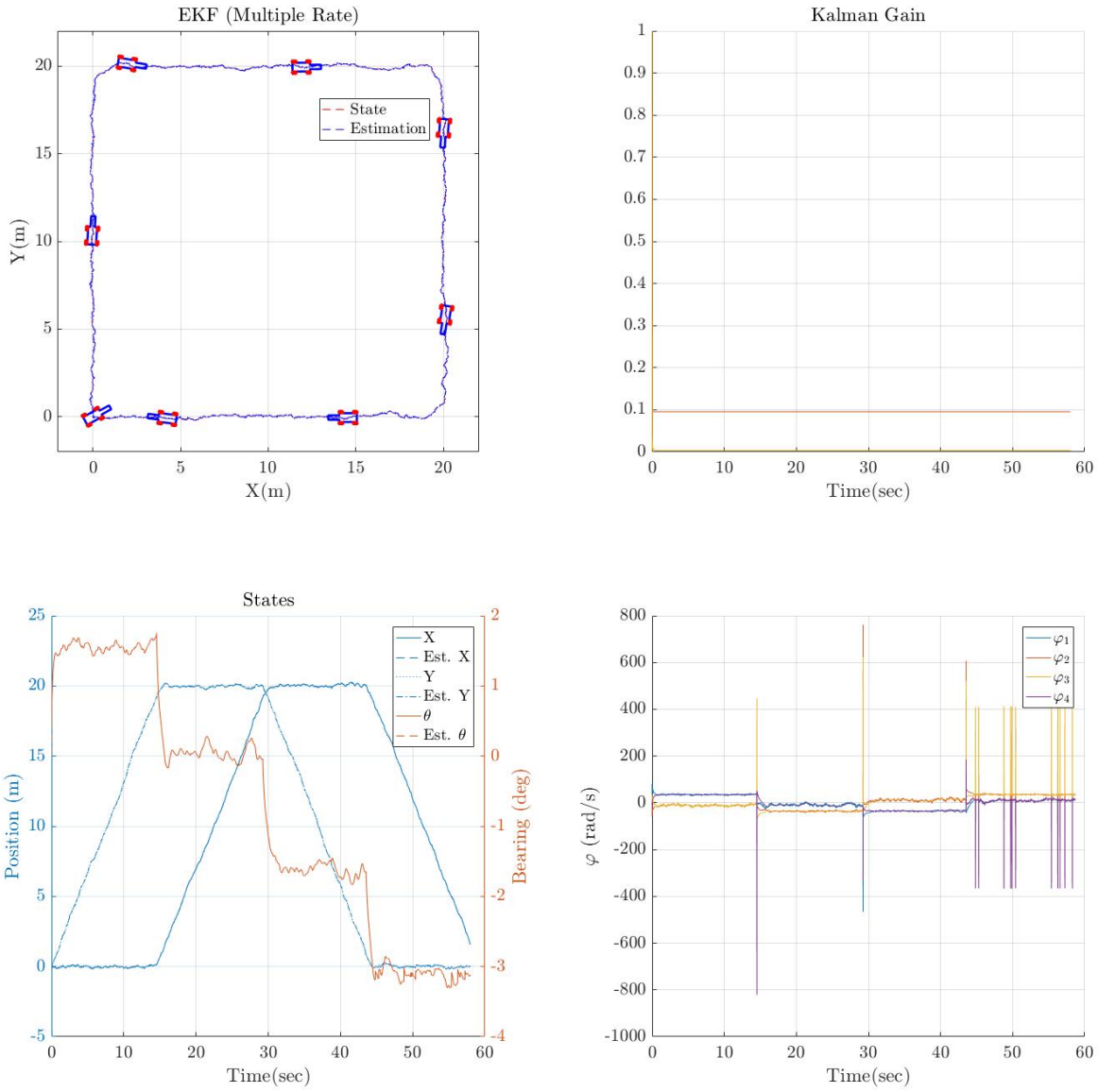

Figure 6.3: Simulation of the omni-mobile platform following a square path (top-left). The kinematic motion model includes an additive Gaussian disturbance to simulate the robot moving over uneven ground. The measurement model also contains Gaussian noise. The Kalman gain (top-right). The actual state and the estimated states (bottom-left). All wheel speeds (bottom-right).

Chapter 7

Experimental Testing

In this chapter, we shortly overview our system setup which comprises the framework and software dependencies in the first section. Consequently, we present the proposed scheme to calibrate the D-H parameters of the robotic arm on our prototype mobile platform. Subsequently, we apply the calibration results to the forward kinematic equation. Finally, using the two approaches to operate at the real mobile robot, we successfully create the 3D base map and detailed map done by the eye-in-hand camera.

7.1 System Setup

The system stack from the lower level of sensing and actuating drivers to the high level of planning is complicated, particularly the system is keeping growing. Therefore, Robot Operating System (ROS) [73] framework provides tools to manage the complexity and rapid prototypes programming, such as running multiple processes (nodes) simultaneously with interprocess communication.

The prototype of our manipulator platform system design is utilized the kinematic models described in Chapter 3 and the visual SLAM scheme in Chapter 4 to build a 3D map. The experimental implementation is mainly based on several open source projects from ROS community: g2o [6], OctoMap [8], OpenCV 3.3 [10], Ubuntu 14.04 with ROS Indigo [13], Rtab-Map [61], ORB-SLAM2 [69], Stereo DSO [97].

In ROS community, a robot model can be defined by a **Unified Robot Description Format** (URDF) file which includes sensors' location, kinematic properties, transmissions(revolute, prismatic, cylindrical) link, simulation properties and the state of a model.

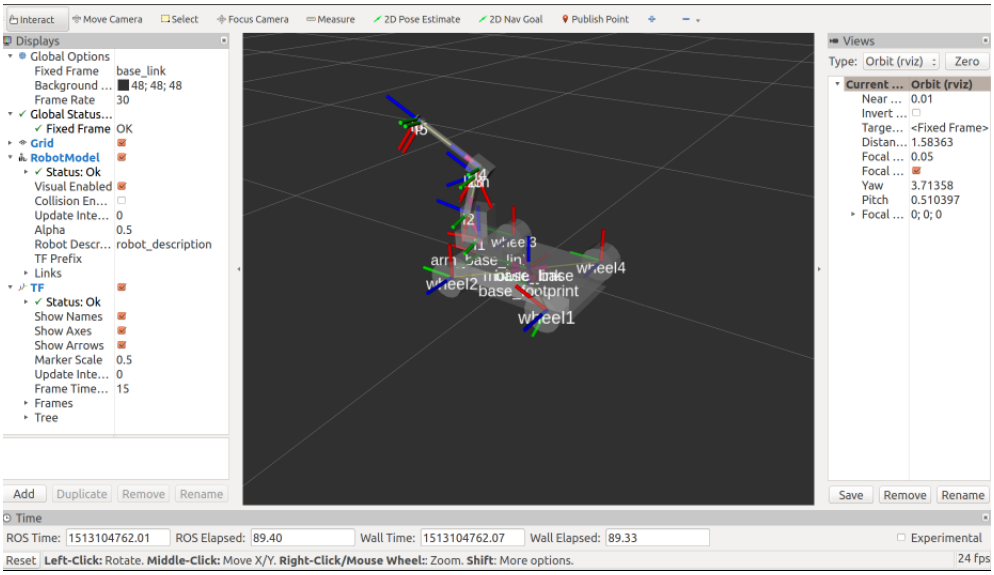

Figure 7.1: Using URDF to describe the mobile platform and robotic arm in Rviz.

In our experiment, we convert the D-H table to the portion of the robotic arm to a URDF file. A ROS node manages the transformation chain between all these sensors, subscribing all encoders outputs and then publishing all transformation matrices. The robot posture and measurements can be visualized in Rviz, as shown in Fig. 7.1.

7.2 Experiments

We have done two experiments in this thesis. One is using the proposed methodology to calibration the uncertain parameters of a 7Bot robotic arm. The other one is to perform the detailed mapping using the eye-in-hand approach.

7.2.1 Eye-in-Hand Calibration

In this section, we implement the methodology proposed in Section 3.4 to calibrate the joint biases and the link offsets. Figure 3.21 illustrates the calibration setup. Five markers are asymmetrically fixed on a rig. We mount this rig on the camera case and perform following steps to achieve our calibration.

Link i	d_i(mm)	a_i(mm)	α_i(rad)	$\theta_i(t)$(rad)
1	77.8+**12.8**	30−**4.3**	$\frac{\pi}{2}$+**0.0015**	$\theta_1(t)$+**0.0029**
2	0+**9**	120+**8.7**	0+**0.0227**	$\theta_2(t)$+**0.0829**
2_v	0−**6.9**	0+**3**	0−**0.0156**	$\theta_{2v}(t)$−**0.0095**
3	0−**6.9**	29.42−**17.1**	$\frac{\pi}{2}$+**0.0009**	$\theta_3(t)$−**0.0094**
4	198.5+**20.9**	0+**4.9**	$\frac{\pi}{2}$−**0.0048**	$\theta_4(t)$+**0.0046**
5	0−**12.1**	0−**7.2**	$-\frac{\pi}{2}$−**0.0008**	$\theta_5(t)$−**0.0004**
6	30−**6.7**	0+**1.7**	0+**0.0003**	$\theta_6(t)$+**0.0001**

Table 7.1: The corrected D-H parameters of the 7Bot robotic arm.

Data Collection

First, we launch the ROS framework and start to record the outputs of all joint encoders by 'ROS Bag'. Second, the motion capture system, OptiTrack, records the trajectory of the centre of the five markers on the eye-in-hand camera. This trajectory of the markers' rig will be treated as the ground truth utilized to calibration the offsets of the 7Bot robotic arm. Third, we move each joint individually at a time through a batch profile of joint angles shown in the Fig. 7.2. Consequently, we can calculate the EF trajectory using the forward kinematic with the measured joint angles in the ROS bag. Finally, using Levenberg-Marquardt optimizer to solve Eq. 3.42, we can derive the corrected D-H table of our 7Bot robotic arm, and the transformation from the EF to the makers rig, $\mathbf{T_{mk,E}}$. The Table7.1 presents the calibration results. Using corrected D-H parameters, the absolute translational error (RMSE) is 0.7$[mm]$.

Likewise, we can use the visual SLAM algorithm to calibrate the transformation from the markers' rig to the left camera. Two trajectories w.r.t. the ground truth of the markers' rig and the left camera poses from a visual SLAM algorithm, ORB-SLAM2, can be aligned through Levenberg-Marquardt optimizer. The results of two trajectories are shown in Fig. 7.5 and 7.6 to derive the transformation matrix, $\mathbf{T_{c1,mk}}$, from the markers' rig to the left camera. Thus, we have calibrated all the transformations in Fig. 4.1.

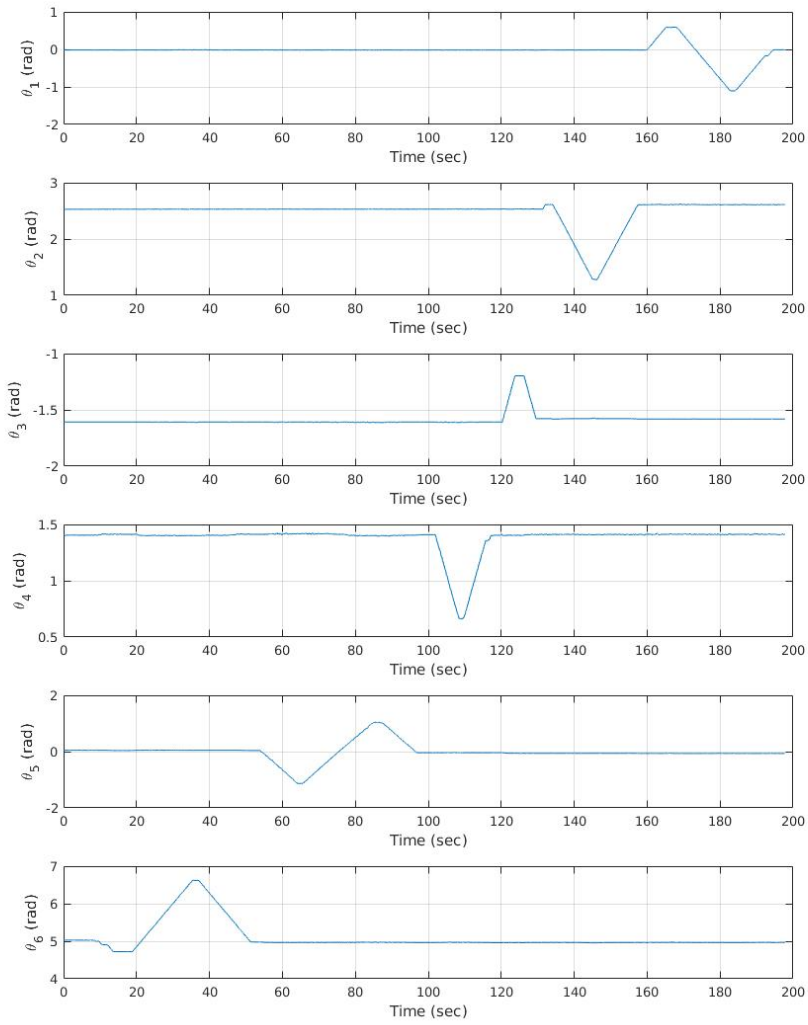

Figure 7.2: During robotic arm calibration, each joint is moved individually one at a time. All encoder readings on the robotic arm are saved to a ROS bag.

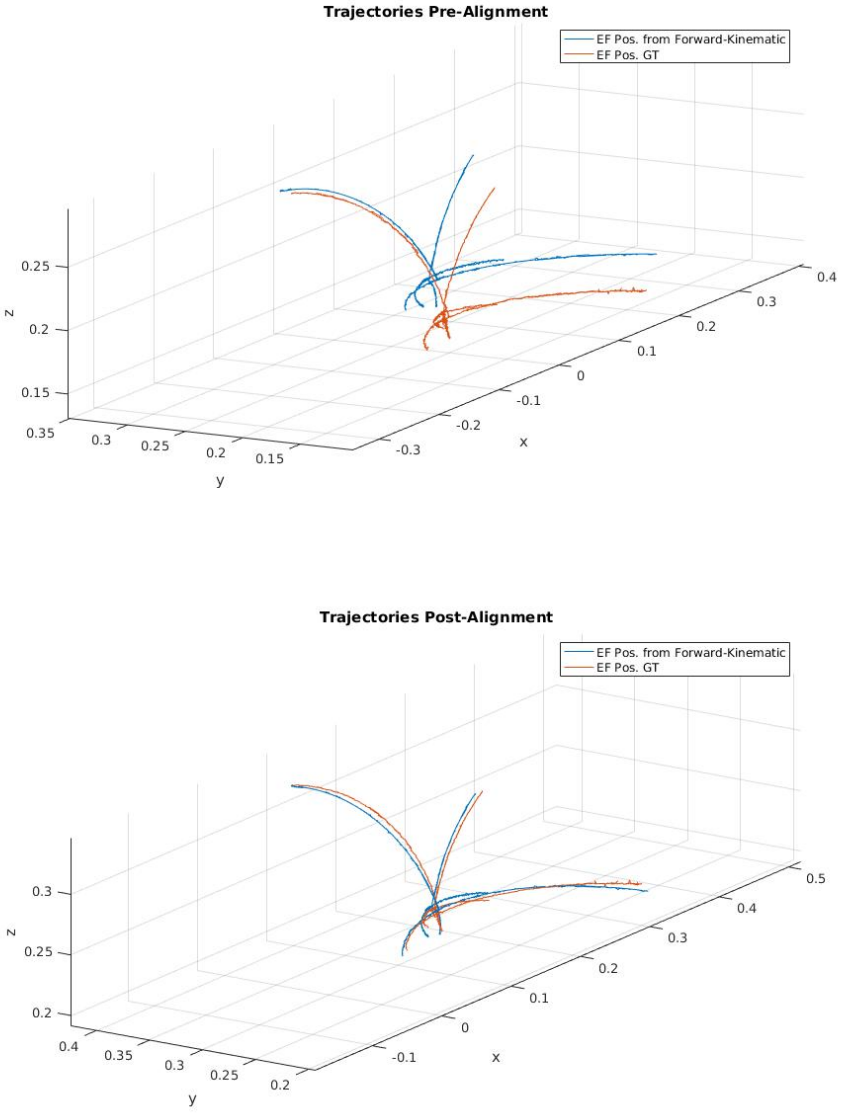

Figure 7.3: Robot arm joint biases calibration.

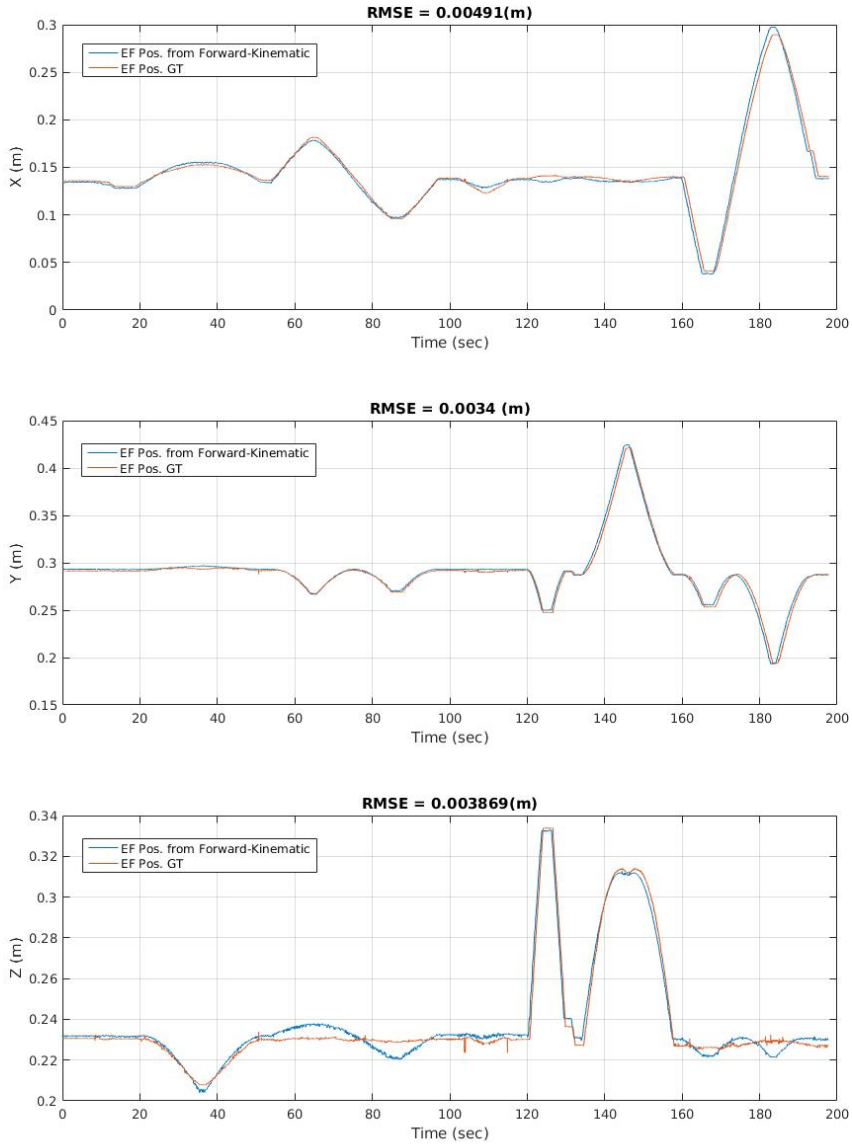

Figure 7.4: Compare the EF trajectories which is gathered by motion capture system and derived by the forward kinematic model using corrected D-H table. RMSE is 0.7[mm] after full calibration.

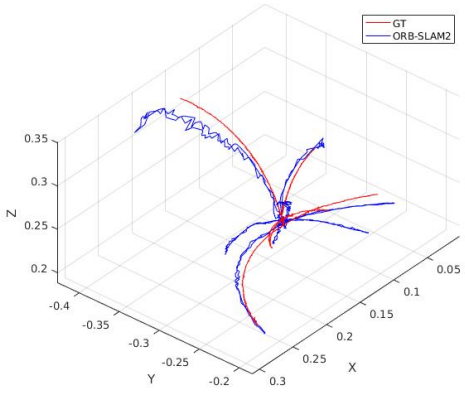

Figure 7.5: Trajectories of ORB-SLAM2 and ground truth in 3D.

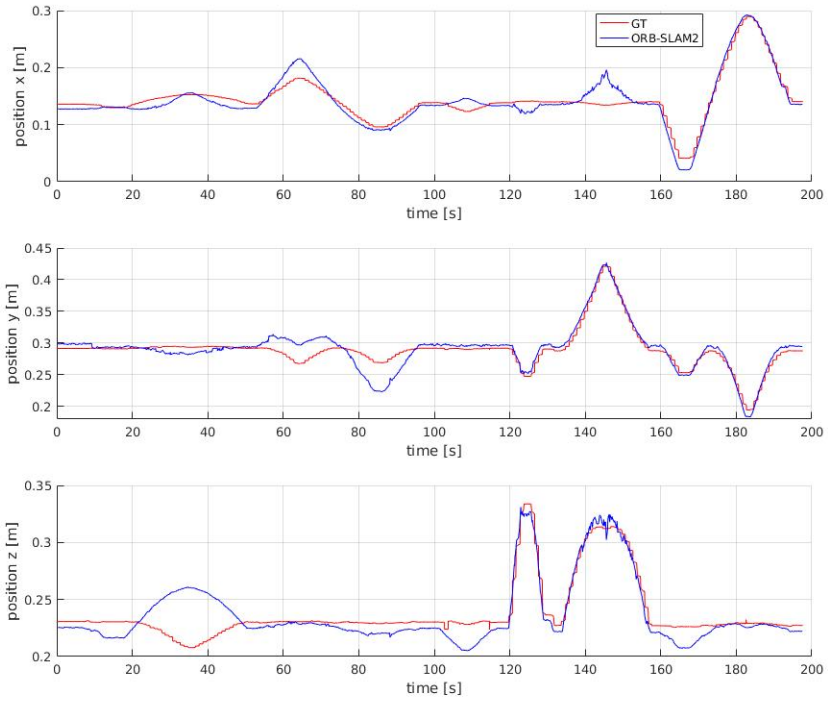

Figure 7.6: Trajectories of ORB-SLAM2 and ground truth.

7.2.2 Mapping Results

Performing the base mapping mode is straightforward when the image quality is adequate. The 3D map built in the base mapping mode is presenting the space where the omni-wheeled mobile robot can move around and observe, as shown in Fig. 4.6. However, the gap mapping is the challenge due to obscure images caused by poor illumination or out of focus, as shown in Fig. 7.7.

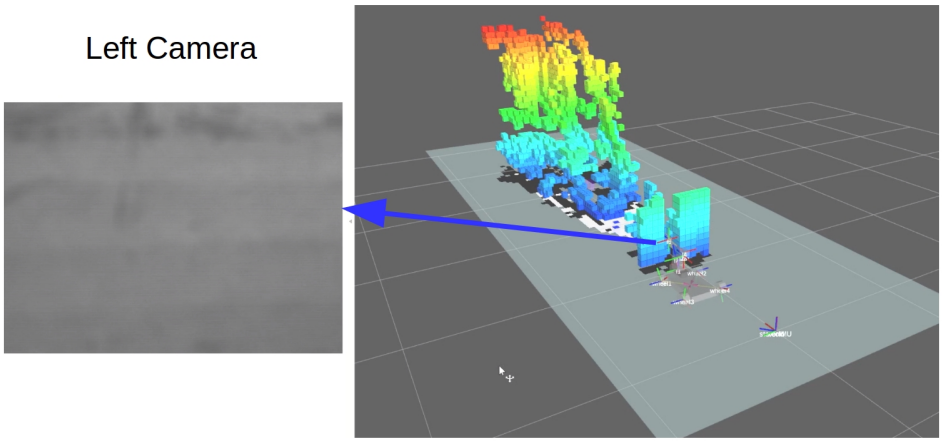

Figure 7.7: The image is blurred when the camera is too close to the obstacle around a gap.

Therefore, the mobile robot needs a secured reference point in the gap mapping process to avoid the main perception sensing, visual SLAM algorithms, break. Hence, we anchor the mobile base in front of a gap during the robotic arm moving. The mobile base position can be fused to the camera poses through the forward kinematic transformation. If the visual SLAM algorithms fail in the middle of gap mapping, it can be reset and then initialized by the sensor fusion. Such manoeuvre can keep visual SLAM algorithms alive and continue mapping, as shown in Fig. 7.10.

As starting the gap mapping, the operators rely on the gap size derived from the 3D metric map to determine whether the eye-in-hand camera can pass the gap or not. The collision detection in this operation is based on the distance from the links of the robotic

arm to the obstacles, solid voxels, on the 3D OctoMap. As the operation, we can see all links of the robotic arm in the real-time map showing on the remote computer in the ground control station and determine the distance in between the robotic arm and around voxels from different viewpoints. The fully autonomous gap mapping will be in our future work at the moment. In Fig. 7.12 and 7.13, the result of the camera trajectory is well maintained in the gap mapping mode. The 3D map beyond a gap has been built.

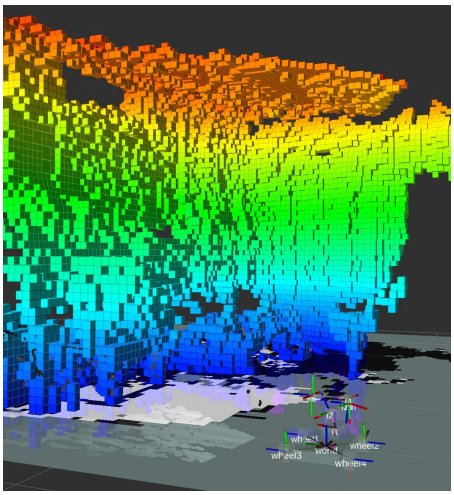

Figure 7.8: Mapping ceiling.

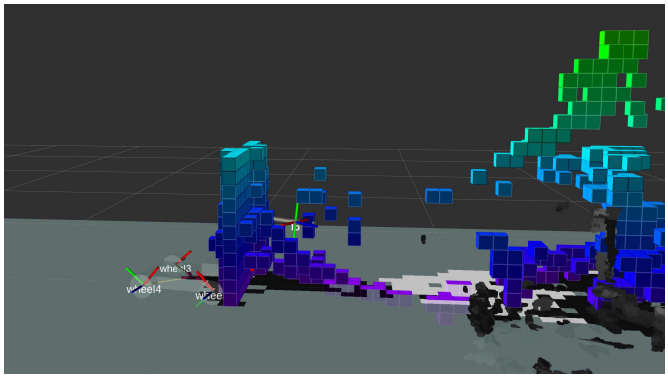

Figure 7.9: Gap mapping (side view).

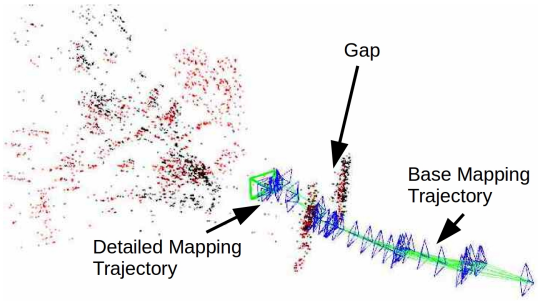

Figure 7.10: The camera trajectory is recovered by the sensor fusion, then the gap mapping can continue. This sparse map is built by ORB-SLAM2.

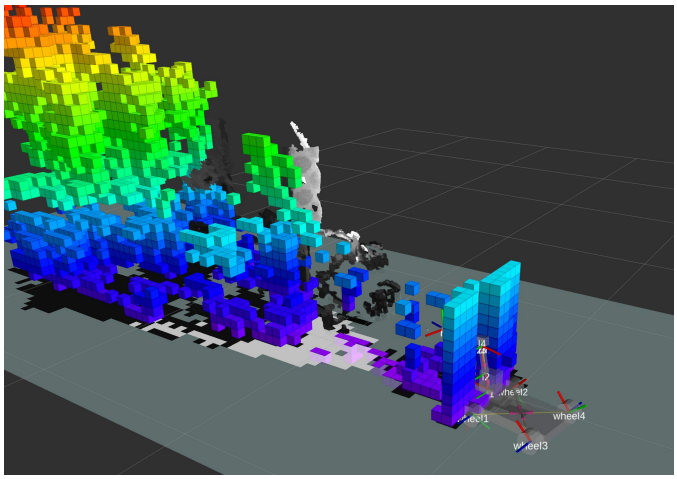

Figure 7.11: Gap mapping.

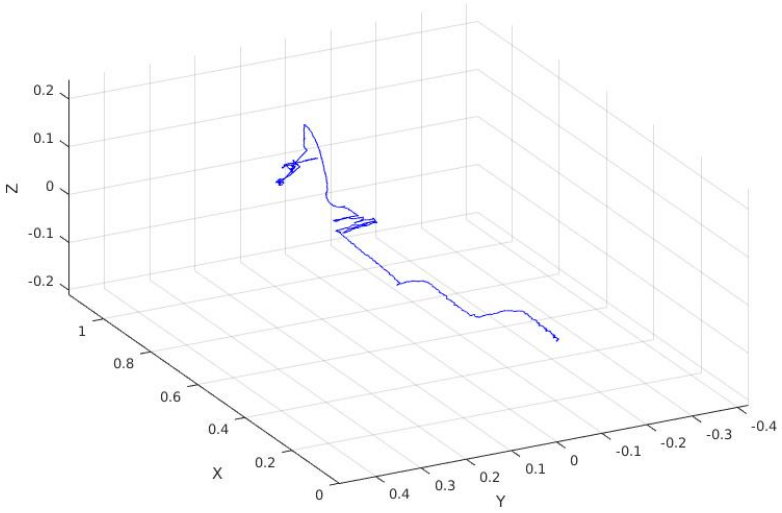

Figure 7.12: The collision-free camera trajectory in the gap mapping mode.

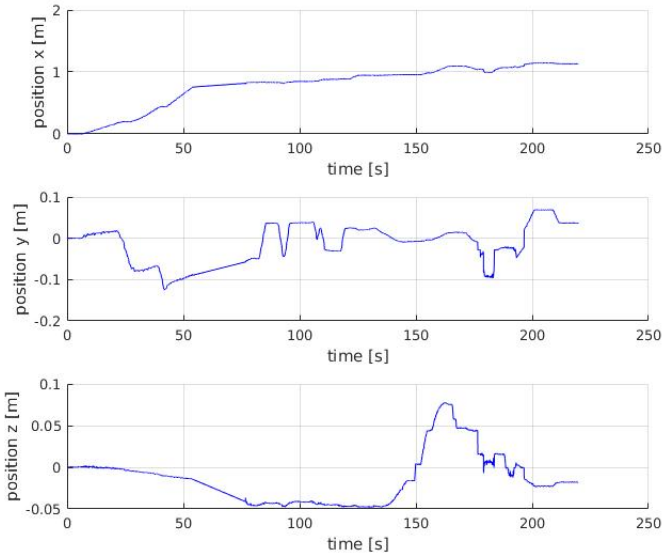

Figure 7.13: The results of the gap mapping experiment. The gap mapping started at the 53-*rd* second and ended at the 80-*th* second.

7.3 Discussion

In this research, we have done the full calibration of the transformation between all sensors. The motion control is as expected. However, there is still an issue we cannot handle properly, such as overexposure.

The accuracy of visual SLAM/odometry algorithms relies on the distinguishable textures. In this thesis, the monochrome camera outputs the 8-bit grayscale images. Therefore, the overexposure areas have the same intensity, 255. These areas turn out the unpredictable depths. In Fig. 7.14, we can see the bright spot in the middle of the image. That is caused by the longer exposure time or the strong illumination. In the saturated area, it forms a hole in the 3D point cloud map in Fig. 7.15. Thus, the mobile robot will assume the spot is a collision-free gap. If the exposure time is set to shorter, the other dark areas will become darker and fewer textures. Using machine learning to build a transformation function to perform an **adaptive histogram equalization** [88] will be the future work.

Figure 7.14: This image shows the saturated areas with the same photometric value, the white colour.

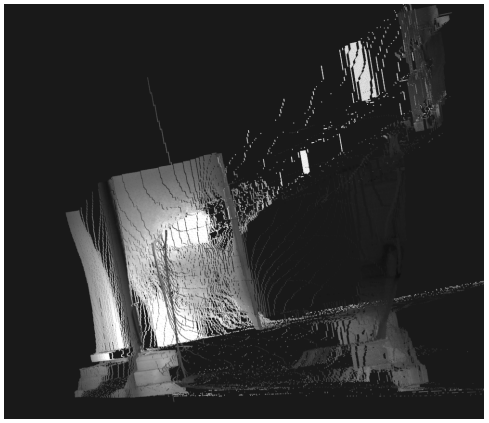

Figure 7.15: The saturated areas turn out the unexpected depths. In this image, the overexposure area becomes a hole.

Chapter 8

Conclusion

In this thesis, a mobile robot has been designed to search for objects in uncertain cluttered indoor environments using shortest path planner algorithms with the ability to avoid collisions with obstacles. This research has application in helping rescue crews find the shortest and safest paths to reach the survivors, in an indoor hazard area.

Visual sensors are commonly fix mounted on the base frame of mobile robots and face the moving direction. The transformation matrices from the C.G. of the robot to the visual sensor bases are known and unchanged once the calibrations have been done. However, such fixed cameras are constrained by nonholonomic motion and limited to horizontal observations. In this thesis, the proposed eye-in-hand setup on an omni-wheeled mobile robot increases the area that visual sensors can monitor, such as ceilings, gaps, floors and the robot itself. The benefit of the visual SLAM approach is the ability to show an intuitive 3D model in a real-time manner. This helps crews make a better informed decision.

In a damaged building, rooms or corridors may be blocked by obstacles, such as shelves or other furniture. The advantage of a compact mobile robot equipped with a robotic arm is that it can reach tight spaces and observe gaps between obstacles. On the contrary, small robots have limited payloads and cannot move heavy obstacles. Future work on cooperative multi-agent systems have the potential to address these issues. Such multi-agent systems can speed up the exploration and maximize its range. The other major subject is to use the state-of-the-art Artificial Intelligence approach to recognize the deformed and dusty objects in the scene. Overall, this is not the end of research. It is a new start to the continuation of the development of a fully autonomous rescue robot.

References

[1] Arduino. http://www.arduino.cc/.

[2] Big O notation. https://en.wikipedia.org/wiki/Big_O_notation.

[3] DBoW2. https://github.com/dorian3d/DBoW2.

[4] Dense-based SLAM. https://github.com/tum-vision/dvo_slam.

[5] FABMAP. http://mrg.robots.ox.ac.uk/fabmap/.

[6] g2o. https://github.com/RainerKuemmerle/g2o.

[7] Mars science laboratory curiosity rover. https://www.jpl.nasa.gov/missions/mars-science-laboratory-curiosity-rover-msl/.

[8] Octo map website. http://octomap.github.io.

[9] OCtree. https://en.wikipedia.org/wiki/Octree.

[10] OpenCV. http://http://opencv.org/.

[11] OpenSLAM. http://www.openslam.org/.

[12] Perspective-n-Point Solver. https://en.wikipedia.org/wiki/Perspective-n-Point.

[13] ROS. http://www.ros.org/.

[14] Sift stitching example. http://www.vlfeat.org/applications/sift-mosaic-code.html.

[15] Stereo Camera Calibration. http://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration.

[16] Waterloo Robotics Course. https://github.com/WaterlooRobotics/mobilerobotics.git.

[17] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[18] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[19] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568. IEEE, 2006.

[20] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[21] ISS Board. Ieee standard specification format guide and test procedure for single-axis interferometric fiber optic gyros. *IEEE Std*, pages 952–1997, 1998.

[22] G. Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[23] Myron Z Brown, Darius Burschka, and Gregory D Hager. Advances in computational stereo. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):993–1008, 2003.

[24] Sachin Chitta, E Gil Jones, Matei Ciocarlie, and Kaijen Hsiao. Perception, planning, and execution for mobile manipulation in unstructured environments. *IEEE Robotics and Automation Magazine, Special Issue on Mobile Manipulation*, 19(2):58–71, 2012.

[25] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[26] Javier Civera, Andrew J Davison, and José Marıa Martınez Montiel. Inverse depth to depth conversion for monocular slam. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2778–2783. IEEE, 2007.

[27] Peter I Corke. A robotics toolbox for matlab. *IEEE Robotics & Automation Magazine*, 3(1):24–32, 1996.

[28] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[29] Arun Das and Steven L Waslander. Calibration of a dynamic camera cluster for multi-camera visual slam. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4637–4642. IEEE, 2016.

[30] Daniele De Gregorio and Luigi Di Stefano. Skimap: An efficient mapping framework for robot navigation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017.

[31] Daniele De Gregorio and Luigi Di Stefano. Skimap: An efficient mapping framework for robot navigation. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2569–2576. IEEE, 2017.

[32] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[33] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[34] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[35] Hugh F Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation*, 4(1):23–31, 1988.

[36] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[37] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[38] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[39] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In *Readings in computer vision*, pages 726–740. Elsevier, 1987.

[40] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. *Robotics: Science and Systems (RSS), Rome, 2015*, 2015.

[41] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[42] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2):249–265, 2017.

[43] Paul Furgale, Timothy D Barfoot, and Gabe Sibley. Continuous-time batch estimation using temporal basis functions. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2088–2095. IEEE, 2012.

[44] Fadri Furrer, Marius Fehr, Tonci Novkovic, Hannes Sommer, Igor Gilitschenski, and Roland Siegwart. Evaluation of combined time-offset estimation and hand-eye calibration on robotic datasets. In *Field and Service Robotics*, pages 145–159. Springer, 2018.

[45] Enric Galceran and Marc Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[46] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[47] Santiago Garrido, Luis Moreno, Dolores Blanco, and Piotr Jurewicz. Path planning for mobile robot navigation using voronoi diagram and fast marching. *Int. J. Robot. Autom*, 2(1):42–64, 2011.

[48] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[49] Giorgio Grisetti, Slawomir Grzonka, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard. Efficient estimation of accurate maximum likelihood maps in 3d. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3472–3478. IEEE, 2007.

[50] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[51] Berthold KP Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[52] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988.

[53] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at http://octomap.github.com.

[54] B.E. Ilon. Wheels for a course stable selfpropelling vehicle movable in any desired direction on the ground or some other base, 1975.

[55] L. Jaulin. *Mobile Robotics*. ISTE Press Limited - Elsevier Incorporated, 2015.

[56] George Karypis and Vipin Kumar. A high performance sparse cholesky factorization algorithm for scalable parallel computers. *Frontiers of Massively Parallel Computation, 1995. Proceedings. Frontiers' 95., Fifth Symposium on the*, pages 140–147, 1995.

[57] Matthew Klingensmith, Siddartha S Sirinivasa, and Michael Kaess. Articulated robot motion for simultaneous localization and mapping (arm-slam). *IEEE Robotics and Automation Letters*, 1(2):1156–1163, 2016.

[58] DE Koditscheck and JM Hollerbach. *Robotics Research: The Ninth International Symposium*. Springer-Verlag New York, Inc., 2000.

[59] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[60] Mathieu Labbe and Francois Michaud. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745, 2013.

[61] Mathieu Labbe and François Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2661–2666. IEEE, 2014.

[62] Adam Leeper, Kaijen Hsiao, Matei Ciocarlie, Ioan Sucan, and Kenneth Salisbury. Methods for collision-free arm teleoperation in clutter using constraints from 3d

sensor data. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, pages 520–527. IEEE, 2013.

[63] Lih-Chang Lin and Hao-Yin Shih. Modeling and adaptive control of an omni-mecanum-wheeled robot. *Intelligent Control and Automation*, 4(02):166, 2013.

[64] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[65] S Lynen, M Achtelik, S Weiss, M Chli, and R Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[66] Sebastian OH Madgwick, Andrew JL Harrison, and Ravi Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1–7. IEEE, 2011.

[67] Jérôme Maye, Paul Furgale, and Roland Siegwart. Self-supervised calibration for robotic systems. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 473–480. IEEE, 2013.

[68] Justinas Miseikis, Kyrre Glette, Ole Jakob Elle, and Jim Torresen. Multi 3d camera mapping for predictive and reflexive robot manipulator trajectory estimation. *arXiv preprint arXiv:1610.03646*, 2016.

[69] R. Mur-Artal and J. D. Tards. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017.

[70] Derek R Nelson, D Blake Barber, Timothy W McLain, and Randal W Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529, 2007.

[71] Gabriel Nützi, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Fusion of imu and vision for absolute scale estimation in monocular slam. *Journal of intelligent & robotic systems*, 61(1):287–299, 2011.

[72] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3859–3866. IEEE, 2012.

[73] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. *ICRA workshop on open source software*, 3(3.2):5, 2009.

[74] Juan D. Tards Ral Mur-Artal. Orb slam2 library. https://github.com/raulmur/ORB_SLAM2.

[75] Robert Reid and Thomas Bräunl. Large-scale multi-robot mapping in magic 2010. In *Robotics, Automation and Mechatronics (RAM), 2011 IEEE Conference on*, pages 239–244. IEEE, 2011.

[76] David M Rosen, Michael Kaess, and John J Leonard. An incremental trust-region method for robust online sparse least-squares estimation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1262–1269. IEEE, 2012.

[77] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011.

[78] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[79] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer, Berlin Heidelberg, 2008.

[80] Roland Siegwart and Illah R. Nourbakhs. *Introducetion to Autonomous Mobile Robots*. The MIT Press, Cambridge, Massachusetts, 2004.

[81] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[82] Joan Sola. Slam toolbox. https://github.com/joansola/slamtb.

[83] Joan Sola, André Monin, Michel Devy, and Teresa Vidal-Calleja. Fusing monocular information in multicamera slam. *IEEE transactions on robotics*, 24(5):958–968, 2008.

[84] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[85] Walter Stockwell. Angle random walk. *Application Note. Crossbow Technologies Inc*, pages 1–4, 2003.

[86] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2, 2010.

[87] P. B. Sujit, S. Saripalli, and J. B. Sousa. An evaluation of uav path following algorithms. *2013 European Control Conference (ECC)*, pages 3332–3337, July 2013.

[88] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[89] Szymon Szomiski, Zbigniew Kaleta, Wojciech Turek, and Krzysztof Cetnarowicz. Predictive planning method for rescue robots in buildings. *Procedia Computer Science*, 76:539 – 546, 2015. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015).

[90] Feng Tan, Winfried Lohmiller, and Jean-Jacques Slotine. Simultaneous localization and mapping without linearization. *arXiv preprint arXiv:1512.08829*, 2015.

[91] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005.

[92] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3-4):253–271, 1998.

[93] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[94] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. Direct visual-inertial odometry with stereo cameras. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1885–1892. IEEE, 2016.

[95] Andrea Vedaldi. Scale Invariant Feature Transform (SIFT) . http://www.vlfeat.org/api/sift.html.

[96] Pakpoom Viboonchaicheep, Akira Shimada, and Yuhki Kosaka. Position rectification control for mecanum wheeled omni-directional vehicles. In *Industrial Electronics*

Society, 2003. IECON'03. The 29th Annual Conference of the IEEE, volume 1, pages 854–859. IEEE, 2003.

[97] Rui Wang, Martin Schwörer, and Daniel Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. *arXiv preprint arXiv:1708.07878*, 2017.

[98] Juyang Weng, Paul Cohen, Marc Herniou, et al. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on pattern analysis and machine intelligence*, 14(10):965–980, 1992.

[99] Oliver J Woodman. An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory, 2007.

[100] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673. Ieee, 1999.

APPENDICES

Appendix A

Robot Arm Calibration Result

After Optimization (Results)

Iteration	Norm of First-order Func-count	f(x)	step	optimality
0	7	9.74017		123
1	14	0.783543	0.0958549	15.4
2	21	0.0636576	0.109766	1.82
3	28	0.004862	0.062801	0.291
4	35	0.000519551	0.0197438	0.0456
5	42	0.000192332	0.0532432	0.00507
6	49	0.000192332	0.111046	0.00507
7	56	0.000162563	0.0277615	0.00567
8	63	0.000146069	0.0277615	0.00538
9	70	0.000146069	0.0277615	0.00538
10	77	0.000146069	0.00694037	0.00538
11	84	0.000133156	0.00173509	0.000465
12	91	0.000132909	0.00173509	0.000726
13	98	0.000132909	0.00173509	0.000726
14	105	0.000132567	0.000433773	0.000273
15	112	0.000132475	0.000433773	0.00027
16	119	0.000132379	0.000433773	0.000274
17	126	0.000132283	0.000433773	0.000326
18	133	0.000132171	0.000433773	0.000347
19	140	0.000132059	0.000433773	0.000336
20	147	0.000131943	0.000433773	0.000268
21	154	0.000131835	0.000433773	0.000235
22	161	0.000131727	0.000433773	0.000183
23	168	0.000131529	0.000867546	0.000209
24	175	0.000131152	0.00173509	0.000171
25	182	0.000130527	0.00347018	0.000267
26	189	0.000129653	0.00694037	0.000145
27	196	0.000129612	0.0138807	0.000427
28	203	0.000129472	0.00347018	0.000515
29	210	0.000129472	0.00347018	0.00056
30	217	0.000129448	0.000867546	0.000586
31	224	0.000129374	0.000216887	0.000287
32	231	0.000129366	0.000216887	0.000255
33	238	0.000129348	5.42216e-05	3.54e-05
34	245	0.000129347	5.42216e-05	3.28e-05
35	252	0.000129345	5.42216e-05	2.97e-05
36	259	0.000129344	5.42216e-05	2.8e-05
37	266	0.000129343	5.42216e-05	2.12e-05
38	273	0.000129341	0.000108443	2.12e-05
39	280	0.000129337	0.000216887	2.54e-05
40	287	0.000129333	0.000433773	1.39e-05
41	294	0.00012933	0.000433773	2.57e-05
42	301	0.00012933	0.000433773	2.57e-05
43	308	0.00012933	0.000108443	2.65e-05
44	315	0.00012933	0.000108443	2.89e-05

```
45          322         0.00012933      2.71108e-05         2.78e-05
46          329         0.00012933       6.7777e-06         5.68e-07
47          336         0.00012933       6.7777e-06         9.74e-07
48          343         0.00012933      1.35554e-05         1.98e-06
49          350         0.00012933      1.35554e-05         3.73e-06
50          357         0.00012933      1.35554e-05         4.67e-06
51          364         0.00012933      3.38885e-06         5.03e-06
52          371         0.00012933      8.47213e-07          1.6e-06

Optimization initial cost: 9.74e+00
Optimization final cost: 1.29e-04
```

Appendix B

Omni Wheel Platform Dynamic Model

We define the vector of configuration coordinates [63, 79], $q = \begin{bmatrix} \xi & \varphi \end{bmatrix}^T$, where ξ is the posture vector from (3.7) and control input $\varphi_i \in (-\pi, \pi]$, $(i = 1, 2, 3, 4)$ are rotation angles of the wheels.

The configuration kinematic model:

$$\dot{q} = S(q)u. \tag{B.1}$$

where $u = \eta = [0 \quad \dot{\varphi}]^T$ is inputs and $S(q) = \begin{bmatrix} R^T(\theta)\Sigma \\ I_{4\times 4} \end{bmatrix}$.

Convert (3.21) into (B.2).

$$J(q)\dot{q} = 0. \tag{B.2}$$

where

$$J(q) = \begin{bmatrix} J_{1_{sw}} R(\theta) & J_2 \end{bmatrix}. \tag{B.3}$$

From (B.1) and (B.2), $S(q)$ and $J(q)$ have the relation

$$S(q)^T J(q)^T = 0. \tag{B.4}$$

Lagrange equation:

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) - \frac{\partial L}{\partial q} = \tau + J^T(q)\lambda. \tag{B.5}$$

The mobile robot platform is moving on the plane. So we can ignore the potential energy.

1. $L = T - V$, where T is kinetic energy and V is potential energy which is zero due to the platform moving on the plane.

2. $T(q, \dot{q})$: kinetic energy of the robot platform, It can be described as a quadratic form

$$T(q, \dot{q}) = \frac{1}{2}\dot{q}^T M \dot{q}. \tag{B.6}$$

   M is symmetric positive-definite matrix.

3. τ : generalized forces associated with the torques generated by the actuators. i.e. There is no external forces applied to the robot platform. So the first row of τ is zero with respect to the posture coordinate.

$$\tau = \begin{bmatrix} 0 \\ \tau_\varphi \end{bmatrix}. \tag{B.7}$$

Then (B.5) can be rewritten as

$$M\ddot{q} + f(q, \dot{q}) = \tau + J^T(q)\lambda. \tag{B.8}$$

After left multiplying $S^T(q)$ in (B.8) and substituting $\dot{q}$ and $\ddot{q}$ by (B.1), the dynamic equation becomes,

$$S^T(q)MS(q)\dot{u} + S^T(q)M\dot{S}(q)u + S^T(q)f(q, S(q)u) = S^T(q)\tau. \tag{B.9}$$

where $M = \begin{bmatrix} m_b & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_b & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & I_b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_w & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_w & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_w & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_w \end{bmatrix}$, m_b: mass of the mobile platform, I_b: moment inertia of the mobile platform, I_w: moment inertia of four wheels (assume all wheels identical).

(B.9) can be rewritten in the compact form

$$H(q)\dot{u} + F(q,u) = S^T(q)\tau. \tag{B.10}$$

Now, we start to derive the kinetic energy of the mobile platform. We consider to involve the shift of the center of mass. $d1$ and $d2$ are the offsets of the center of the mass from the geometry centre corresponding to the X and Y axis of the robot frame. The velocity w.r.t. the C.M. is

$${}^R V_{G'} = {}^R V_G + \dot{\theta}\hat{z}_R \times {}^R r_{G'/G}. \tag{B.11}$$

The transformation matrix of C.M. offset w.r.t. the geometry centre is T_r^I.

$$T_r^I = \begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{B.12}$$

$$\text{where} \quad S(q) = \begin{bmatrix} (T_R^I)^{-1}\Sigma \\ I_{4\times 4} \end{bmatrix}, \quad S^T(q) = \begin{bmatrix} \Sigma^T((T_R^I)^{-1})^T & I_{4\times 4} \end{bmatrix},$$

$$J(q) = \begin{bmatrix} J_{1w}(T_R^I) & J_2 \end{bmatrix}, \quad J^T(q) = \begin{bmatrix} (T_R^I)^T J_{1w}^T \\ J_2 \end{bmatrix},$$

$$S^T(q)J^T(q) = \begin{bmatrix} \Sigma^T((T_R^I)^{-1})^T & I_{4\times 4} \end{bmatrix} \begin{bmatrix} (T_R^I)^T J_{1w}^T \\ J_2 \end{bmatrix} = 0 \tag{B.13}$$

.

$$\begin{aligned} L = T = \frac{1}{2}\dot{q}^T M \dot{q} = {} & \frac{1}{2} m_b[(\dot{X}_I c\theta + \dot{Y}_I s\theta + \dot{\theta} d_2)^2 + (-\dot{X}_I s\theta + \dot{Y}_I c\theta - \dot{\theta} d_1)^2] + \frac{1}{2} I_b \dot{\theta}^2 + \\ & \frac{1}{2}(m_w + \frac{I_w}{r_w^2})\{[(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I - (L+l)\dot{\theta}]^2 + \\ & [(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I + (L+l)\dot{\theta}]^2 + \\ & [(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I - (L+l)\dot{\theta}]^2 + \\ & [(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I + (L+l)\dot{\theta}]^2\} \end{aligned} \tag{B.14}$$

By Lagrangian Eq.

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}}) - \frac{\partial L}{\partial q} = F_i \tag{B.15}$$

where i=1,2,3 with respect to forces on X_r, Y_r and θ

For the force on F_{IX}:

$$\underbrace{\frac{d}{dt}(\frac{\partial L}{\partial \dot{X}_I})}_{(1)} - \underbrace{\frac{\partial L}{\partial X_I}}_{(2)} = F_x \tag{B.16}$$

Term (1) in (B.16)

$$\begin{aligned}
\frac{d}{dt}(\frac{\partial L}{\partial \dot{X}_I}) = \frac{d}{dt}\{&m_b[(\dot{X}_I c\theta + \dot{Y}_I s\theta + \dot{\theta} d_2)c\theta + (\dot{X}_I s\theta - \dot{Y}_I c\theta + \dot{\theta} d_1)s\theta] + \\
&(m_w + \frac{I_w}{r_w^2})\{[(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I - (L+l)\dot{\theta}](c\theta + s\theta) + \\
&[(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I + (L+l)\dot{\theta}](c\theta - s\theta) + \\
&[(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I - (L+l)\dot{\theta}](c\theta - s\theta) + \\
&[(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I + (L+l)\dot{\theta}](c\theta + s\theta)\}\} \\
= [m_b + 4(m_w + \frac{I_w}{r_w^2})]&\ddot{X}_I + m_b(d_2 c\theta + d_1 s\theta)\ddot{\theta} + m_b\dot{\theta}^2[-d_2 s\theta + d_1 c\theta]
\end{aligned}$$

Term (2) in (B.16)

$$\frac{\partial L}{\partial X_I} = 0$$

$$[m_b + 4(m_w + \frac{I_w}{r_w^2})]\ddot{X}_I + m_b(d_2 c\theta + d_1 s\theta)\ddot{\theta} + m_b\dot{\theta}^2[-d_2 s\theta + d_1 c\theta] = F_X \tag{B.17}$$

Finally, repeating the same derivation for the force on F_{IY}:

$$\underbrace{\frac{d}{dt}(\frac{\partial L}{\partial \dot{Y}_I})}_{(1)} - \underbrace{\frac{\partial L}{\partial Y_I}}_{(2)} = F_y \tag{B.18}$$

(1)

$$\begin{aligned}\frac{d}{dt}(\frac{\partial L}{\partial \dot{Y}_I}) &= \frac{d}{dt}\{m_b[(\dot{X}_I c\theta + \dot{Y}_I s\theta + \dot{\theta}d_2)s\theta + (-\dot{X}_I s\theta + \dot{Y}_I c\theta - \dot{\theta}d_1)c\theta] + \\ &\quad (m_w + \frac{I_w}{r_w^2})\{[(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I - (L+l)\dot{\theta}](s\theta - c\theta) + \\ &\quad [(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I + (L+l)\dot{\theta}](s\theta + c\theta) + \\ &\quad [(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I - (L+l)\dot{\theta}](s\theta + c\theta) + \\ &\quad [(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I + (L+l)\dot{\theta}](s\theta - c\theta)\}\} \\ &= [m_b + 4(m_w + \frac{I_w}{r_w^2})]\ddot{Y}_I + m_b(d_2 s\theta - d_1 c\theta)\ddot{\theta} + m_b\dot{\theta}^2[d_2 c\theta + d_1 s\theta]\end{aligned}$$

(2)

$$\frac{\partial L}{\partial Y_I} = 0$$

$$[m_b + 4(m_w + \frac{I_w}{r_w^2})]\ddot{Y}_I + m_b(d_2 s\theta - d_1 c\theta)\ddot{\theta} + m_b\dot{\theta}^2[-d_2 s\theta + d_1 c\theta] = F_y \tag{B.19}$$

For the torque on C.G. $F_{I\tau}$:

$$\underbrace{\frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}})}_{(1)} - \underbrace{\frac{\partial L}{\partial \theta}}_{(2)} = F_\tau \tag{B.20}$$

(1)

$$\begin{aligned}\frac{d}{dt}(\frac{\partial L}{\partial \dot{\theta}}) &= \frac{d}{dt}\{m_b[(\dot{X}_I c\theta + \dot{Y}_I s\theta + \dot{\theta}d_2)d_2 + (\dot{X}_I s\theta - \dot{Y}_I c\theta + \dot{\theta}d_1)d_1] + I_b\dot{\theta} \\ &\quad (m_w + \frac{I_w}{r_w^2})\{[(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I - (L+l)\dot{\theta}](-L-l) + \\ &\quad [(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I + (L+l)\dot{\theta}](L+l) + \\ &\quad [(c\theta - s\theta)\dot{X}_I + (s\theta + c\theta)\dot{Y}_I - (L+l)\dot{\theta}](-L-l) + \\ &\quad [(c\theta + s\theta)\dot{X}_I + (s\theta - c\theta)\dot{Y}_I + (L+l)\dot{\theta}](L+l)\}\} \\ &= [m_b + 4(m_w + \frac{I_w}{r_w^2})]\ddot{Y}_I + m_b(d_2 s\theta - d_1 c\theta)\ddot{\theta} + m_b\dot{\theta}^2[d_2 c\theta + d_1 s\theta]\end{aligned}$$

(2)

$$\frac{\partial L}{\partial \theta} = 0$$

$$[m_b + 4(m_w + \frac{I_w}{r_w^2})]\ddot{Y}_I + m_b(d_2 c\theta + d_1 s\theta)\ddot{\theta} + m_b\dot{\theta}^2[-d_2 s\theta + d_1 c\theta] = F_\tau \tag{B.21}$$

Combining (B.17), (B.19) and (B.21), results in,

$$\begin{bmatrix} m_b + 4(m_w + \frac{I_w}{r_w^2}) & 0 & m_b(d_2 c\theta + d_1 s\theta) \\ 0 & m_b + 4(m_w + \frac{I_w}{r_w^2}) & m_b(d_2 s\theta - d_1 c\theta) \\ m_b(d_2 c\theta + d_1 s\theta) & m_b(d_2 s\theta - d_1 c\theta) & (d_1^2 + d_2^2) + I_b + 4(m_w + \frac{I_w}{r_w^2})(L+l)^2 \end{bmatrix} \begin{bmatrix} \ddot{X}_I \\ \ddot{Y}_I \\ \ddot{\theta} \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & m_b\dot{\theta}(-d_2 s\theta + d_1 c\theta) \\ 0 & 0 & m_b\dot{\theta}(d_2 c\theta + d_1 s\theta) \\ m_b\dot{\theta}(d_1 c\theta - d_2 s\theta) & m_b\dot{\theta}(d_1 s\theta + d_2 c\theta) & 0 \end{bmatrix} \begin{bmatrix} \dot{X}_I \\ \dot{Y}_I \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} F_{X_I} \\ F_{Y_I} \\ F_\tau \end{bmatrix} \tag{B.22}$$

Ignoring the C.G. offset, (B.22) becomes,

$$\begin{bmatrix} m_b + 4(m_w + \frac{I_w}{r_w^2}) & 0 & 0 \\ 0 & m_b + 4(m_w + \frac{I_w}{r_w^2}) & 0 \\ 0 & 0 & I_b + 4(m_w + \frac{I_w}{r_w^2})(L+l)^2 \end{bmatrix} \begin{bmatrix} \ddot{X}_I \\ \ddot{Y}_I \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} F_{X_I} \\ F_{Y_I} \\ F_\tau \end{bmatrix} \tag{B.23}$$