Doctoral Dissertation

Academic Year 2019

Lidar-based Localization and Pose Estimation Approaches in the Map Built by Monocular SLAM

Su Wang

Division of Human Mechanical Systems and Design,
Graduate School of Engineering
Hokkaido University

Abstract of Doctoral Dissertation of Academic Year 2019

Lidar-based Localization and Pose Estimation Approaches in the Map Built by Monocular SLAM

Summary

Nowadays, the autonomous navigation of mobile robots and vehicles is a promising technology. For a mobile robot, self-localization in its work area is an essential requirement for practical applications. The Simultaneous Localization and Mapping (SLAM) is a widely used algorithm to reconstruct the environment and provide a map to other robots to perform self-localization and navigation. There is a large amount of SLAM solutions based on various types of sensors used for environmental perception. Among them, the monocular SLAM, in which only a single camera is employed, attracts plenty of interest in the robotic community because of its simple structure and low cost. However, since the distances between the camera and obstacles are not observable in a single image, the monocular SLAM suffers from an inherent problem that the estimated motion is determined without an absolute scale factor. Additionally, in case that there is no fixed baseline between two frames, the map reconstructed by monocular SLAM is also prone to be scale drifted over time. This natural drawback makes it very difficult for other types of sensors to reuse the map built by monocular SLAM, since the distance information is of key importance to the range sensors such as sonar and laser range finder. Nonetheless, it should be noticed that the geometrical structure of the obtained map is still available for performing localization. Using the geometrical features without scale information, it is still possible to match the perception from

range sensors and the map. This thesis mainly focuses on solving the localization problem against the map from monocular SLAM. The structure of this thesis is listed as follows:

- **Chapter 1:** The motivation of this study is demonstrated. Additionally, this chapter also gives a brief review of the technologies that are relevant to this study, including the localization methods, state-of-the-art monocular SLAM frameworks, and the solutions for dealing with the monocular scale drift.

- **Chapter 2:** A scale-aware Monte Carlo localization method. An extended Monte Carlo localization (MCL) is proposed to localize a robot with 2D laser scanner in the map from monocular SLAM. Through Monte Carlo sampling, the proposed method can estimate the pose of robot in the target environment as well as the local scale factor of the map. The proposed method can also achieve global localization on the real time with the help of Kullback-Leibler Divergence (KLD) sampling. The performance of the method is first tested on a public dataset to evaluate the accuracy and then carried out on a real robot in practical environment.

- **Chapter 3:** A novel scan registration algorithm with stretching the scale is presented. Although MCL can estimate the robot pose efficiently, it is expensive in computational cost and therefore can hardly solve the 6DoF pose estimation. Scan registration is another widely utilized technique which can directly estimate the relative pose of the sensor by matching the scan and reference point cloud. In this chapter, a novel point scan registration method named Scaled Normal Distribution Transform (sNDT) is proposed to handle the alignment between two-point clouds with different size. Both the run-time and convergence performance of sNDT are analyzed by comparing with

the ICP-based solutions. Furthermore, this study proposes to combine a UKF (unscented Kalman filter)-based back-end with the sNDT to verify whether it is able to handle the 6D Lidar-based localization in an incomplete and scale-drifted map from the monocular SLAM.

– **Chapter 4:** Conclusions are presented in this chapter. The defects and further improvement of this study are pointed out as well.

Keywords:

Localization, Laser Range Finder, SLAM

Division of Human Mechanical Systems and Design,
Graduate School of Engineering

Su Wang

Acknowledgements

I would like to express my deep gratitude to Prof. Yukinori Kobayashi for providing me the chance to study in his laboratory. With his priceless criticisms and encouragement, I was able to short many detours during the three years. Also, I am grateful for what I have learned from him, including not only the way of thinking but also the attitude towards life I could not imagine having a better during these years in Hokkaido University.

Besides my supervisor, I would like to show my sincere appreciation to Prof Takanori Emaru for his guidance. He kept cconcerning about my life in Hokkaido University and has always been ready to offer supports to me. And I also would like to extend my thanks to Prof. Ankit A. Ravankar and Prof. Abhijeet Ravankar from Kitami Institute of Technology, who taught me a lot of basic concepts about my research topic and helped me to fulfil my work.

I thank all the members of the Laboratory of Robotics and Dynamics, especially Jinxin Lv and Shuhei Yoshida. They gave me help and suggestions on both the daily life and research. I'm also grateful for the time spent with my friends, which made my life in Sapporo colorful.

My sincere thanks also goes to China Scholarship Council (CSC), who have economically supported me for enabling me to accomplish my doctoral course.

Finally, I offer my best regards to my parents for all their love and unfailing support throughout my whole life. They give me a happy and harmonious family,

otherwise I could hardly get any accomplisment.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Research Background

In recent years, benefiting from the rapid improvement of computer science, the full autonomous navigation becomes an increasingly viable option for controlling mobile robots and vehicles. Like the way people planing their own paths while walking, a precondition of the effect self-navigation of robots is the capability of estimating its actual position with respect to the external environment. Therefore, the self-localization problem frequently regards a map as the prior knowledge.

More often than not, this prior map used for localization is generated through a simultaneous localization and mapping (SLAM) process, in which the robot builds the map of the unknown environment based on the information from sensors while estimating its pose at the same time. Various sensors, including optical cameras [1, 2], ultrasonic wave sensors [6], laser range finders [7–9], or multiple sensors, can be used in SLAM systems depending on different applications and environments.

In the past years, vision-based SLAM approaches become more and more popular. The monocular SLAM, in which only a single camera is used, has been proven to be a desirable solution for building high quality maps with swift robots due to its simple hardware structure. However, monocular vision is not able to provide an absolute scale estimation of the landmarks. This leads to a problem that

when performing monocular SLAM, the movement of the camera has to be fixed on an ambiguous scale factor. Furthermore, without a stable baseline between adjacent frames, this scale factor is also unstable and unreliable. Accordingly, the map built by monocular SLAM will suffer from metric inconsistency, which means that the scale of the map obtained by monocular SLAM is unobservable and gradually drifts over time.

With the help of a wealth of information provided by images, it is possible to perform localization in the map generated by monocular SLAM using another camera by leveraging the semantic features in the images. However, considering a typical multi-robot system in robot search and rescue applications, as described in [10], it is reasonable to send a swift "mapper" robot with a single camera first to give an overview of the target environment and build a map through monocular SLAM at the same time since the environment to be explored is always unknown and complicated. Then, a "follower" unmanned ground vehicle (UGV), which has the load carrying capacity and often uses LRF for environment perception, can be deployed to the same area. By localizing themselves on the map provided by the "mapper" robot, the "follower" robots can carry out rescue activities more efficiently. It should be noted, in this scheme, that the "mapper" is only able to provide a map without the absolute scale. Hence, for the "follower" who wants to reuse this metric inconsistent map with only a range sensor, the unknown scale of the map could be a fatal drawback because the map without reliable range information is meaningless to the LRF and prevents the acquired sensor data from correctly associating with the map.

Since the correction of scale drift is one of the fundamental problems in monocular SLAM, there is a large amount of works presented for dealing with the scale drift or estimating the absolute scale of the robot motion. Some of these previous works have provided constructive ideas and achieved ideal performances, a brief

review of these methods can be found in Section 1.2.1. However, it can be found that the goal of all of these techniques is to fix the scale drift in the map building process. In the present work, this problem is considered in another perspective: if a map has already been built by monocular SLAM, and the metrical inconsistency of a map is irreversible, is it possible to directly localize another robot in it with only an LRF while estimating the local scale at the same time? In the presence of the range information from LRF, new constraints can be introduced into the scale-drifted map, which makes the absolute scale of the map observable.

1.2 Related Works

The main purpose of this dissertation is to solve the problem of localizing robot on the map constructed by a monocular SLAM process. In this section, an introduction of both the monocular SLAM method and the related Lidar localization approaches is made.

1.2.1 Monocular Simultaneous Localization and Mapping

State-of-the-art Monocular SLAM Solutions

The term "SLAM" refers to the technique that allows a robot to built the map of the environment around it and estimate its position in the map at the same time. The SLAM problem was firstly proposed in the landmark paper by Smith Self and Cheseman [11]. In the recent year, the visual SLAM is increasingly becoming one of the hottest topics in the robotic community. Some similar early works are presented as structure from motion (SfM) in the computer vision field [12], which is usually conducted off-line owing to the limitation of computing power. As the development of the computer performance, more and more real-time visual

SLAM and visual odometry solutions have been proposed. The visual SLAM can estimate the relative pose of camera in an unknown environment according to the captured images, and establish the map of the target environment. The Figure 1.1 shows an example of the monocular visual SLAM, which is also the study emphasis of this thesis. Assume that the robot moves with a monocular camera in an unknown environment. For convenience, the continuous motion of the camera is discretized into discrete moments $t_1, t_2 ... t_n$. The environment can be represented by a number of landmarks, which are visually identifiable objects in the real world. In every image, the positions of the landmarks is identified, and then the motion can be estimated based on these observed landmarks.

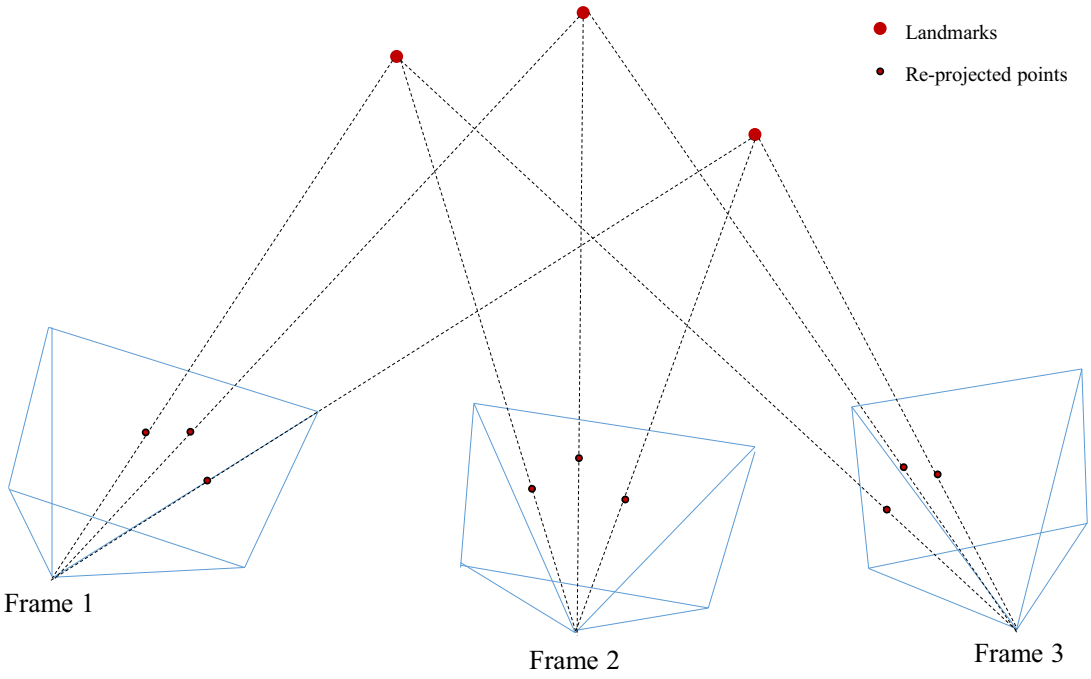


Figure 1.1: A sketch map of the monocular SLAM system.

The monocular SLAM solutions can be classified into several types according to their principles:

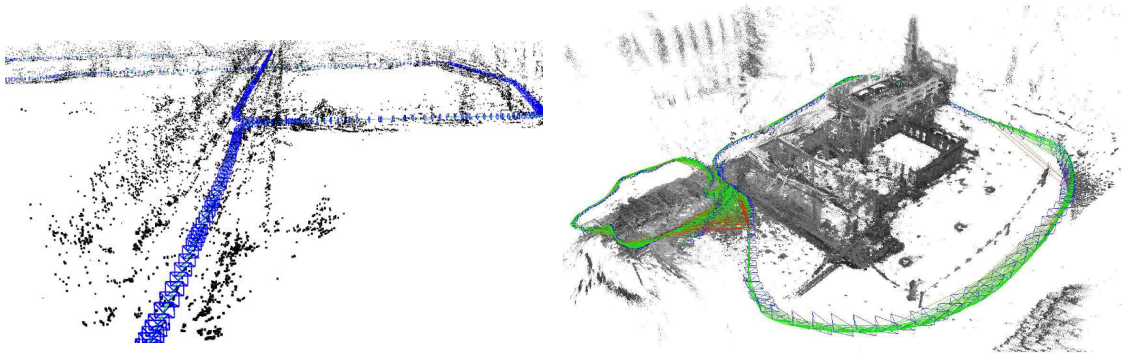
- **The filter-based monocular SLAM.** In the early development of SLAM technique, filter-based methods are most widely utilized due to its low computational complexity. The monoSLAM developed by Davison et al. [13] is the first real-time monocular solution for visual SLAM. The state vector in monoSLAM consists of the camera pose C_t at time t and the position of all the 3D landmarks in the environment. Each landmark also carries a corresponding error with probability, which can be depicted by a ellipsoid. The extended Kalman filter has been employed to handle the uncertainty of the camera state. However, maintaining all the map features consumes a lot of computing power. In order to reduce the computational complexity of EKF, MSCKF [14] has been proposed, in which only the most recent camera states are kept in the state vector and the 3D features are also marginalized during the SLAM process.

- **Key frame-based monocular SLAM.** PTAM proposed by Klein et al. [15] is an open sourced algorithm that first realizes the key frame bundle adjustment (key frame BA). The basic idea of PTAM is to treat the mapping and camera tracking as two individual task and executed them in two parallel threads. The mapping thread only needs to maintain the sparse key frames and the 3D point clouds. Therefore the BA can be solved more effectively. In ORB-SLAM [16] [1], the feature has been changed to the ORB feature, which makes the system more robust. Additionally, the loop closure has also been done based on the bag of words technique [17].

- **Direct tracking visual odometry.** The direct odometry gets rid of the pre-computation step of image feature and directly utilize the light information from sensors at a certain direction. The direct method optimize the photometric error instead of the reprojection error in order to minimize

the geometric error of camera motion. The examples of the direct SLAM methods include DTAM [18], LSD-SLAM [2], and DSO [19].

The main focus of this study is not on the monocular SLAM technique but on developing an efficient localization method that reuses the map built by monocular SLAM. Hence ready-made monocular SLAM methods, ORB-SLAM2 [1] and LSD-SLAM [2] as shown in the Figure 1.2, are employed only for providing the map.

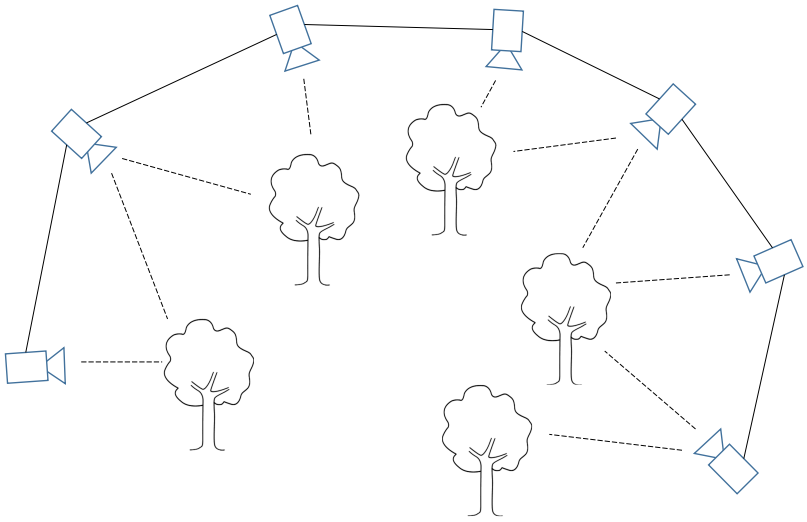


(a) (b)

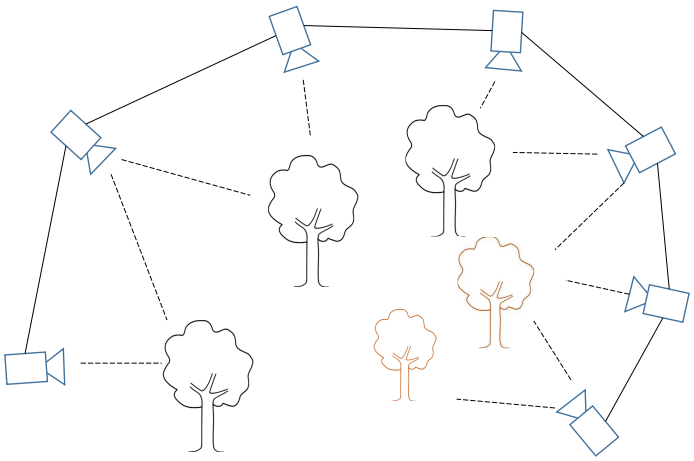
Figure 1.2: The monocular SLAM methods used in this study for mapping: (a)ORB-SLAM2 [1] and (b) LSD-SLAM [2]

Scale Correction Approaches

An essential drawback of the monocular SLAM system is that the monocular vision suffers from the scale-drift between frames. Without any prior knowledge, both the reconstructed 3D features and the motion will be drifted over time. Figure 1.3 depicts how the scale drift happens during the monocular SLAM.



(a) The camera trajectory with correct scale.



(b) The scale-drifted trajectory obtained by a monocular SLAM

Figure 1.3: The mechanism of scale drift phenomenon in monocular SLAM. The trees represent the landmarks observed by the camera. If the size of landmark can always been correctly estimated as shown in (a), the camera motion will be estimated with the correct scale. However, with only a single camera, there exists an inevitably accumulated error in estimating the scale of motion. In Figure (b), the motion between frames may drift to a smaller value since the size of landmark (yellow trees) can not provide a fixed measurement.

Recently, various methods have been proposed to solve the scale-drift problem in monocular SLAM. The most intuitive way to tackle the scale-drift problem is by fusing other types of sensors to make the metrical information in visual SLAM observable, e.g., the laser distance meter [20–22], IMU [23], or stereo system [24]. However, the most remarkable advantage of monocular SLAM is its portability. On the other hand, using additional sensors always means less compact, less power-saving and higher cost as compared to the one based on a pure monocular camera. In order to reduce the influence of the non-Gaussian distribution of the estimated depth, Civera et al. [25] proposed the inverse depth parameterisation for the distance between the camera center and the landmark. Detecting the loop closure and drift scale recovery by using global bundle optimization [26] is also a well-studied approach for correcting scale drift. Moreover, Strasdat [27] has also proposed the 7DoF SLAM, in which the similarity transformation is taken into consideration as a value to be estimate. The scale drift can be efficiently controled by updating this extra dimension. However, the problem is that the closed loop is not always permitted, and even if the loop can be successfully closed, the absolute scale is still unknown. In [28], the fixed height of the monocular camera is leveraged to impose the restrictions for scale correcting. Similarly, as proposed in [29, 30], using the object recognition and size estimation of the items, the absolute scale can also be recovered during monocular SLAM.

1.2.2 Lidar-based Localization Methods

Lidar is a typical range sensor which is able to directly measure the distance between the laser emitter and the obstacle in front of it by transmitting laser scans. This characteristic can ensure the reliability of the Lidar measurements in large-scale complex environment. Most of the localization methods are enhenced

by the Baysian filter, including Kalman filter-based methods [31] [32], particle filter [33]. The sensor model for the state correction can be just a ray-beam model [33] or the scan registration [34]. Most of the localization methods are also assisted by other type of odometries in order to enhence the robustness. The wheel encoder is the simplest odometry and widely equipped on most of the mobile robots. The wheel encoder is used to give an estimate of the displacement of the wheel surface with respect to the ground plane [35]. Though the wheel encoder is a very simple and low-cost motion estiamtion method, as a linear positioning device, it is influenced by the relative drift owing to the wheel slippage [36], hence it can hardly been utilized in the long-term pose estimation. In this work, the wheel odometry has been utilized for motion prediction.

This study is aiming to realize Lidar-based localization on an incomplete map. A few works have been developed to address and solve the similar problem. Koenig et al. [37] proposed an unsupervised method that learns the distance in a topological map built with errors. Kmmerle et al. [38] created a SLAM framework utilizing aerial view images as prior information, in which the Monte Carlo localization (MCL) method has been employed to obtain the global constraints in the map extracted from aerial images. Mielle et al. [39] presented a graph-based localization method in an emergency map. They tackled the deformation of the emergency map by optimizing the shape of the matched graph. Sketch map drawn by hand is another typical metrically inconsistent map. Boniardi et al. [40, 41] converted a hand-drawn draft into pixel coordinate and utilized a local scale factor to approximate the deformation of the draft.

Chapter 2
Scale-aware 2D Lidar localization

2.1 Introduction

In this chapter, we aim at designing an LRF-based method for localizing a robot in the map built from monocular SLAM. The application scenario of the proposed method is the multi-robot cooperation in a single-floor indoor space. The map obtained through the monocular SLAM process or visual odometry is a 3D point cloud, while the sensor used for localization is a 2D LRF. Therefore, the dimensionality of the point cloud is reduced by projecting it onto a 2D grid map to make the landmarks match the 2D scans. Then, to locate a robot in this map with scale ambiguity, the relative motion of the robot and the measurement of LRF are adjusted based on a flexible scale factor in order to make them adaptable to the size of the map. For achieving this goal, a variant of the MCL approach is presented to estimate not only the robot pose, but also the relative scale of the local map.

2.2 System Overview

The outline of the proposed system is shown in Fig. 2.1. The target environment is explored by a "mapper" agent, which can be a swift robot (UAV) or even a

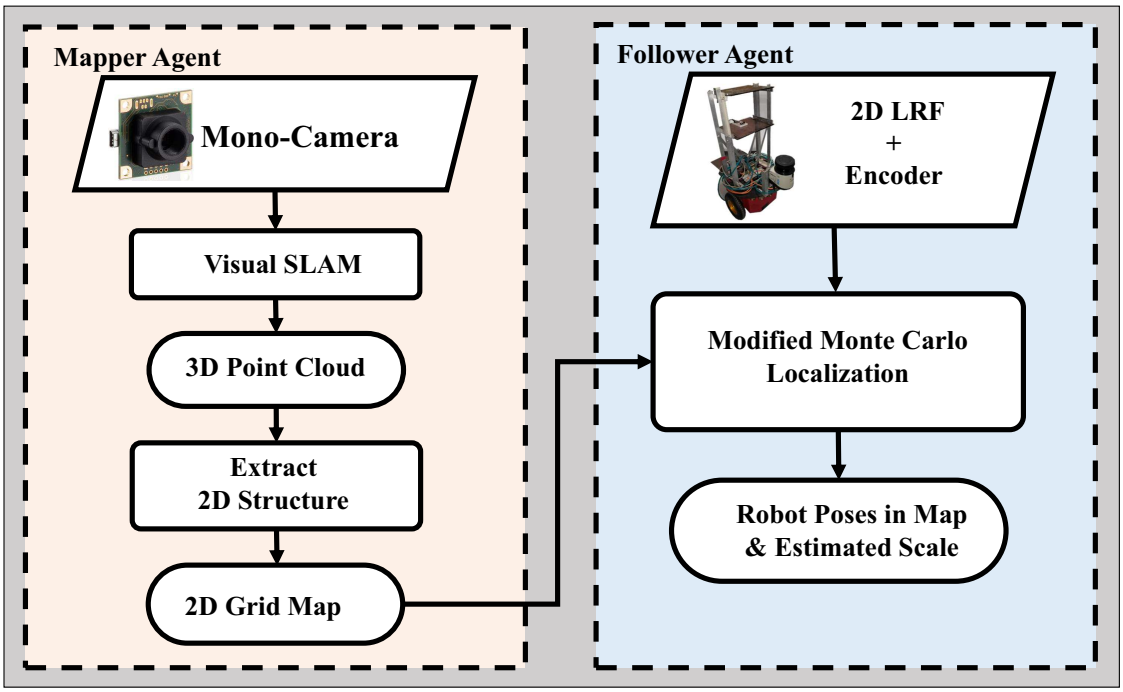


Figure 2.1: An overview of the proposed system, which is mainly composed of a "mapper" and a "follower" agent.

hand-held monocular camera. First, using the monocular SLAM, a 3D point cloud map is generated from the incoming image data and then pre-processed into a 2D representation. Next, a "follower" wheeled robot with LRF is deployed into the same area to perform localization in the constructed map. The LRF data, encoder data, and the map are inputs for the localization system. Whereas, the outputs are the local scale factor and the "follower" robot's pose.

The idea behind our approach is based on the assumption that the scale drift in monocular SLAM is a gradual process, therefore, the local deformation of the built map is always insignificant and equal in every direction. In a local map of visual SLAM, a point set $\mathcal{P}$ is maintained to depict the landmarks $\mathcal{L}$ in the real-word. For the agent "mapper", the absolute distance d between a landmark

$l_i \in \mathcal{L}$ and the camera center O_c is unobservable. Instead, the monocular SLAM produces an arbitrary distance parameter d_e between the corresponding point $p_i \in \mathcal{P}$ and O_c to give an estimate of d in the obtained map. For laser-based localization, the "follower" is sent to this certain local map, and it measures the same landmark l_i with LRF from another perspective. The distance between the robot and landmark l_i in the map is modeled as d_m. The actual distance, which is measured by the laser beam, is denoted as z_i. Under the previous assumption, the scale factor in the same local map should be invariable. Therefore, the local scale factor s_t of the robot state at time t is obtained as:

$$s_t = \frac{d_e}{d} = \frac{d_m}{z_i} + v_s \tag{2.1}$$

where v_s is the noise of scale estimation. In Equation (2.1), d is unknown, while d_e is given by the monocular SLAM and z_i is the observation from the laser sensor. Hence, for a pose $\boldsymbol{x}_t$ of the "follower" robot in the map frame, there exists a relative distance d_m and its corresponding scale factor s_t, which describes the relationship between the drifted scale and the absolute one. Therefore, it is possible to search for the robot poses and local scale using the Monte Carlo method.

In this work, our main focus is on the development of a localization approach rather than addressing the problem of visual SLAM. The maps to be used in our system are extracted from the one built by the monocular SLAM method beforehand. In theory, the proposed localization algorithm is not confined to a specific SLAM method and works well on the point cloud maps generated by any monocular SLAM systems. For instance, in Section 2.5, our approach is experimentally evaluated on different types of 3D maps (semi-dense and sparse) built by various SLAM methods [1, 2].

2.3 2D Representation of the 3D Environment

In this section, our goal is to convert the 3D point cloud map into a 2D map in order to match the scans from 2D LRF; 2D structure extraction from 3D map is a technique commonly used for 2D–3D sensor fusion and data compression. Korah [42] presented a vertical structure representation method named the Strip Histogrim Grid, in which the scene is encoded as histograms and the objective identification is also enabled. A classifier named HLD is leveraged in [43]. It parameterizes the obstacles and classifies them into positive and negative ones in order to provide information for self-navigation. An improvement of the HLD (Height-length-density) classifer is given by Goeddel et al. [44], in which they assorted the obstacles based on the slope and developed a compact map structure. Other works proposed probability-based methods to handle the rasterization of 3D map. For instance, in [45, 46], the Gaussian Mixture model is exploited to characterize the large-scale point clouds into a 2D set to achieve fast LiDAR localization.

For achieving the 2D representation, we employed the 2D occupancy grid map [47] where each location in the environment is assumed to be either occupied, free, or unknown. Here, the lines of sight from the camera center to the landmarks are considered as measurements. Every key frame from monocular SLAM is treated as an independent observation. Next, all the measurements from the camera are projected to a specified plane and the occupied probability of each grid is calculated using the approach employed in the laser-based mapping method [48–50].

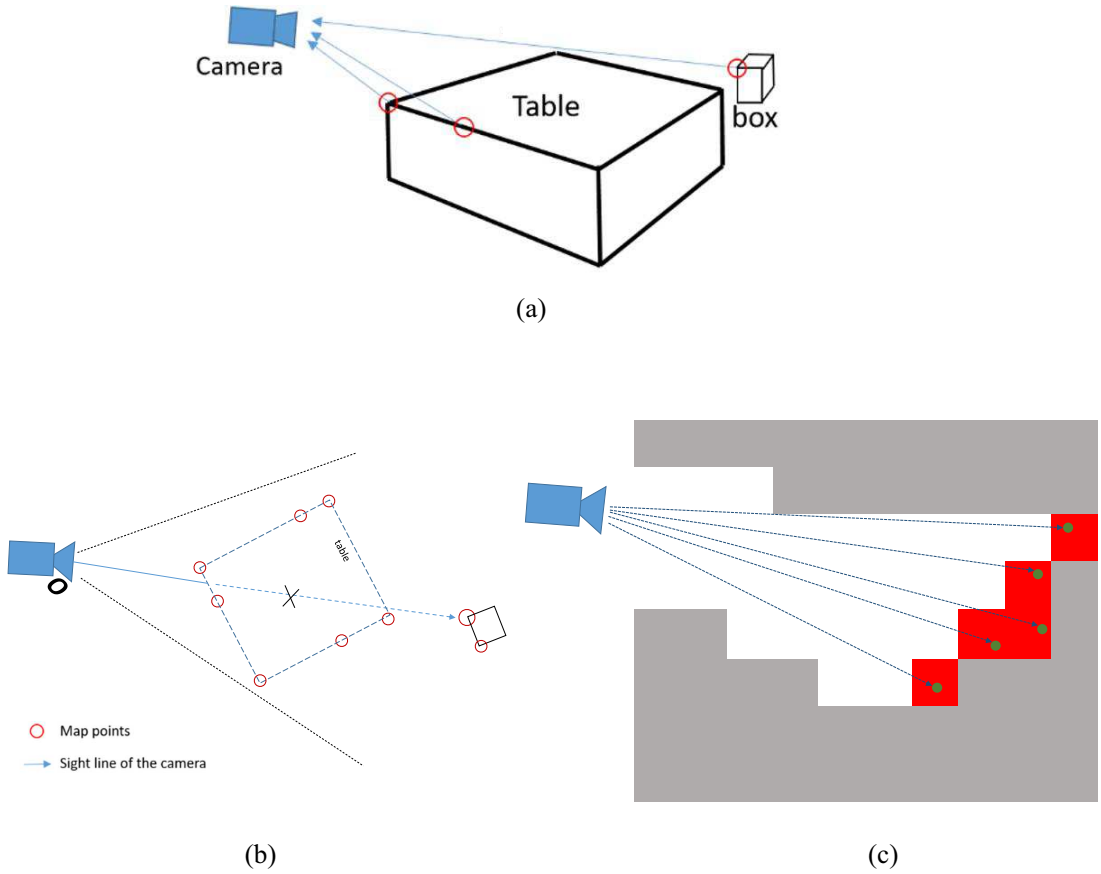


Figure 2.2: (**a**) (**b**) the 3D features in a single camera frame. (**c**) 2D grid map extracted from the single frame.

Since the absolute pose of camera is unknown, it is necessary to decide the projection direction. For example, most indoor environment consists of flat floor surface that can be used to determine the coordinate system. To obtain the floor plane, RANSAC (Random Sample Consensus)-based plane detection is applied to the point cloud map to extract the plane coefficients. However, the spurious planes that snatch parts of the points from the other items may lead to misidentification [51]. For instance, points of the planes segmented by the original RANSAC in the indoor environment frequently belong to parts of the edges of tables or chairs

on the floor. The original RANSAC plane detection method is slightly modified by splitting all the 3D map points $\mathcal{P}$ with the hypothetical plane. We assume that the y-axis of the first camera frame, which is always defined as the y-axis of the world frame, should point to the ground. Since the points under the hypothetical plane can be considered as "underground" points that should not be observed by camera, the hypothesis plane with the maximum probabilistic $\widehat{M}$ in the modified RANSAC is denoted as follows:

$$\widehat{M} = argmax_M[\sum_{p_i \in \mathcal{P}} \Gamma_{in}(p_i, M)], \tag{2.2}$$

where Γ_{in} is the indicator of the plane model:

$$\Gamma_{in}(p_i, M) = \begin{cases} 1, & \mid d_i \mid \leq d_t \\ -\mu, & \mid d_i \mid > d_t \quad and \quad F_M(p_i) > 0 \\ 0, & otherwise \end{cases}, \tag{2.3}$$

where $F_M(x)$ is the plane function of the hypothesis plane M . If the point-plane distance d_i between the plane and point p_i is within a threshold d_t, $\Gamma_{in}(p_i, M)$ will return 1; if p_i is an "underground" point below the hypothesis plane, Γ_{in} will return a negetive value $-\mu$. The parameter $-\mu$ is the weighting coefficient of the points under the hypothesis plane, which needs to be adjusted depending on different densities of the maps. In other words, the idea is to detect a plane with fewest points under it. Fig. 2.3 (a) provides an example of the point cloud from monocular SLAM on the EuRoC MAV dataset [3]. As shown in Fig. 2.3 (b), the blue points are the inlier ones that belong to the floor plane extracted from the point cloud, and the "underground" points, which make a negative contribution to the weight of the detected plane, are marked in red.

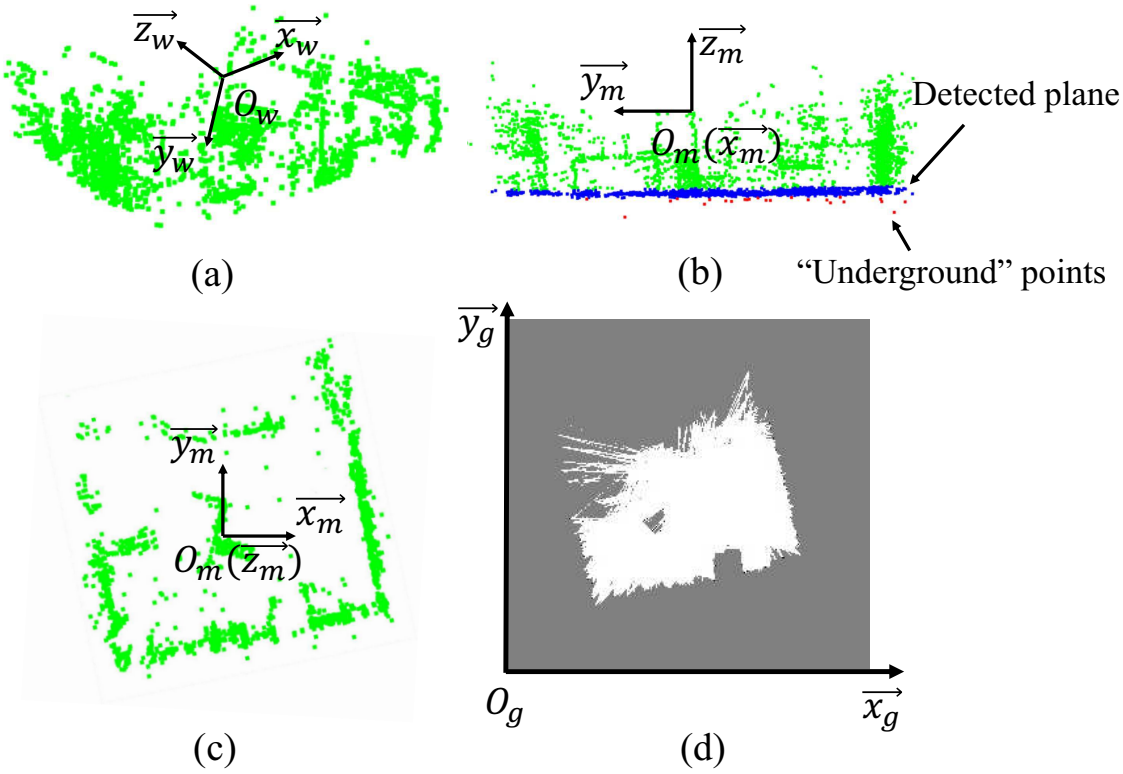


Figure 2.3: An example of 2D structure extraction on the EuRoC dataset [3]: (**a**) the raw point cloud built by monocular SLAM on the Euroc dataset, (**b**) floor plane detected by RANSAC and the new map frame O_m, (**c**) top view of the point cloud map, (**d**) extracted 2D grid map. The black, gray and white pixels represent the obstacle, unknown, and free grids, respectively.

The world frame of monocular SLAM is denoted as $\{O_w, \overrightarrow{x_w}, \overrightarrow{y_w}, \overrightarrow{z_w}\}$, where $\overrightarrow{x_w}, \overrightarrow{y_w}, \overrightarrow{z_w}$ are base vectors of the world frame that is centred at O_w. Let $\overrightarrow{n}$ denote the normal vector of the plane obtained by RANSAC, which satisfies $\overrightarrow{n} \cdot \overrightarrow{y_w} > 0$. Thus, the new map frame before projection can be defined as $\{O_m, \overrightarrow{x_m}, \overrightarrow{y_m}, \overrightarrow{z_m}\}$, in which the unit vectors $\overrightarrow{z_m}$ is obtained by projecting $\overrightarrow{z_w}$ onto the detected plane $\widehat{M}$; $\overrightarrow{z_m} = -\overrightarrow{n}$; $\overrightarrow{x_m}$ is the vector which is perpendicular to $\overrightarrow{y_m}$ and $\overrightarrow{z_m}$; O_m coincides with the point O_w. Therefore, the transformation matrix from the world frame to

the map frame is ${}^{m}T_{w} = [\overrightarrow{x_m}, \overrightarrow{y_m}, \overrightarrow{z_m}]^T$.

If the resolution of the 2D grid map is $(X_{max} \times Y_{max})$, and r is the size of each grid (m/grid), the transformation from a world point $\mathcal{P} = \{x_p, y_p, z_p\}$ to its corresponding 2D map grid $G = \{x_g, y_g\}$ is given as:

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & 0 & 0 \\ 0 & \frac{1}{r} & 0 \end{bmatrix} \begin{bmatrix} \overrightarrow{x_m} \\ \overrightarrow{y_m} \\ \overrightarrow{z_m} \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} + \begin{bmatrix} X_{max}/2 \\ Y_{max}/2 \end{bmatrix}, \tag{2.4}$$

Consequently, the 3D map can be processed into a 2D grid map by calculating the occupancy probability of each grid. Furthermore, sometimes the camera FOV (field of vision) may go through spaces above the obstacles and therefore will not be able to observe them due to the limitation of the field of view. As a result, these unobserved landmarks may be regarded as paths which are free to pass. To get rid of this kind of misjudgement, a grid is considered as the constant obstacle if its occupied probability has already reached a threshold for a number of frames.

The top view of the point cloud built on Euroc dataset can be seen in Fig. 2.3 (c). The extracted 2D grid map is shown in Fig. 2.3 (d). Some of the landmarks and the shape of the target environment has been well described by the grid map. However, the left-top side of the grid map in Fig. 2.3 (d) is improperly marked as "unknown". This is because the source map is generated by a sparse SLAM method [1], and the FOV of the camera in this dataset rarely focuses on this area.

2.4 2D Laser-Based Localization in Scale Drifted Map

2.4.1 Extended Monte Carlo Localization

An extended MCL algorithm is employed to solve the LRF based localization problem. The extended MCL algorithm is built upon the conventional one proposed by Dellaert et al. [33, 52] to make an extra estimation of the local scale

factor s_t at time t. We assume that s_t only depends upon the position $\boldsymbol{x}_t$ and the map m, the state of the robot can be updated by:

$$p(\boldsymbol{x}_t, s_t \mid \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}, m) \propto p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, s_t, m) \cdot \int_{s_{t-1}} \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_{t-1}, s_{t-1} \mid \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t-1}, m) \cdot p(\boldsymbol{x}_t, s_t \mid \boldsymbol{x}_{t-1}, s_{t-1}, \boldsymbol{u}_t) d\boldsymbol{x}_{t-1} ds_{t-1}, \tag{2.5}$$

where, $\boldsymbol{x}_t$ denotes the pose of the robot at time t in the map m, $\boldsymbol{z}_{1:t}$ is the sensor observations. Given the control inputs $\boldsymbol{u_{1:t}}$, the motion model $p(\boldsymbol{x}_t, s_t \mid \boldsymbol{x}_{t-1}, s_{t-1}, \boldsymbol{u}_t)$ predicts the probability of updating the robot state into $\{\boldsymbol{x}_t, s_t\}$. The sensor model, or observation model, $p(\boldsymbol{z}_t \mid \boldsymbol{x}_t, s_t, m)$ denotes the likelihood of receiving sensor signal z_t over the current state $\boldsymbol{x}_t, s_t$ in the map m.

In MCL, the probability of the robot states is randomly sampled and represented by a number of particles. Generally, the update of the belief is a three-step recursive process, also known as the particle filter.

As discussed in Section 2.2, a prerequisite for achieving robust localization is the correct scale estimation. As a result, although the scale factor s_t and robot pose $\boldsymbol{x}_t$ are calculated by particle filter at the same time, the convergence of s_t always precedes the robot pose $\boldsymbol{x}_t$. In this case, the convergence process of the MCL method can be divided into two phases based on the behaviours of particles: (1) Scale factor initialization: the scale factor s_t is first converged at a limited range. A parameter σ_c is defined to judge whether the scale estimation is well converged. (2) Robot pose tracking: the pose of robot will be calculated only if the scale factor s_t has already been converged, meanwhile, s_t will also be continuously revised. Here, σ_c is calculated by:

$$\sigma_c = \sum_{i=1}^{n_p} (\log s_t^{mean} - \log s_t^i)^2, \tag{2.6}$$

where, n_p is the total number of employed particles at time t, s_t^i is the scale factor of the i-th particle at time t, s_t^{mean} is the average scale factor of all the particles at time t. The scale factor will be considered as converged only if the σ_c is smaller than 0.8 for 5 updates continuously. This threshold value is determined empirically.

2.4.2 Motion Model

The task of the robot motion model is to compute the posterior distribution $p(\boldsymbol{x}_t, s_t \mid \boldsymbol{x}_{t-1}, s_{t-1}, \boldsymbol{u}_t)$. In our work, the odometry model based on the encoder will be used to predict the motion. Fig. 2.4 (a) illustrates the robot odometry model. When the robot advances from $(x_{t-1}, y_{t-1}, \theta_{t-1})$ to (x_t, y_t, θ_t), the relative motion can be indicated by a parameter vector $\boldsymbol{\delta_m} = \{\delta_{trans}, \delta_{rot1}, \delta_{rot2}\}$, in which the element δ_{trans} is the translation, δ_{rot1} and δ_{rot2} are the first and second rotation of robot head. Given the reading of encoder, δ_m can be computed as:

$$\begin{cases} \delta_{trans} = \sqrt{(\overline{x}_t - \overline{x}_{t-1})^2 + (\overline{y}_t - \overline{y}_{t-1})^2} \\ \delta_{rot1} = atan2(\overline{y}_t - \overline{y}_{t-1}, \overline{x}_t - \overline{x}_{t-1}) - \overline{\theta}_{t-1} \\ \delta_{rot2} = \overline{\theta}_t - \overline{\theta}_{t-1} - \delta_{rot1} \end{cases}, \tag{2.7}$$

where, $\{\overline{x}_t, \overline{y}_t, \overline{\theta}_t\}$ is updated by the raw measurement of the encoder. Through modelling the motion error under Gaussian distribution, the rotation and translation can be sampled by:

$$\begin{cases} \hat{\delta}_{trans} = \delta_{trans} - G(\alpha_3 \mid \delta_{trans} \mid + \alpha_4 \mid \delta_{rot1} + \delta_{rot2} \mid) \\ \hat{\delta}_{rot1} = \delta_{rot1} - G(\alpha_1 \mid \delta_{rot1} \mid + \alpha_2 \mid \delta_{trans} \mid) \\ \hat{\delta}_{rot2} = \delta_{rot2} - G(\alpha_1 \mid \delta_{rot2} \mid + \alpha_2 \mid \delta_{trans} \mid) \end{cases}, \tag{2.8}$$

where, the function $G(b)$ means Gaussian noise with variance b and zero mean. α_x denotes the motion error parameter of the robot, which specifies the error of robot state accrued with the pose changing, similar to the defination in [47].

Additionally, the scale factor s_t should also be updated with noise. It has been observed that one of the main reasons of the scale drift is the large rotations of camera. As a result, serious scale drifts frequently occur at the corners in the map. In this case, the rotation is taken into account for the prediction of s_t in the motion model. The $p(s_t \mid s_{t-1})$ is defined as a Gaussian distribution:

$$s_t \sim \mathcal{N}(s_{t-1}, (\sigma_s + \alpha_5(\delta_{rot2} + \delta_{rot1})/\pi)^2), \tag{2.9}$$

where $(\delta_{rot2} + \delta_{rot1})$ is the angle of robot's rotation between t and $t-1$, and σ_s is the standard deviation of variable s_t of the particles in the cluster with the highest weight.

Then the pose of robot can be updated by:

$$\begin{cases} x_{t+1} = x_t + s_t \cdot \hat{\delta}_{trans} \cdot \cos(\theta + \hat{\delta}_{rot1}) \\ y_{t+1} = y_t + s_t \cdot \hat{\delta}_{trans} \cdot \sin(\theta + \hat{\delta}_{rot1}) \\ \theta_{t+1} = \theta_t + \hat{\delta}_{rot1} + \hat{\delta}_{rot2} \end{cases}, \tag{2.10}$$

Fig. 2.4 (b),(c) depict the distributions of 1000 samples from the classical odometry model and the one with uncertain scale factor. In these two figures, the sampling has been done under the same parameters of motion model, however, it can be seen that our model is with higher uncertainty owing to the metrical inconsistencies. Therefore, the proposed localization method usually takes longer time to be converged in contrast to the original MCL method.

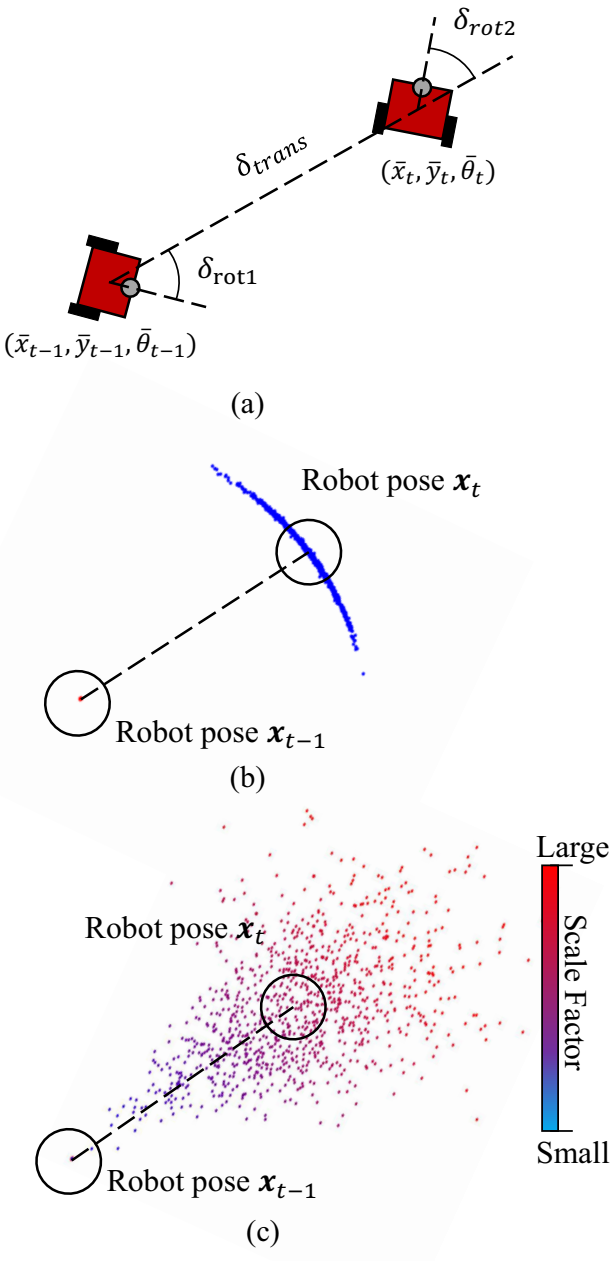


Figure 2.4: (**a**) motion model of the robot, (**b**) sampling 1000 particles from the classical motion model, (**c**) sampling 1000 particles from the motion model with scale uncertainty. The colors of the points are used to represent the specified scale factors. The darker color represents smaller scale factor s_t.

2.4.3 Sensor Model

The beam model, or the laser ray model [47], is utilized to calculate the importance of each particle. Intuitively, the working principle of beam model in a MCL algorithm is to fix the laser scanner onto the poses of all the particles in the map m. These hypothetical beams are cast along their current directions until hitting obstacles or reaching their maximum ranges. The weight of each particle will be determined based on the degree of agreement between these beams and the map m. Let the i-th measurements on map m be z_t^i. The likelihood of the full scan is composed of all the n_z beams:

$$p(z_t \mid \boldsymbol{x}_t, s_t, m) = \prod_{i=1}^{n_z} p(z_t^i \mid \boldsymbol{x}_t, s_t, m), \tag{2.11}$$

where, the model of a single beam incorporates four error types sourcing from four different conditions (correction detection, unexpected obstacles, missed measurements, and random measurements):

$$p(z_t^i \mid \boldsymbol{x}_t, s_t, m) = \begin{pmatrix} \phi_{hit} \\ \phi_{short} \\ \phi_{max} \\ \phi_{rand} \end{pmatrix}^T \cdot \begin{pmatrix} p_{hit}(s_t \cdot z_t^i \mid \boldsymbol{x}_t, m) \\ p_{short}(s_t \cdot z_t^i \mid \boldsymbol{x}_t, m) \\ p_{max}(s_t \cdot z_t^i \mid \boldsymbol{x}_t, m) \\ p_{rand}(s_t \cdot z_t^i \mid \boldsymbol{x}_t, m) \end{pmatrix}, \tag{2.12}$$

where $\phi_{hit}, \phi_{short}, \phi_{max}, \phi_{rand}$ are the weights defined for averaging the four types of distributions. The method to calculate $p_{hit}, p_{short}, p_{max}$ and p_{rand} is identical to the original beam model described in [47].

Since the map built by monocular SLAM cannot perfectly depict the target scene, some free space gets marked as obstacles by mistake. Therefore, the particles are allowed to travel through the unreachable parts in the grid map. However, this will cause a new problem that some particles may pass the "walls" in the map and increase the risk of localization failure. Therefore, the weights of the particles

in the unknown grids are penalized by multiplying a coefficient $\lambda_u = 0.9$. Similarly, the penalty coefficient for the particles in the occupied grids is $\lambda_o = 0.4$.

2.4.4 Initialization of Particle Filter

At the start of the MCL approach, the particles need to be set based on an initial guess of the robot's state. Since the size of the original map is unknown, the initial scale factors of all the particles are randomly set in a range obeying uniform distribution. For example, if the particles are initialized with scale factors in the range [0.01 m/grid, 3 m/grid], a grid map with the size 640 $\times$ 640 will be able to represent the environment with a dynamic size from 6.4 m $\times$ 6.4 m to 1920 m $\times$ 1920 m, which covers all the possible sizes of the map for a typical indoor environment.

Furthermore, if the robot has no knowledge about its starting position, it is necessary to estimate its pose in the entire map, which is known as the global localization problem. In this work, one of the key advantages of using MCL algorithm is the ability to handle the global localization by distributing particles over all the possible grids in the map. For the evaluations in Section 2.5, the method is tested with a manually set initial guess around the starting point, and also under a global distribution covering all the possible hypotheses states for global localization.

During the initialization process, a large number of particles are required for searching among all the possible scale factors. However, after a few steps, most of the particles with incorrect scales will be eliminated in the resampling step and the variance of s will also be decreased accordingly. Then, the performance of particle filter in the proposed method become similar to the original MCL, thus tracking of robot pose can be done with much fewer particles. To determine the number of particles, the KLD (Kullback-Leibler divergence) sampling approach [53] is

utilized, which gives an approximate estimation of the KL Divergence between the true distribution and the estimated one.

Denote p and q as two probability distributions, the Kullback-Leibler distance is defined as a measurement of the difference between two probabilities. Using the result presented in [54], the chi-square distribution can be approximated by the Wilson-Hilferty transformation:

$$\begin{aligned} n &= \frac{1}{2\epsilon}\chi^2_{k-1,1-\delta} \\ &= \frac{k-1}{2\epsilon}\left[1 - \frac{2}{9k-1} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right]^3, \end{aligned} \tag{2.13}$$

where, $z_{1-\delta}$ represents the upper quantile of $N(0,1)$ distribution. In the implementation, the space is divided by three dimension data ($\{x_t, y_t, \theta_t\}$). In the correction phase of the particle filter, if a number of particles are deployed at the same position $\{x_t, y_t, \theta_t\}$ on the map and observing the similar landmarks, their scale factors should be fixed based on the sensor perception. Therefore, s_t is not taken into account for adjusting the number of bins.

During the sampling step of particle filter, in order to determine the value k, the bins number is counted. If a new sample of the robot state is generated based on the motion model $p(x_t, s_t \mid x_{t-1}, s_{t-1}, u_{t-1}Bel(x_{t-1}, st-1))$, the KD tree checks whether a new leaf needs to be established or the new sample just falls into a non-empty one. The sampling of particles will be continued until the number of KD tree leaves reaches the number n.

2.4.5 The Problem of Particle Collapse

In the resample process of the particle filter, if only a small number of particles carries the vast majority of weights whereas the rest of the particles has the

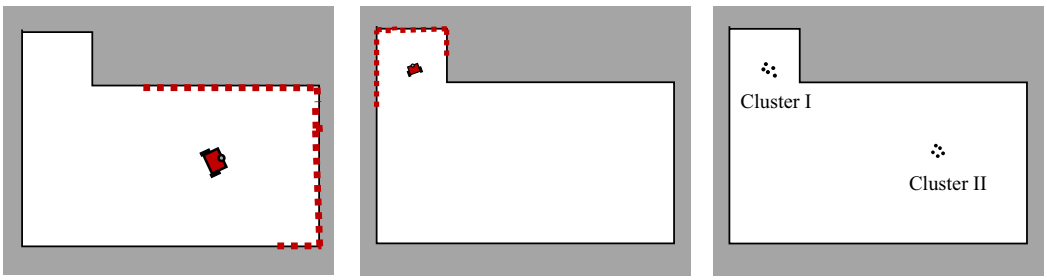


Figure 2.5: An example scenario where particle collapse may happen: (**a**,**b**) depict two different robot states which are about equally possible according to the sensor model. (**c**) shows two clusters of particles generated at the same moment, around the hypothesis states.

minimum weights, the diversity of particles will be reduced. This may lead to a situation where the algorithm may fall into a single peak, which is a common problem in the particle filter, known as the particle collapse. Normally, this can be solved by setting trigger conditions for the resampling and applying low variance sampling.

Nevertheless, in our case, the uncertain scale could also be an underlying cause of the particle collapse. For instance, Fig. 2.5 (a),(b) shows two possible robot states carrying different scale factors but observing similar laser measurements with different poses. With similar probabilities, two clusters of particles will be generated around these two positions as shown in Fig. 2.5 (c). Assume that the Cluster I in Fig. 2.5 (c) represents the correct robot pose. If Cluster I is eliminated after a few updates, all the resampled particles will be incorrectly gathered at the Cluster II, that is how the particle collapse happens.

In our practical tests, the proposed method is prone to fall into local optimal with very small scale factors because of the ambiguity of the map scale, especially in the global localization. To avoid this collapse problem, the candidate particles in a few sub-optimal clusters should be prevented from being discarded to make

the system more robust against the multi-optimal situations, similar to the idea in [55]. Then, the particles are clustered according to the distance in $\{x, y, \theta\}$ into several clusters and then sort them on their total weights. A number of clusters with the highest weight are selected for tracking multi-model hypotheses in the initialization step.

In the clustered MCL [55], several clusters of particles are updated independently for global trace, which is not required for our case. Instead, our strategy is to put a part of the particles in each selected cluster into a "safe box" and keep them from being abandoned during the resample operation. Moreover, particles in all the clusters are involved in the same MCL process, which means that these clusters are dynamically changed in every update. Whereas, this operation will introduce bias to the system owing to artificially increasing the probability of the states carried by the particles in the "safe box". Hence, this method is only applied until the scale is considered as converged.

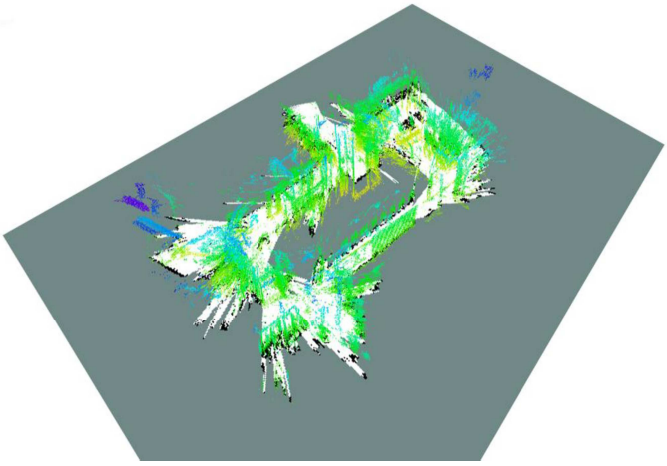
2.5 Experimental Evaluation

In this section, the performance of the proposed localization method is evaluated on two datasets. First, a group of tests is given on a public benchmark dataset, the MIT Stata dataset [4], in which data from both 2D LRF and camera is included. Then, the method is evaluated on our own devices in a real-world environment.

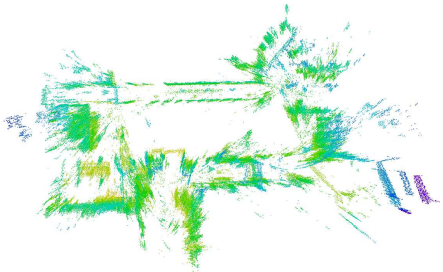
2.5.1 Results on Benchmark Dataset

Experiment Setup

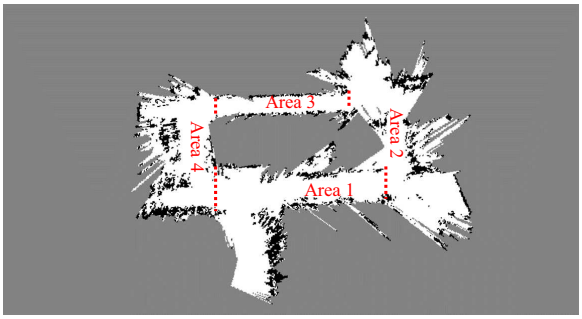
Three sequences, the *2011-01-19-07-49-38*, *2011-01-21-09-01-36*, and *2011-04-06-38-27* (referred as sequence *ms-38*, *ms-36*, *ms-27* henceforth, respectively) in the MIT Stata dataset [4] are used. Since these sequences are acquired in the same building, it is possible to simulate all the necessary tasks, including the monocular SLAM process and localization tests, with these sequences. First, the LSD-SLAM [2], a popular semi-dense monocular SLAM framework, is applied on the image data in *ms-38* to obtain a 3D map and its 2D-grid projection. Then, using the laser data in *ms-38*, *ms-36*, and *ms-27*, the proposed scale-aware localization approach is performed in the obtained map. The point cloud map built by monocular SLAM and its 2D projection is shown in Fig. 2.6. The map is divided into several areas and label them with different area numbers for convenient description.



(a)



(b)



(c)

Figure 2.6: Map of the target environment, built based on the sequence *ms-38* of the MIT Stata dataset [4]. (**a**) The point cloud map that is transformed to the map coordinate and its 2D projection. (**b**) The top view of the point cloud map. (**c**) The extracted 2D grid map.

In all these experiments, including the ones on the real dataset presented in the next subsection, the same set of parameters for the MCL method are used. We set $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\} = \{0.1, 0.5, 0.8, 0.5, 0.01\}$ for the motion model; and $\phi_{hit} = 0.5, \phi_{rand} = 0.1, \phi_{short} = \phi_{max} = 0.05$ for the observation model.

Evaluation of Pose Tracking

First, in order to test only the tracking and scale estimating performance, the global localization is not taken into account. Therefore, the rough location of the robot's starting point is considered as a known condition and the initial samples of the robot's pose will be generated in a wide area around its starting position, under a normal distribution. First, the proposed method is tested with the particle number 1000–3000. This interval indicates that the lower limit of the particle number for KLD sampling is set to 1000, and the maximum number of the employed particles is 3000, which is also the particle number in the first iteration of the particle filter.

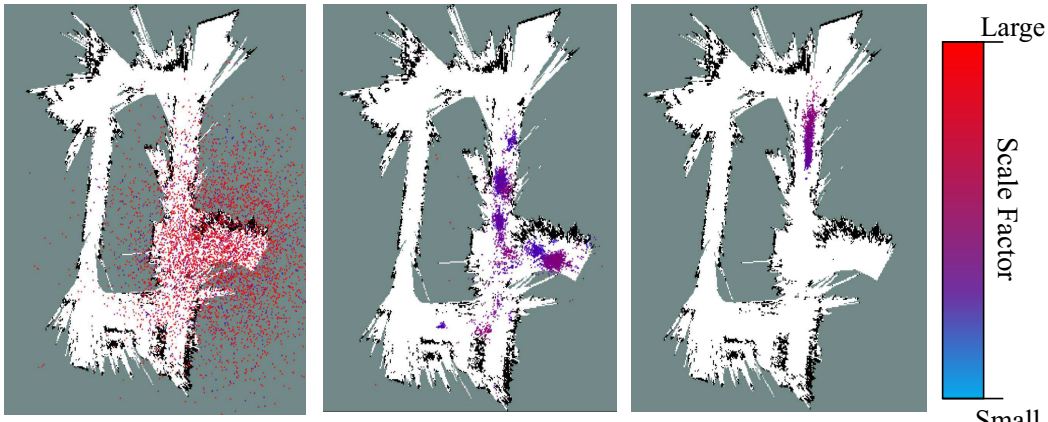
The MCL algorithm is a highly non-deterministic process. Therefore, we performed 5 runs for each sequence. The trajectories estimated by the proposed localization approach on the three sequences are plotted in Fig. 2.8. Here, a comparison between the 2D grid map from monocular SLAM and the one built by the LRF [56] with correct scale is provided. The ground truth path is shown in Fig. 2.8 (a)–(c) for comparison. The trajectories from Fig. 2.8 (d) through Fig. 2.8 (e) are recorded after the particle filter are converged in scale. Both the *ms-38* and *ms-36* start in the "Area 1" and end in the "Area 4" (the area division is referred to Fig. 2.6), while the active range of *ms-27* is within the "Area 4". It can be noticed that although the grid map is deformed in contrast to the LRF built map, the shapes of the trajectories still tally with the ground truth. A video of the localization process can be found in the supplementary section.

In the MIT Stata dataset, ground truth of mobile robot's poses in all the sequences is provided. It is difficult to directly evaluate our experimental result based on it because the ground truth is recorded in the real world coordinate, while our experiments are carried out in the deformed map. Therefore, we turn to validate the error of the estimated translation between $\{\boldsymbol{x_t}, s_t\}$ and $\{\boldsymbol{x_{t-1}}, s_{t-1}\}$, which is defined as:

$$e_{trans,t} = \frac{\boldsymbol{x}_t - \boldsymbol{x}_{t-1}}{s_t} - \boldsymbol{x}_t^g + \boldsymbol{x}_{t-1}^g, \tag{2.14}$$

where, $\boldsymbol{x}_t^g$ is the robot's ground truth pose at time t. Both the localization and scale estimation accuracy can be accessed by the translational error because once the localization fails or the scale factor is estimated incorrectly, abrupt changes will appear on $e_{trans,t}$. The translational error $e_{trans,t}$ on three sequences are plotted in Fig. 2.9. It can be seen that the error fluctuates within a narrow range. This shows that our localization method provides a reliable state estimation for the robot.

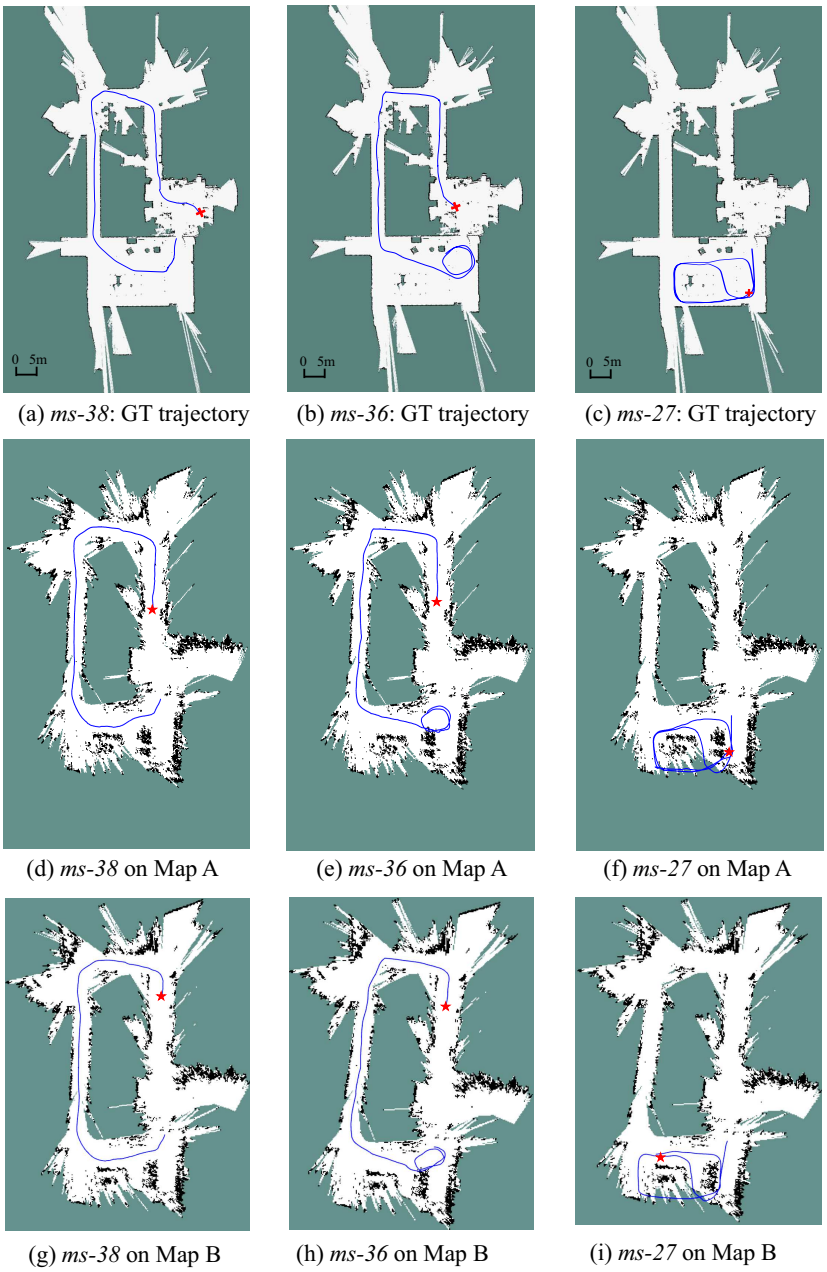
The scale estimation results of a group of runs on three sequences is shown in Fig. 2.10. Since *ms-38* and *ms-36* are acquired following the same path, Fig. 2.10 (a),(b) show a similar trend, starting from 0.12 m/grid and then drifting to a small value gradually. The *ms-38* and *ms-36* ends in the "Area 4", which illustrates that the scale factor in "Area 4" is around 0.08 m/grid. This is also in consistent with the result shown in Fig. 2.10 (c), in which the scale factor in "Area 4" is estimated to be within the range from 0.068 m/grid through 0.082 m/grid.



(a) 10k particles (b) 7.8k particles (c) 2.1k particles

Figure 2.7: The particle distribution in the initialization phase of the extended MCL.

The number of particles is one of the influencing factors of the MCL method. Hence, it is worth discussing the state estimation performance with different particle numbers. Another group of tests is performed with a larger particle number: 2000–10,000. A comparison between the translational errors in the tests with different particle numbers are depicted in Fig. 2.11. Some data of the localization results is summarized in Table 2.1. In this table, the translational error rate is calculated by $[\sum_{t=0}^{t_{end}} \|(e_{trans,t})/(\boldsymbol{x}_t^g - \boldsymbol{x}_{t-1}^g)\|]/n_{end}$, where t_{end} stands for the final update, and n_{end} denotes the the number of updates. Fig. 2.7 shows an example of the particle behaviours and the change of particle number from the initialization phase to the stage that the particle filter has already been converged.

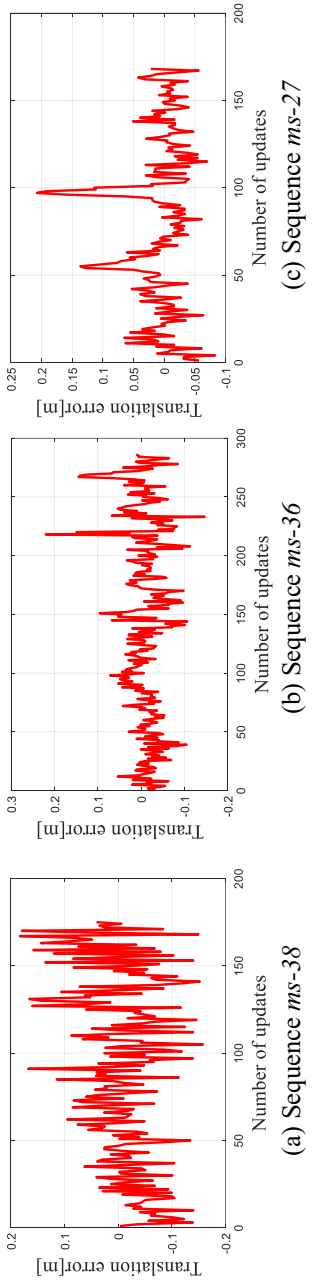


(a) *ms-38*: GT trajectory (b) *ms-36*: GT trajectory (c) *ms-27*: GT trajectory

(d) *ms-38* on Map A (e) *ms-36* on Map A (f) *ms-27* on Map A

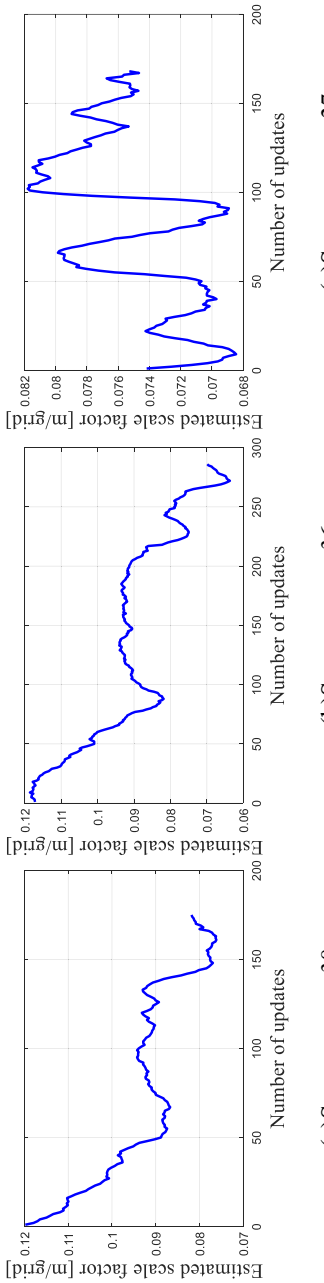
(g) *ms-38* on Map B (h) *ms-36* on Map B (i) *ms-27* on Map B

Figure 2.8: Real-time self-localization trajectories on the MIT Stata benchmark dataset. The first row plots the trajectory with ground truth. The second and the third row depicts the trajectories estimated by the proposed localization method in two maps with different scale-drift conditions. The red cross marks in (**a**–**c**) are the initial locations of the robot. The red star marks in (**d**–**i**) are the locations where MCL method is converged (also the position where the trajectories start to be recorded).



(a) Sequence *ms-38* (b) Sequence *ms-36* (c) Sequence *ms-27*

Figure 2.9: The translational errors $e_{trans,t}$ in the localization tests on MIT Stata dataset.



(a)Sequence *ms-38* (b)Sequence *ms-36* (c)Sequence *ms-27*

Figure 2.10: The estimated scale factor in the tests on the benchmark dataset.

Table 2.1: Performance on the MIT Stata dataset.

	Particle Number: [1 k, 3 k]		
	ms-38	***ms-36***	***ms-27***
Success rate	100 %	100%	40%
Translational Error [%]	25.9	21.1	15.1
Convergence time [s]	25	23	6
	Particle Number: [2 k, 10 k]		
	ms-38	***ms-36***	***ms-27***
Success rate	100%	100%	100%
Translational Error [%]	19.2	14.3	15.0
Convergence time [s]	28	22	19

Fig. 2.11 verifies that the larger amount of particles contribute to the higher accuracy. This is also supported by the data in Table 2.1. Furthermore, Fig. 2.11 also demonstrates that the boxplots of the tests with 1 k–3 k particles is more likely to be skewed to the right, which indicates that the pose estimation with fewer particles drifts more frequently.

Table 2.1 illustrates that the proposed method is able to provide effective state estimations on *ms-36* and *ms-38* for all these runs. For the *ms-27*, because of the incompleteness of the map of "Area 4", the robot has to pass through the "unknown" grids as shown in Fig. 2.8 (f). This may lead to the localization failures and greater error. At the beginning of sequence *ms-36*, the robot stays statics for about 10 s, therefore the convergence speed is slower as compared to other sequences. Note that, the larger number of particles lead to a greater particle diversity as well, therefore, sometimes the particle filter with 1–3 k particles has faster convergence ability.

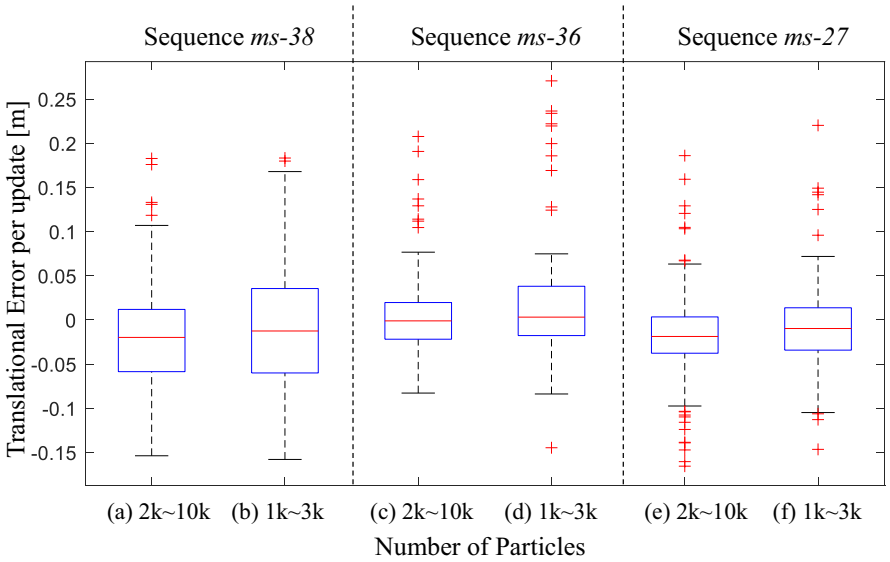


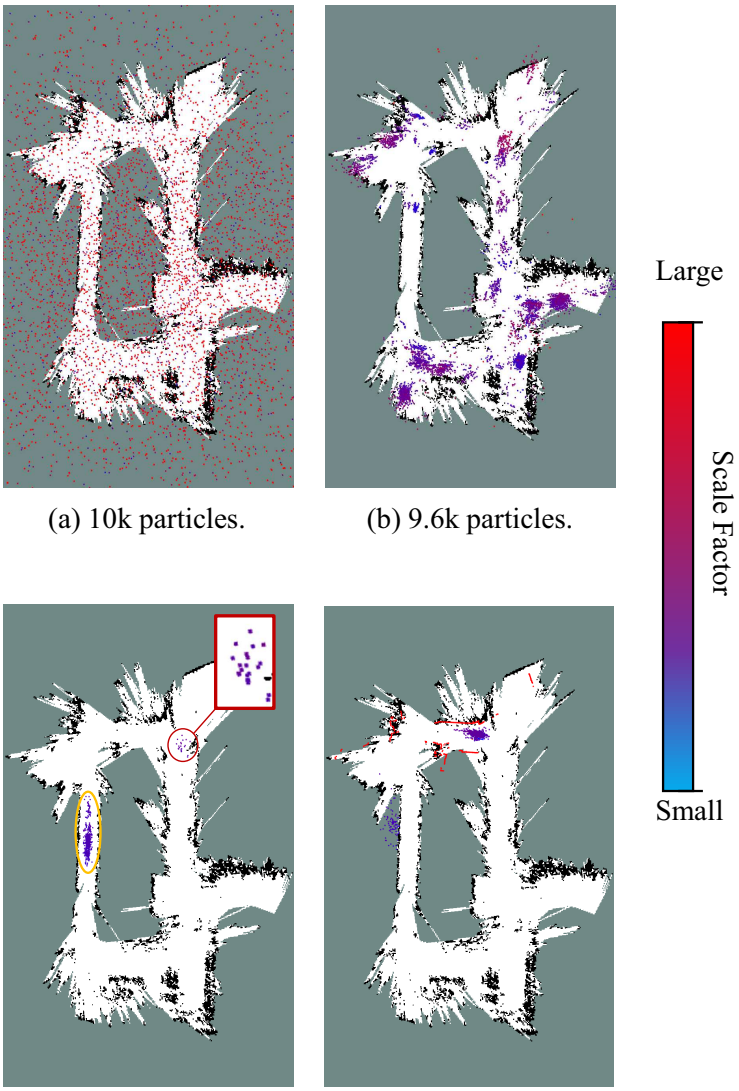
Figure 2.11: The translational errors in the tests with different amount of particles.

Global Localization Tests

If the robot starts off with no idea of its position relative to the map, the MCL method needs to be initialized as a global localization problem. In our case, the global localization can be achieved by initially distributing the state hypotheses over all the possible poses and scale factors. Since more particles are needed for the global localization, 2000–10,000 particles are employed in the global localization tests. Here, it should be noticed that in consideration of the incompleteness of the map, the particles are initially generated not only on the free grids (white grids), but also in the unknown parts (gray grids).

Fig. 2.12 quantitatively depicts a converging process of the particle filter in the test of global localization on *ms-36*. The color of a particle represents its scale factor. In the first iteration (Fig. 2.12 (a)), particles with varying poses and scale factors are randomly sampled. Within seconds (Fig. 2.12 (b)), the scales of particles is rapidly converged into a small range. However, there still exist multiple candidate clusters centred at different poses. Soon afterwards, there are only a few clusters left (Fig. 2.12 (d)). The red circle indicates the cluster which should be the optimal solution, whereas it carries very low weight. Finally, at the end of initialization phase (Fig. 2.12 (d)), the particles are successfully localized in the same cluster with the similar scales, and the state estimation of the robot will be considered as converged.

The global localization performance on the sequence *ms-38* and *ms-27* is evaluated as well. The state estimation in all these tests can be initialized successfully, however, with significant longer convergence time. After the global localization is converged, the performance of the MCL method will be identical to a pose tracking process.



(a) 10k particles. (b) 9.6k particles.

(c) 2k particles. (d) 2k particles, converged.

Figure 2.12: Particle distribution during the initial stage of the global localization process on the sequence *ms-38*. The scale factor of the particles are indicated by their colors. (**a**) The initial distribution of the particle filter. (**b**) The particle filter yields a highly diverse range of particle clusters after a few seconds. (**c**) A small number of clusters survive after several updates. The correct robot state is represented by the cluster labelled with the red circle. (**d**) The particle cloud is converged, suggesting that the scale estimation has been converged.

Figure 2.13: The mobile robot platform used in the real experiment.

2.5.2 Results in Manually Collected Experiments

To test the feasibility of the proposed method, experiments are performed in real environments with a ground robot. In our practical experiments, a hand-held web camera paired with a wide-angle lens is used to acquire the image data and perform the mapping. A Pioneer-P3DX wheel robot as shown in Fig. 2.13 is used as the experimental platform to perform localization on the obtained map. The robot is equipped with a 2D LRF at about 0.3 meters above the ground. The average speed of the robot is about 1.2 m/s. Several key parameters of the camera and the LRF are displayed in Table 2.2 and Table 2.3, respectively. The scan from LRF is uniformly downsampled to 40 beams according to their emitting angle.

Figure 2.14: Experimental environment, the engineering building of Hokkaido University.

Table 2.2: Basic parameters of the camera and lens used in experiments.

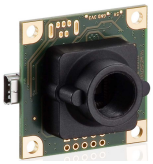
	**Camera Model**	UI-1221-LE
	Sensor Type	CMOS Color
	Shutter	Global Shutter
	Resolution	0.36 Mpix
	Resolution (hxv)	752 × 582 Pixel
	Aspect Ratio	14:9
	Color Depth	8 bit
	Optical Sensor Class	1/3”
	Optical Sizes	4.512 mm× 2.880 mm
	Optical sensor diagonal	5.35 mm
	**Lens Model**	BM 2420
	Mount	S-Mount
	Sensor	1/3 inch
	Aperture	2
	Focal Length	2.4 mm
	Baack Focal	4.56
	Angle of View	D: 132 H: 104 V: 77
	IR Correction	Yes
	Distortion	-25

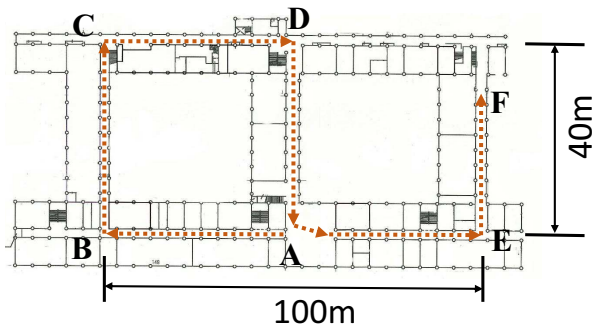
Table 2.3: The LRF used in experiments.

Model	LMS111-10100
Light Source	Infrared (905 nm)
Aperture Angle	270
Scanning Frequency	25 Hz/50 Hz
Angular Resolution	0.25 − 0.5
Heating	Yes
Working Range	5 m − 20 m
Amount of evaluated echoes	2

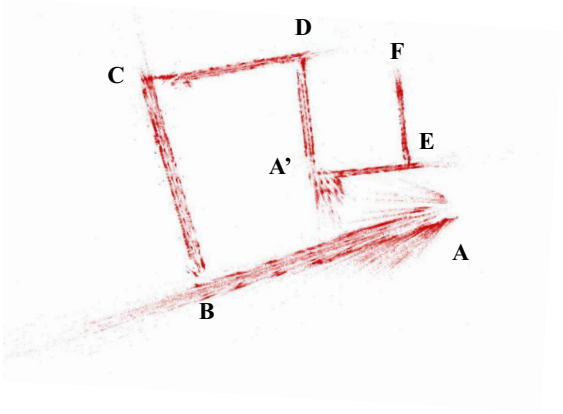
The data collection is carried out in a long corridor as shown in Fig. 2.14. A floor plan of the target environment with the correct size is shown in Fig. 2.15 (a). To collect the image data, we hold the camera and walk from point A to F, the travel path on the floor plan is a S-shape one depicted as the red dashed lines in Fig. 2.15 (a). The monocular version of ORB-SLAM2 [1] without loop closure is employed for visual mapping. The obtained raw map is shown as the point cloud in Fig. 2.15 (b). Compared with the real path in Fig. 2.15 (a), it can be noticed that significant scale drift occurs during the SLAM process, and the shape of the map is distorted. This also confirms the fact that the scale of monocular SLAM drifts severly when a large rotation occurs [27]. In the obtained map, point A and A' should be the same point if the map is metrically consistent or the loop-closure is carried out. As the result, the point A is not directly linked to point E. Finally, the raw map is processed into a 2D grid one for performing localization as shown in Fig. 2.15 (c).

Next, the P3-DX is controlled to move following the identical path of the camera and collect a sequence of LRF data named *hu-1*, from point A to F. Then, the same robot is driven along the opposite direction, from point E to A, to collect another sequence named *hu-2*. The localization tests are performed based on both the two sequences.

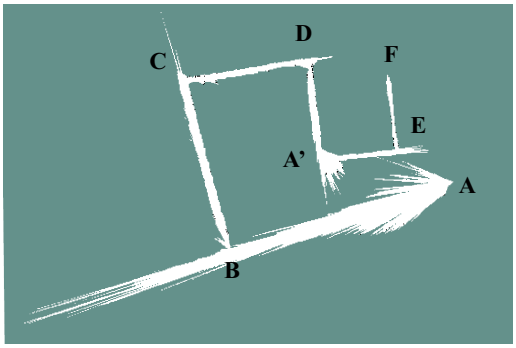
In this experiment, the particle filter is initialized around the positions where the robot starts off. The number of particles is set as 2000–6000. The estimated trajectories of both the two sequences are plotted in Fig. 2.16. Because of the low quality of the map around point A, it takes 12 s for the *hu-1* to be converged into a single solution, while the *hu-2* is more rapidly converged in 4 s. Five runs are performed on the two sequences respectively. The proposed method is able to provide correct localization result on all these runs. In rare occasions, the particles spread out along different directions as multi-model distributions.



(a)

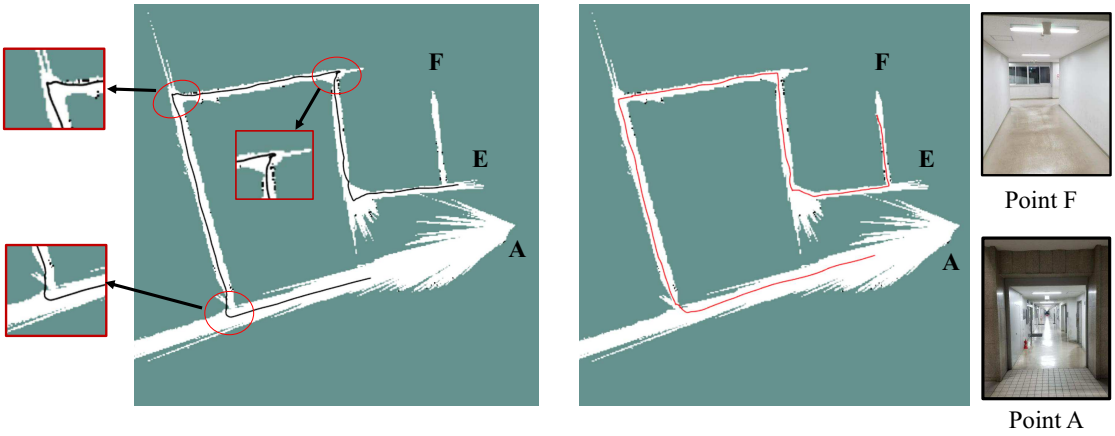


(b)



(c)

Figure 2.15: (**a**) The plan of the experimental environment. Red dot arrows represent the direction of camera trajectory, (**b**) scale-drifted point cloud map obtained by monocular SLAM, (**c**) 2D grid map. In (**b**,**c**), point A and A' should be at the same position if the map is metrically correct.

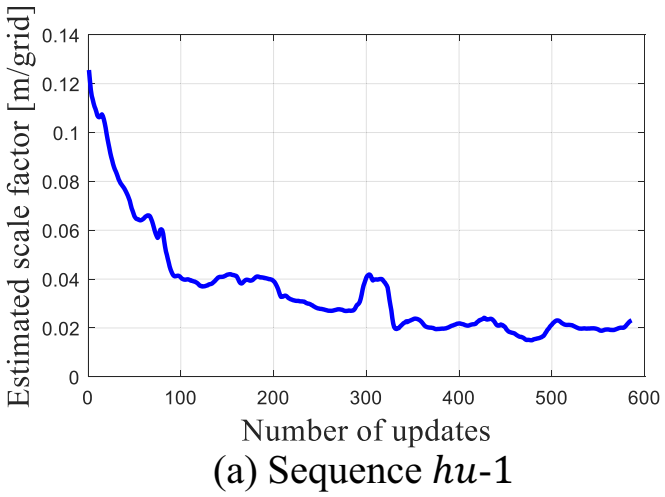


(a) *hu-1*, travel from A to E (b) *hu-2*, travel from F to A

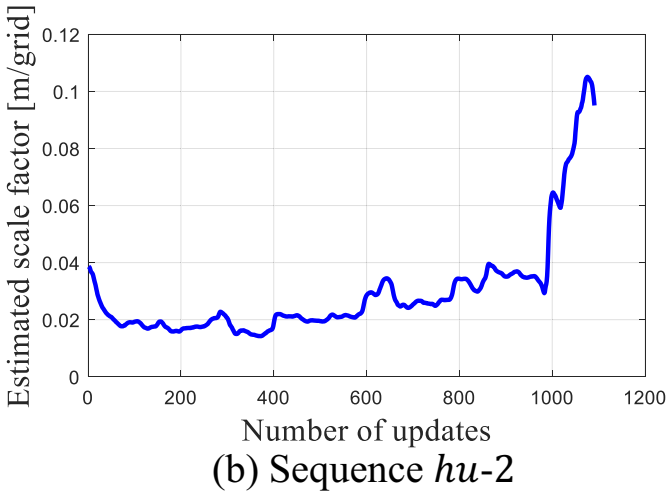
Figure 2.16: Estimated trajectories on the manually collected dataset. The red circles highlight the distorted parts caused by the accumulated error. Snapshots at point A and F are shown at the right side.

It may be found that the trace is distorted at the position as labelled by the red circles in Fig. 2.16 (a). This is because of the lack of geometry features in the corridors that prevents the laser measurement model from correcting the error of the estimated displacement along the corridor. This displacement error will be accumulated until the robot comes across a corner and thereby lead to these distortions of traces. This issue can hardly be fixed depending on only the LRF, even though, the proposed MCL is still able to re-converge and output the correct trajectories.

Then, the effect of scale estimation during localization is investigated. Fig. 2.17 shows the estimated scale factors in these tests. It can be seen that the trends of the two sequences are symmetric to each other since they travel along opposite directions. The scale factor at point A is around 0.9, and drifted to 0.2 after reaching point F.



(a) Sequence *hu*-1



(b) Sequence *hu*-2

Figure 2.17: Estimated scale factor in the tests on manually collected dataset.

Fig. 2.18 describes the processes of how the scale factor is adjusted to align the drifted map. The red points shown in Fig. 2.18 are the laser points that are rescaled based on the estimated scale factors s_t. It can be seen that once the robot comes across a corner, there is a large disagreement between the laser beam and the map. After being corrected by the measurement model, the particle fil-

ter updates the s_t in order to adapt the LRF measurements to the map. In the experiments, these trajectories are visually inspected in order to ensure that no localization failure happens. The experiments illustrate that the proposed method is able to withstand the large scale drift in the map of the low-texture environment. Moreover, the video of the practical experiments can also be found in the supplementary section.

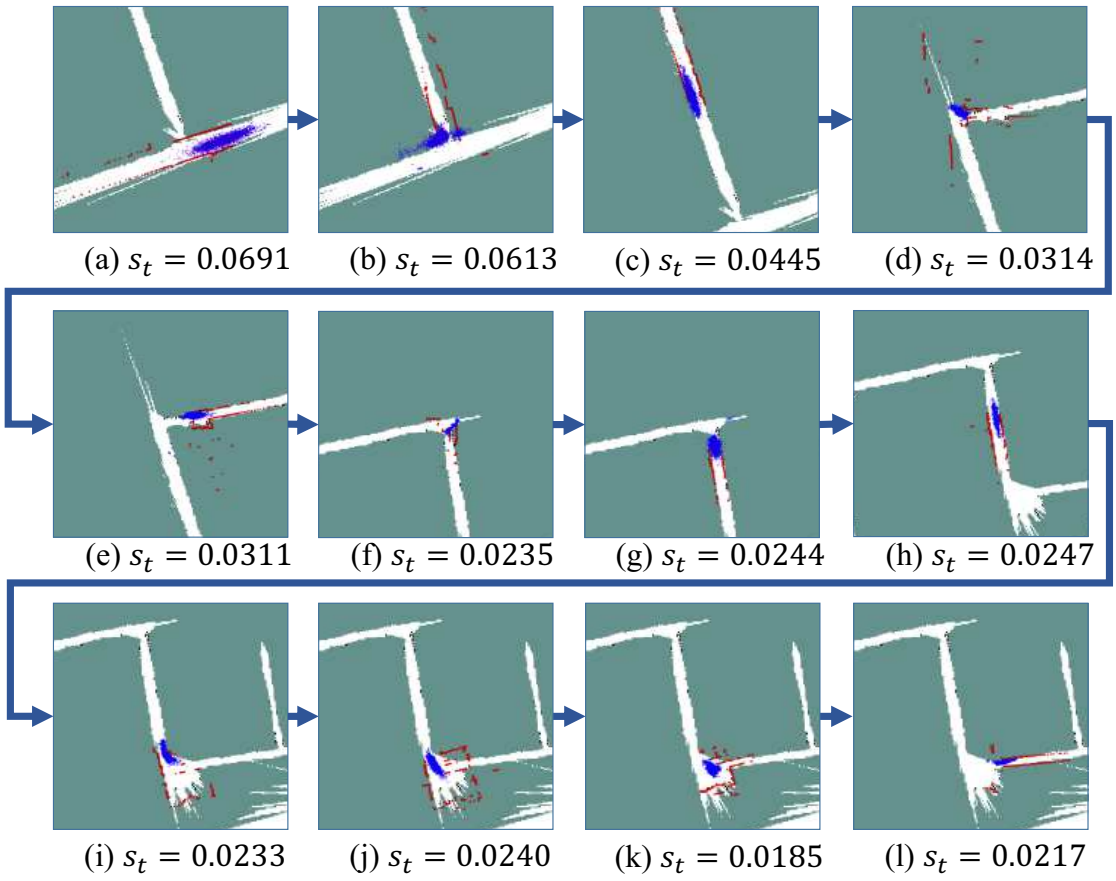


Figure 2.18: The scale correction for making rescale measurements match the drifted map. Red dots are the laser beams that are rescaled based on the estimated s_t. Blue points are the particles employed by the MCL method.

2.5.3 Runtime Performance

All the evaluations were carried out on a PC with Intel Core i7-3740m CPU and 8 GB memory. Fig. 2.20 shows the time cost of the proposed 2D localization algorithm at each update. As discussed in [57], the computational cost of the particle filter is proportional to the number of particles. The main cost of an update is the calculation of the observation likelihood based on the beam model and the resample process. In the beginning, because of the high uncertainty of the robot state, a large number of particles are required, hence it takes about 130 ms to manipulate 3 k particles. Then a steep reduction of the time cost can be observed owing to the effect of KLD-sampling. Finally, the particle number reaches the low limitation, and the time is optimized at about 40 ms. Moreover, the time cost of each update is around 350 ms with 10 k particles, which is the maximum computational cost of all the evaluations in Section 2.5. It is of note that the particle filter is not updated under a fixed frequency. Instead, the update is triggered only if the robot makes a certain transformation. Under the moving velocity of our robot platform (0.6 m/s), the average time interval between two updates is about 390 ms, thus the real-time performance can be achieved at most of the time.

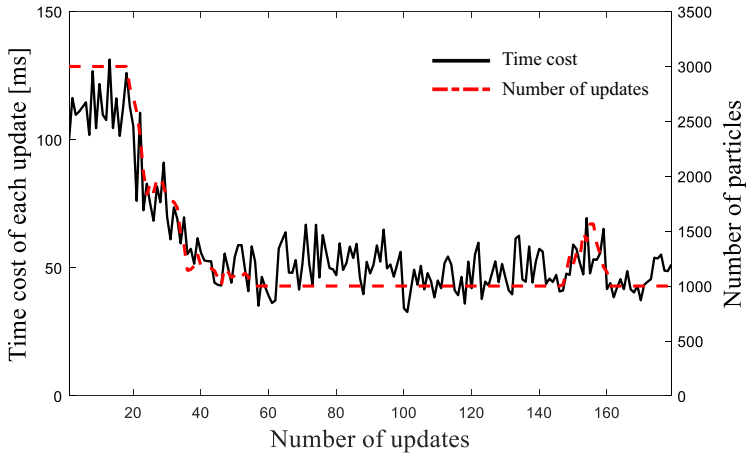


Figure 2.19: Runtime performance of the extended MCL algorithm's each iteration in local localization.

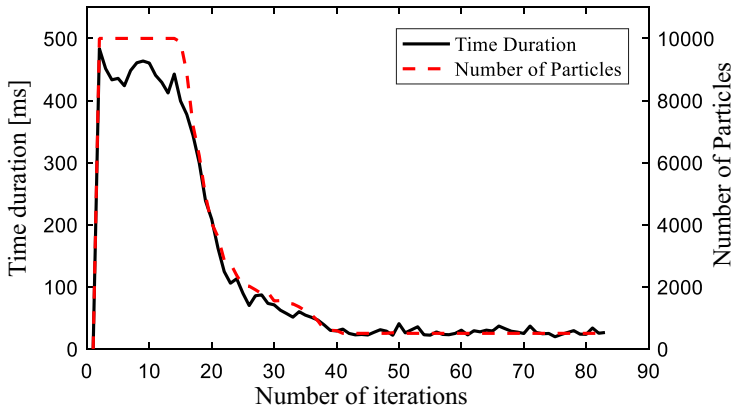


Figure 2.20: Runtime performance of the extended MCL algorithm in global localization.

2.6 Conclusions

In this chapter, we propose a novel localization approach which allows the robot to reuse the scale-drifted map from monocular SLAM, with only a laser range finder. To the best of our knowledge, the framework proposed in this chapter is

the first trial to address the problem with this setup. By extracting 2D structures from the point cloud, the 3D point cloud from monocular SLAM is converted to a 2D grid map for localization. The proposed localization approach is built upon a variant of Monte Carlo localization method, where both the robot's pose and map scale are handled. The proposed method is tested on a public dataset and real environments. The result shows that the localization method can provide a relatively accurate estimation of the robot pose. Moreover, in feature-less environments, where the monocular SLAM produces significant scale drift, the proposed method is still able to give a quick and consistent localization result with correct scale estimation.

The current work can be fulfilled in multiple directions. Some limitations of the proposed method have already been discussed. Utilizing a more robust measurement model, such as likelihood field model [58] or normal distribution representation [59], it may be possible to enhancing the localization accuracy. Also, the pose estimation can be extended to 6 DoF that the usage of 3D LIDAR sensors will be involved in order to improve the robustness against the point cloud map. Futrthermore, the auto-navigation can also be achieved via combining the existing path planning method [60].

Chapter 3

3D Point Cloud Registration Method with Scale Stretching

3.1 Introduction

In the previous chapter, a Lidar-based extended MCL method is developed against the map built by monocular SLAM to handle the scale-drift problem. In order to achieve the global searching and optimize the time cost at the same time, the extended MCL method degraded the map from 3D point cloud to the 2D grid map. However, this operation may decrease the accuracy of the localization result because of the discarded vertical information. Additionally, sometimes 2D pose estimation is not enough for the robot's self-navigation. Compared with a 2D Lidar, a 3D one can provide far more abundant information about the environment. Additionally, more feature points can be utilized via associating the data between the 3D map and the 3D Lidar scans thus the accuracy of the localization system can be improved. Therefore, this chapter, discusses a possible solution for robot's 6DoF self localization in a scale-drifted map. To be specific, a 3D scan matching method is proposed to calculate the spatial transformation between the reference point cloud and the source point cloud even if the source point cloud is resized.

The point clouds registration algorithm, as one of the key technologies in data processing, plays an important role in the robot mapping and localization. One of the most popular used point cloud alignment solution is the ICP (iterative closest points) pipeline. A prototype of ICP algorithm can be traced to the year 1981, when Lucas [61] proposed a method for estimating the rotation between 3D images. The original ICP method [62] is a method to solve the rigid registration between two point cloud. Denote the point cloud $\mathcal{P}^r$ as the reference point cloud and $\mathcal{Q}$ as the source point cloud, the goal of an ICP procedure is to find an optimal transformation, with which $\mathcal{Q}$ can be transformed to perfectly align $\mathcal{P}$. As depicted in Fig. 3.1, the standard ICP consists of two steps: 1. Find the correspondence of the nearest point pairs between two point clouds. 2. Compute a new transformation which can minimize the sum of squared distances between the point pairs. By performing these two steps iteratively, the rigid transformation can be obtained very rapidly.

Several variants of the stand ICP have been proposed by researchers to handle the scale estimation. In the paper [63], the author used least squre method to solve a bounded scale matrix. Ying et al. [64] proposed the Scale-ICP algorithm, in which the registration problem was formulated into quadric constraint problem in a 7-dimension space and solved by the singular value decomposition (SVD) technique. In [65] [66], the ICP method is extended to calculate the affine transformation between point clouds. However, the main purpose of these methods is to solve the non-rigid transformation and the time cost of them are too heavy to be performed on the real time.

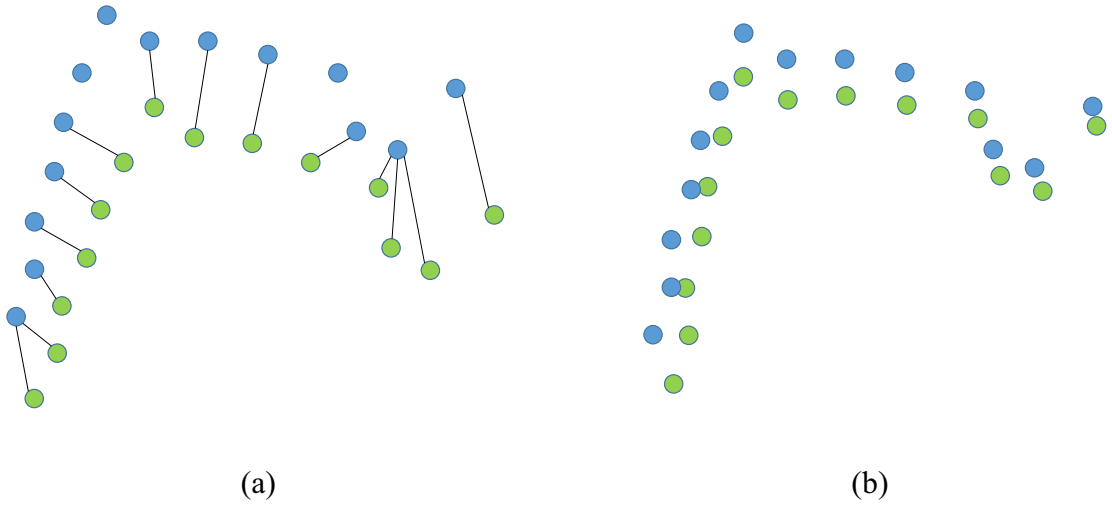


Figure 3.1: (a) ICP point association between point clouds. (b) Point transformed after one iteration of ICP

Normal distribution transform (NDT) is another widely used point cloud registration method. The goal of using NDT for scan registration is also to estimate the rigid transformation two point clouds. The difference is that the main idea of ICP method is to find the transformation which can minimize the total distance of the point pairs, while the NDT method is aiming to maximize the value of the likelihood function which represents the score of the points in the transformed source point cloud lying on the reference point cloud. An example of the transferred normal distribution of a point cloud is shown in Fig. 3.2. The NDT method is proved to be more robust than the ICP method when a large initial error exists [67]. Additionally, since the normal distribution of the reference point cloud in NDT is computed off-line, the speed of NDT is faster than the ICP method for a large point cloud.

These advantages of the NDT make it a more appropriate algorithm of localizing a since scan in a prior point cloud map. However, the original NDT

method does not take the scale changing into consideration. In this chapter, a scale-stretching NDT method (sNDT) will be introduced. In order to verify the robustness and performance of the sNDT, evaluations are provided for both the scan-to-scan and scan-to-map matches.

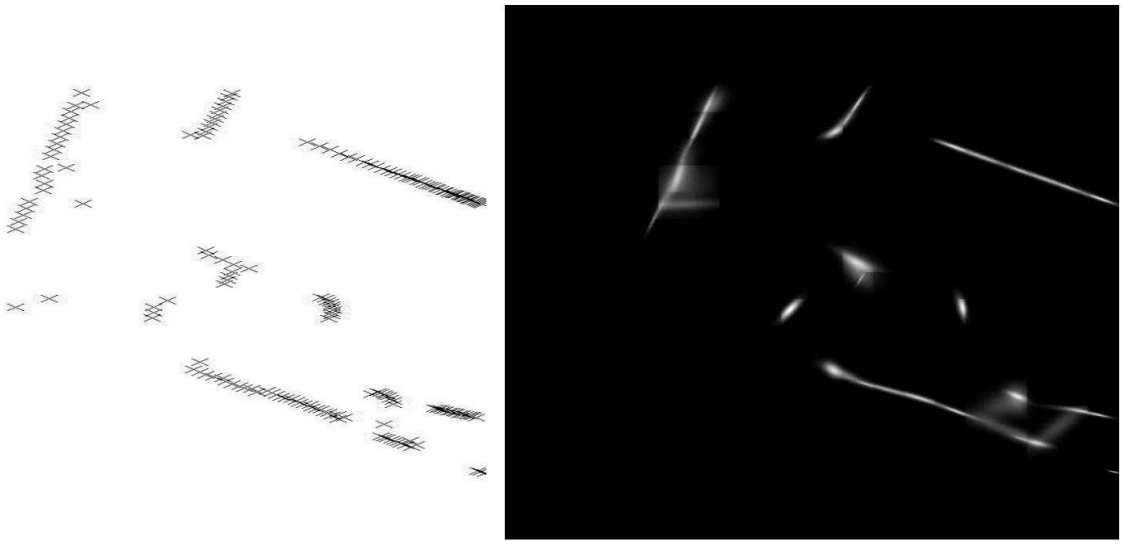


Figure 3.2: NDT scan matching. The reference point scan (left) and its normal distribution density (right). [5]

3.2 Point Cloud Registration with Scale Stretching

The input of the proposed sNDT is a reference point cloud (in our case, the map point cloud, sometimes also called target point cloud) $\mathcal{P}$ and a source point cloud (current scan from the Lidar) $\mathcal{Q}$. The transformation between $\mathcal{P}$ and $\mathcal{Q}$ is composed of a scale factor and a 6DoF rigid transformation. Thus this relative transformation can be represented by a similarity transformation $T_s(\overrightarrow{\xi})$, in which $\overrightarrow{\xi}$ is a vector that encodes the transformation.

Assume that the space of the reference point cloud is divided into cells, and the points in each cell follow a specified distribution. If the source point cloud is transformed by the transformation $T_s(\overrightarrow{\xi})$, the optimal transformation should obey the same distribution of the reference point cloud. The NDT method optimizes a likelihood function structured based on the distribution instead of directly using the distances between point sets, in which the transformation which maximizes the total likelihood function of all the cells will be considered as the optimal transformation:

$$T_s^{opt} = argmax \prod_{i=1}^{n} p(T(\overrightarrow{\xi}, \mathcal{Q})), \tag{3.1}$$

in which n is the number of the points in $\mathcal{P}^t$.

The proposed sNDT method follows the pipeline of the standard NDT method [5]. Denote one single point in a cell as $\overrightarrow{q}$, the PDF (probability density function) of each cell is a combination between a uniform distribution and a normal distribution [68]:

$$p_x(\overrightarrow{q}) = c_1 \exp(\frac{(\overrightarrow{q} - \overrightarrow{\mu})\Sigma^{-1}(\overrightarrow{q} - \overrightarrow{\mu})}{2}) + c_2 p_o, \tag{3.2}$$

where p_o is determined by the outlier ratio, c_1 and c_2 are constants for ensuring the PMF (probability mass function) within the range of a cell is 1. By approximating the negative log-likelihood function of the PDF, the score of an arbitrary point in the reference scan can be given by:

$$s(\overrightarrow{q}) = -d_1 \exp(-\frac{\mathrm{d}_2}{2}(\overrightarrow{q} - \overrightarrow{\mu})\Sigma^{-1}(\overrightarrow{q} - \overrightarrow{\mu})), \tag{3.3}$$

where $\overrightarrow{\mu}$ and Σ is the mean and covariance of the specified cell which the point $\overrightarrow{q}$ is located.

In order to fit the boundary conditions of the distribution:

$$\begin{aligned} d_1 &= -\log(\mathrm{c}_1 + \mathrm{c}_2) - \mathrm{d}_3 \\ d_2 &= -2\log((-\log(\mathrm{c}_1 \exp(-1/2) + \mathrm{c}_2) - \mathrm{d}_3)/\mathrm{d}_1), \\ d_3 &= -\log(\mathrm{c}_2) \end{aligned} \tag{3.4}$$

Eq. (3.3) illustrates how much the point $\overrightarrow{q}$ affects the likelihood function. The goal of the sNDT method is to estimate the best transformation which can maximize the total influence of all the points in source point cloud. To solve this optimization problem, the Newton's algorithm has been utilized to calculate the optimal parameter. For a function $F(x)$ and an initial value x_n, the Newton's method can be recursively written as:

$$x_{n+1} = x_n - \nabla^2 F(x)^{-1} \nabla F(x),, \tag{3.5}$$

where $\nabla^2 F(x)^{-1}$ is the inverse of Hessian matrix, $\nabla F(x)$ is the gradient matrix of $F(x)$. The Newton's method will be used to approximate the optimal x value along the search direction.

For the sNDT score function, the gradient is given by:

$$
\begin{aligned}
g_i &= \frac{F_{score}(\overrightarrow{\xi})}{\overrightarrow{\xi}} \\
&= \sum_{k-1}^{n} d_1 d_2 \boldsymbol{q}_k^T \Sigma_k^{-1} \frac{\delta \boldsymbol{q}_k}{\delta \overrightarrow{\xi}_i} exp(\frac{-d_2}{2} \boldsymbol{q}_k \Sigma_k^{-1} \boldsymbol{q}_k)
\end{aligned}, \tag{3.6}
$$

The Hessian Matrix $\boldsymbol{H}$ is calculated by:

$$
\begin{aligned}
H_{ij} &= \frac{\partial^2 s}{\partial \xi_i \partial \xi_j} = \partial(\frac{\partial s}{\partial \xi_i})/\partial \xi_j \\
&= \partial(\boldsymbol{q}^T \Sigma^{-1} \frac{\partial \boldsymbol{q}}{\partial \xi_i}) exp(\frac{-\boldsymbol{q}^T \Sigma^{-1} \boldsymbol{q}}{2})/\partial \xi_j \\
&= \partial((\boldsymbol{q}^T \Sigma^{-1} \frac{\partial \boldsymbol{q}}{\partial \xi_i})/\partial \xi_j) exp(\frac{-\boldsymbol{q}^T \Sigma^{-1} \boldsymbol{q}}{2}) + (\boldsymbol{q}^T \sigma^{-1} \frac{\partial \boldsymbol{q}}{\partial \xi_i})(\partial exp(\frac{-\boldsymbol{q}^T \Sigma^{-1} \boldsymbol{q}}{2})/\partial_j), \\
&= -exp(\frac{-\boldsymbol{q}^T \Sigma^{-1} \boldsymbol{q}}{2})[(\boldsymbol{q}^T \Sigma^{-1} \frac{\partial \boldsymbol{q}}{\partial \xi_i})(\boldsymbol{q}^T \Sigma^{-1} \frac{\partial \boldsymbol{q}}{\partial \xi_j}) - (\frac{\partial \boldsymbol{q}^T}{\partial \xi_j} \Sigma^{-1} \frac{\partial \boldsymbol{q}}{\partial \xi_i}) - \\
&(\boldsymbol{q}^T \Sigma^{-1} \frac{\partial^2 \boldsymbol{q}}{\partial \xi_i \partial \xi_j})]
\end{aligned} \tag{3.7}
$$

The Euler angles and translation vectors are used to describe the transformation function since the Euler angles are constraint-free representations. Therefore the 7DoF similarity transformation can be encoded by a vector $\overrightarrow{p}_7$:

$$
\overrightarrow{p}_7 = \{t_x, t_y, t_z, \phi_x, \phi_y, \phi_z, s_s\}, \tag{3.8}
$$

where s_s is the inverse scale factor of the source point cloud, the value ϕ_x, ϕ_y, ϕ_z are Euler angles of the rotation.

The similarity transformation with scale stretching factor between point cloud can be written as:

$$
T_S = s_s \boldsymbol{R} \overrightarrow{q} + \overrightarrow{t}, \tag{3.9}
$$

where $\boldsymbol{R}$ is the rotation matrix presented Euler Angles, and $\overrightarrow{t}$ is the translation vector. Using Euler angles, the rotation matrix can be written as:

$$s_s \cdot \boldsymbol{R} = s_s \cdot \begin{bmatrix} c_y c_z & -c_y s_z & s_y \\ c_x s_z + s_x s_y c_z & c_x c_z - s_x s_y s_z & -s_x c_y \\ s_x s_z - c_x s_y c_z & c_x s_y s_z + s_x c_z & c_x c_y \end{bmatrix}, \tag{3.10}$$

where $c_{\{x,y,z\}}$ and $s_{\{x,y,z\}}$ denote the cos and sin value of the Euler angle $\phi_{\{x,y,z\}}$, respectively.

The translate vector is denoted as:

$$\overrightarrow{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.11}$$

To perform the nonlinear optimization, numeric differential is widely used in computing the Jacobian matrix and the Hessian matrix, for example, the differentiation of the transformation has been given numerically in [69]. Whereas the Euler angles allow for very intuitive differentiation, thus the analytical result of $\boldsymbol{H}_T$ and $\boldsymbol{J}_T$ are utilized in the implementation.

Therefore, the Jacobian matrix of the transformation is 3×7 matrix, in which the ith column stands for $\frac{\delta \boldsymbol{q}_k}{\delta \overrightarrow{\xi}_i}$ in Eq. 3.6:

$$\boldsymbol{J}_T = \begin{bmatrix} 1 & 0 & 0 & 0 & c & f & i \\ 0 & 1 & 0 & a & d & g & j \\ 0 & 0 & 1 & b & e & h & k \end{bmatrix}, \tag{3.12}$$

in which:

$$
\begin{aligned}
a &= [(-s_x s_z + c_x s_y c_z)q_1 + (-s_x c_z - c_x s_y s_z)q_2 + (-c_x c_y)q_3]s_s \\
b &= [(c_x s_z + s_x s_y c_z)q_1 + (-s_x s_y s_z + c_x c_z)q_2 + (-s_x c_y)q_3]s_s \\
c &= [(-s_y c_z)q_1 + (-s_x s_y s_z)q_2 + (-s_x c_y)q_3]s_s \\
d &= [(s_x c_y c_z)q_1 + (-s_x c_y s_z)q_2 + (s_x s_y)q_3]s_s \\
e &= [(-c_x c_y c_z)q_1 + (c_x c_y s_z)q_2 + (-c_x s_y)q_3]s_s \\
f &= [(-c_y s_z)q_1 + (c_x c_y s_z)q_2]s_s \\
g &= [(c_x c_z - s_x s_y s_z)q_1 + (-c_x s_z - s_x s_y c_z)q_2]s_s \\
h &= [(s_x c_z + c_x s_y s_z)q_1 + (c_x s_y c_z - s_x s_z)q_2]s_s \\
i &= c_y c_z - c_y s_z + s_y \\
j &= c_x s_z + s_x s_y c_z + c_x c_z - s_x s_y s_z - s_x c_y \\
k &= s_x s_z - c_x s_y c_z c_x s_y s_z + s_x c_z + c_x c_y
\end{aligned}
\tag{3.13}
$$

The second-order derivative is 21×7 matrix, in which each 3×1 submatrix corresponds $\frac{\partial^2 \boldsymbol{q}}{\partial \xi_i \partial \xi_j}$ in Eq. 3.7:

$$
\boldsymbol{H}_T = \begin{bmatrix}
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} \\
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & a_{3\times1} & b_{3\times1} & c_{3\times1} & g_{3\times1} \\
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & b_{3\times1} & d_{3\times1} & e_{3\times1} & h_{3\times1} \\
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & c_{3\times1} & e_{3\times1} & f_{3\times1} & i_{3\times1} \\
0_{3\times1} & 0_{3\times1} & 0_{3\times1} & 0_{3\times1} & g_{3\times1} & h_{3\times1} & i_{3\times1} & 0_{3\times1}
\end{bmatrix},
\tag{3.14}
$$

in which:

$$a_{3\times1} = \begin{bmatrix} 0 \\ [(-c_x s_z - s_x s_y c_z)q_1 + (-c_x c_z + s_x s_y s_z)q_2 + (s_x c_y)q_3]s_s \\ [(-s_x s_z + c_x s_y c_z)q_1 + (-c_x s_y s_z - s_x c_z)q_2 + (-c_x c_y)q_3]s_3 \end{bmatrix}$$

$$b_{3\times1} = \begin{bmatrix} 0 \\ [(c_x c_y c_z)q_1 + (-c_x c_y s_z)q_2 + (c_x s_y)q_3]s_s \\ [(s_x c_y c_z)q_1 + (-s_x c_y s_z)q_2 + (s_x s_y)q_3]s_s \end{bmatrix}$$

$$c_{3\times1} = \begin{bmatrix} 0 \\ [(-s_x c_z - c_x s_y s_z)q_1 + (-s_x s_z - c_x s_y c_z)q_2]s_s \\ [(c_x c_z - s_x s_y s_z)q_1 + (-s_x s_y c_z - c_x s_z)q_2]s_s \end{bmatrix}$$

$$d_{3\times1} = \begin{bmatrix} [(-c_y c_z)q_1 + (c_y s_z)q_2 + (-sy)q_3]s_s \\ [(-s_x s_y c_z)q_1 + (s_x s_y s_z)q_2 + (s_x c_y)q_3]s_s \\ [(c_x s_y c_z)q_1 + (-c_x s_y s_z)q_2(-c_x c_y)q_3]s_s \end{bmatrix}$$

$$e_{3\times1} = \begin{bmatrix} [(s_y s_z)q_1(s_y c_z)q_2]s_s \\ [(-s_x c_y s_z)q_1 + (-s_x c_y c_z)q_2]s_s \\ [c_x c_y s_z q_1 + (c_x c_y c_z)q_2]s_s \end{bmatrix}$$

$$f_{3\times1} = \begin{bmatrix} [(-c_y c_z)q_1 + (c_y s_z)q_2]s_s \\ [(-c_x s_z - s_x s_y c_z)q_1 + (-c_x c_z + s_x s_y s_z)q_2]s_s \\ [(-s_x s_z + c_x s_y c_z)q_1 + (-c_x s_y s_z - s_x c_z)q_2]s_s \end{bmatrix}$$

$$g_{3\times1} = \begin{bmatrix} 0 \\ [(-s_x s_z + c_x s_y c_z)q_1 + (-s_x c_z - c_x s_y s_z)q_2 + (-c_x c_y)q_3]s_s \\ [(c_x s_z + s_x s_y c_z)q_1 + (-s_x s_y s_z + c_x c_z)q_2 + (-s_x c_y)q_3]s_s \end{bmatrix}$$

$$h_{3\times1} = \begin{bmatrix} [(-s_y c_z)q_1 + (-s_x s_y s_z)q_2 + (-s_x c_y)q_3]s_s \\ [(s_x c_y c_z)q_1 + (-s_x c_y s_z)q_2 + (s_x s_y)q_3]s_s \\ [(-c_x c_y c_z)q_1 + (c_x c_y s_z)q_2 + (-c_x s_y)q_3]s_s \end{bmatrix}$$

$$i_{3\times1} = \begin{bmatrix} [(-c_y s_z)q_1 + (c_x c_y s_z)q_2]s_s \\ [(c_x c_z - s_x s_y s_z)q_1 + (-c_x s_z - s_x s_y c_z)q_2]s_s \\ [(s_x c_z + c_x s_y s_z)q_1 + (c_x s_y c_z - s_x s_z)q_2]s_s \end{bmatrix}$$

In conclusion, the whole sNDT process is illustrated by the Algorithm 1. The iterative process can be speed up using multi-core calculation in both the division of cells and the update of Hessian matrix. The implementation of the proposed sNDT is based on the open sourced version from Point Cloud Library [70].

Algorithm 1: The flow of sNDT method.

Data: Reference point cloud: $\mathcal{P}^s$, target scan: $\mathcal{P}^t$.

Result: Relative Transformation: transformation vector $\overrightarrow{\xi}_7$

Divide space into cells $\mathcal{C}$ `// Initialization`;

for *each point* $p_i^s \in \mathcal{P}^s$ **do** `// Associating points`

  Search for its associated cell c_i in $\mathcal{C}$;

  Push back p_i^s into c_i;

end

for *each cell* $c_i \in \mathcal{C}$ **do**

  Calculate the expectation $\overrightarrow{\mu}_i$ of the point set in c_i;

  Calculate the covariance $\mathbf{\Sigma}_i^c$ of the point set in c_i;

end

while *Not converged* **do** `// optimization`

  $s = 0$;

  $\overrightarrow{g} = 0$;

  $\boldsymbol{H} = 0$;

  for *each point* $p_i^t \in \mathcal{P}^t$ **do**

    Find the cell ID c_k, in which p_i^t is projected by $T_s(\overrightarrow{\xi})$;

    if c_k *contains more than 5 points* **then**

      $score+ = F_{score}(T_s(\overrightarrow{\xi}_7, p_i^t))$;

      Calculate the gradient $\overrightarrow{g}$;

      Calculate the Hessian $\boldsymbol{H}$;

    end

  end

  $\delta\overrightarrow{\xi} = -H^{-1}\overrightarrow{g}$ `// Search direction of Newton's method`;

  $\overrightarrow{\xi} += \delta\overrightarrow{\xi}$;

end

3.3 Experiments

3.3.1 Evaluation of Scan-to-scan Registration

The proposed sNDT is first tested on resized scan pairs to evaluate both the accuracy and convergence. A ICP method with scale estimation (scaled ICP) is also involved in the evaluation to give a collaborative comparison between the sNDT.

Dataset

For the evaluation, two pairs of point cloud have been used to test their performance: a very simple 3D curve (*3D-curve*), a dataset obtained in an indoor environment (*cloud-office*), which is composed of two scan from the same building. Besides, another dataset extracted from the KITTI dataset (*KITTI-07*) is used. In the (*KITTI-07*), ten point clouds and their adjacent scans are randomly selected from the sequence 07 in the KITTI dataset. *3D-curve* contains 300 points. For both the dataset *pair-office* and *KITTI-07*, the point clouds have been down sampled with the 0.3-meter leaf size in order to ensure the speed of computation. The number of points in *KITTI-07* is from 15k to 22k after down sampling. For each point cloud pair, one of them is used as the reference point set, while the other one is resized by an affine transformation according to a scale ratio to evaluate its scale recovery ability.

In the KITTI dataset, a Velodyne HDL-64E Lidar is used to obtain the point cloud. Some core parameters of this sensor is as shown in Table 3.1.

Table 3.1: Basic parameters of the Velodyne HDL-64E Lidar.

	specifications
Number of lines	64 laser
degree of view	360
resolution (azimuth)	0.09 degree
resolution (elevation)	+2 degree up to -24.8 degree down
Distance accuracy	less than 2 cm
Field of view update	5-15 Hz
Range	50 meters
Number of points per sec	more than 1.333 M

Cell Size

The size of the cell in the NDT optimization is an important parameter. Since the points in each cell is assumed to obey a Gaussian, if the size of cell is set to be too large, the Gaussian distribution will distort the real shape of the point cloud. On the contrary, if the cell is too small, there may be very few points in each cell and the influence of noise will be increased. The size of the cells need to be chosen depending on the source of scan at hand. For the *3D-curve*, the size of cell is $0.25 \times 0.25 \times 0.25$. For the other two datasets, this value is set to $1 \times 1 \times 1$.

Results

To judge whether the algorithms are well converged or not, the error of the estimated scale factor is employed:

$$e_s = ||s_s - s_r|| \tag{3.15}$$

where s_s is the scale factor estimated by the registration methods and s_r is the pre-set scale ratio (ground truth) of the source point cloud. Here, the rotational and translational errors are not investigated since the translational error is also influenced by the scale factor s_s and the rotational error is frequently very small even through the sNDT method has already fallen into a local optimal.

Several runs on the *3D-curve* and *cloud-office* are conducted with different pre-set scale ratios. The results using sNDT and the scaled ICP on the *3D-curve* is graphically presented in the Fig. 3.4. The performances of both the algorithms are very similar when the initial scale ratio is a value around 1. In the interval $s_r \in [0.9, 1.3]$, the scaled ICP is superior to sNDT that the latter's result suffers from small errors. However, when the scale ratio of the source point cloud is increased to 1.6, the scaled ICP will be converged to an incorrect result while the NDT can still work stably until s_r reaches 2.0. And it is of importance to notice that, when the scale ratio is very small (less than 0.6), both the two algorithms fail to estimate the pose.

The similar conclusion can also be supported by the tests on *cloud-office*. The alignment results on *cloud-office* is shown in Fig. 3.6. It may be noticed that, the scale estimation error is typically smaller in the result of sNDT rather than the scaled ICP. In Fig. 3.5, an enlargement of the Fig. 3.8 shows the partial details of the reference point cloud and the point cloud aligned by sNDT, it is clear that the point cloud has been well overlapped to each other. The behaviour of the algorithms in this test is slightly different from the one on *3D-curve* on account that this pair of point sets is extracted from the real sensor scans, in which outliers exist and the point numbers are also much larger. Note that, in contrast to the result in Fig. 3.7, it can be seen from Fig. 3.8 that when the re-scale factor is 1.9, although the estimated scale is roughly correct, the rotation is not well estimated.

In the trials on the *KITTI-07* point pairs, sNDT also successes in registering

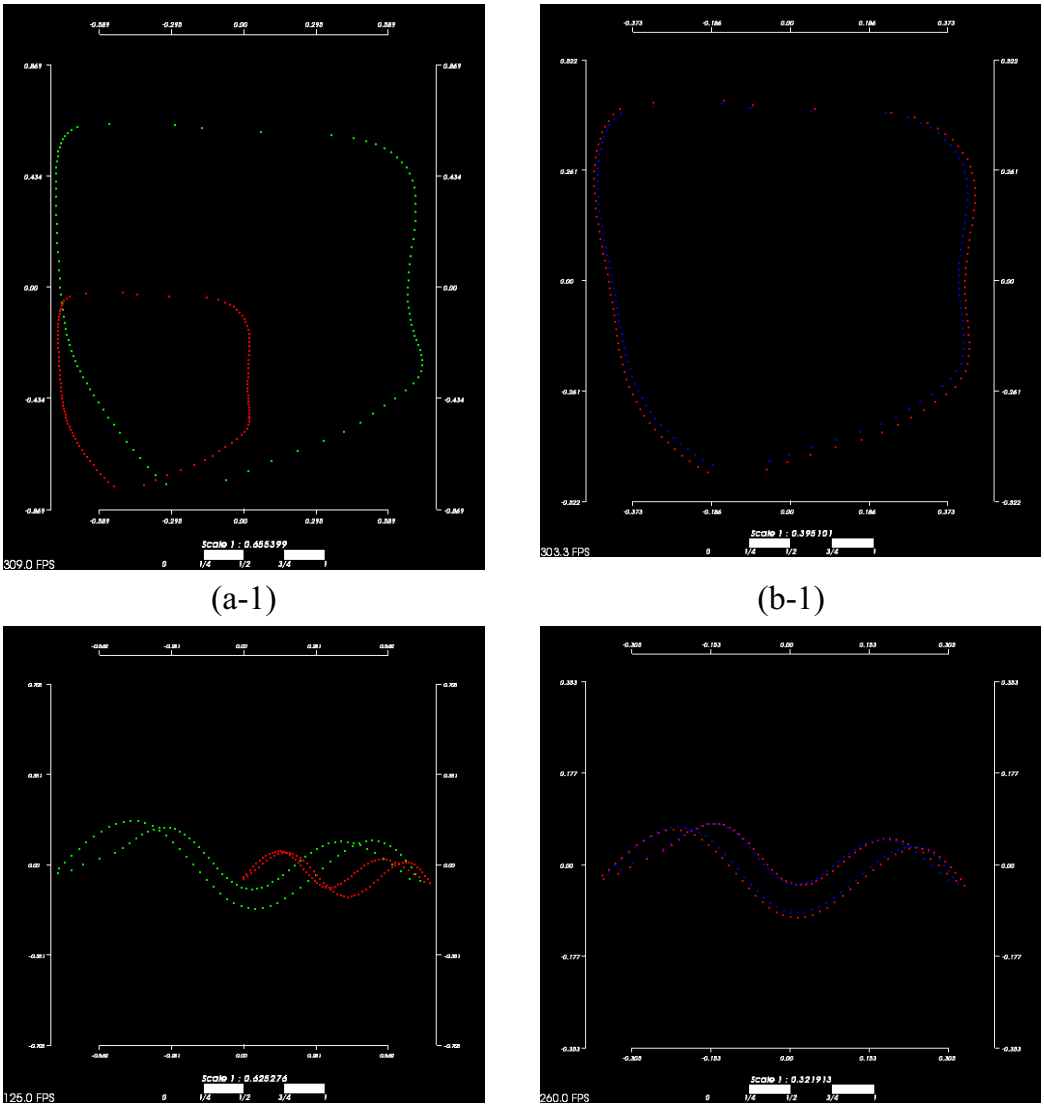


Figure 3.3: (a-1) and (a-2): top view and side view of the point clouds in *3D-curve*. (b-1) and (b-2): the alignment result of sNDT. Red point set represents the reference point cloud, while the green and blue ones represent the point cloud to be aligned and the transformed point cloud, respectively. The re-scale ratio of the source point cloud is 1.9.

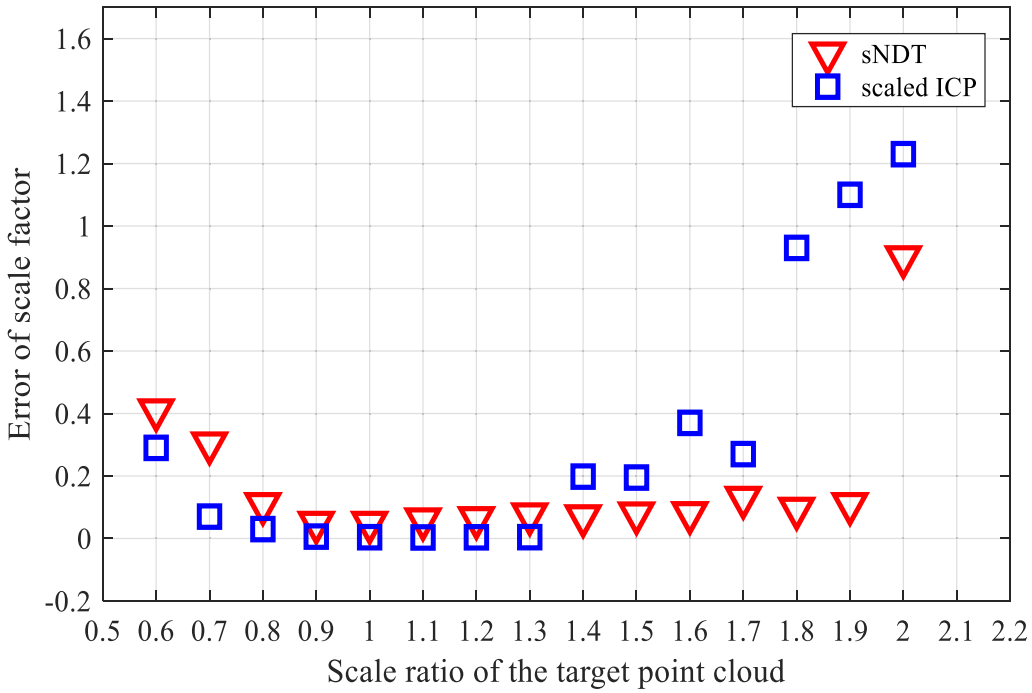


Figure 3.4: Scale error of the tests on *3D-curve*.

the point cloud with different scale ratios of the source set. Partial results can be seen in Fig. 3.9.

All the tests are conducted on a computer with Core i7 7820hk CPU. The execution time of 10 attempts of both the sNDT and scaled ICP on the three datasets with the scale ratio 1.3 and the original 3D NDT without scale changing is represented in the Table 3.2. The time cost of the sNDT is similar to the original NDT on both the *cloud-office* and *KITTI-07* while the scaled ICP takes longer time on all the three datasets. This conclusion also agrees with the comparison between the standard ICP and NDT given by [71]. The single run of the sNDT on the down-sampled point cloud pair from KITTI is under 50 ms, while the signal refreshing rate of this dataset is 100ms. This allows the sNDT to be a qualified selection for performing the 6DoF pose tracking.

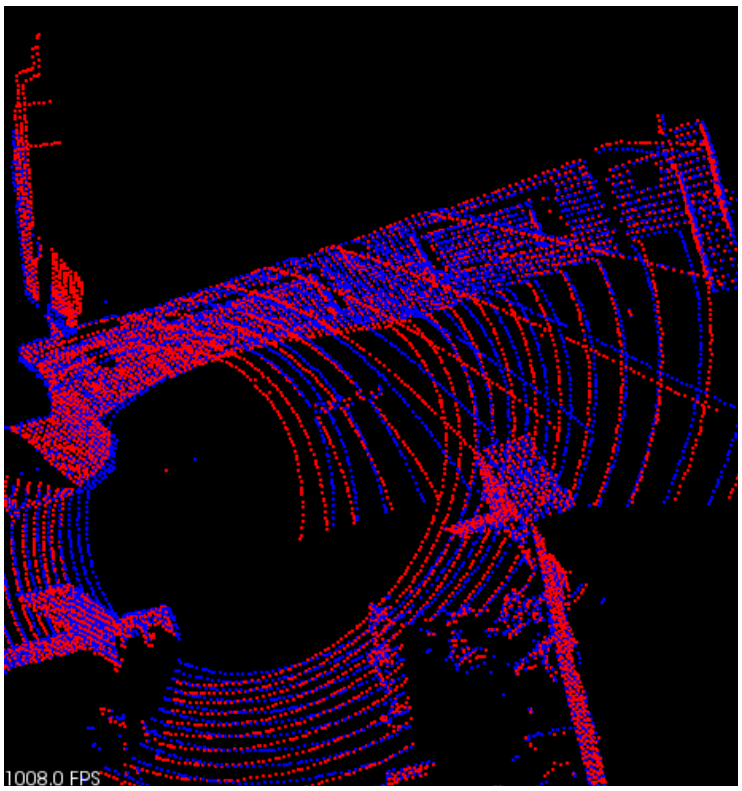


Figure 3.5: Enlarged image of the point cloud registration result.

Table 3.2: Execution time of 10 runs of the scan registration approaches [s].

	scaled ICP	sNDT	NDT
3D-curve	0.135	0.037	0.023
cloud-office	1.126	0.425	0.269
KITTI-07	0.788	0.240	0.205

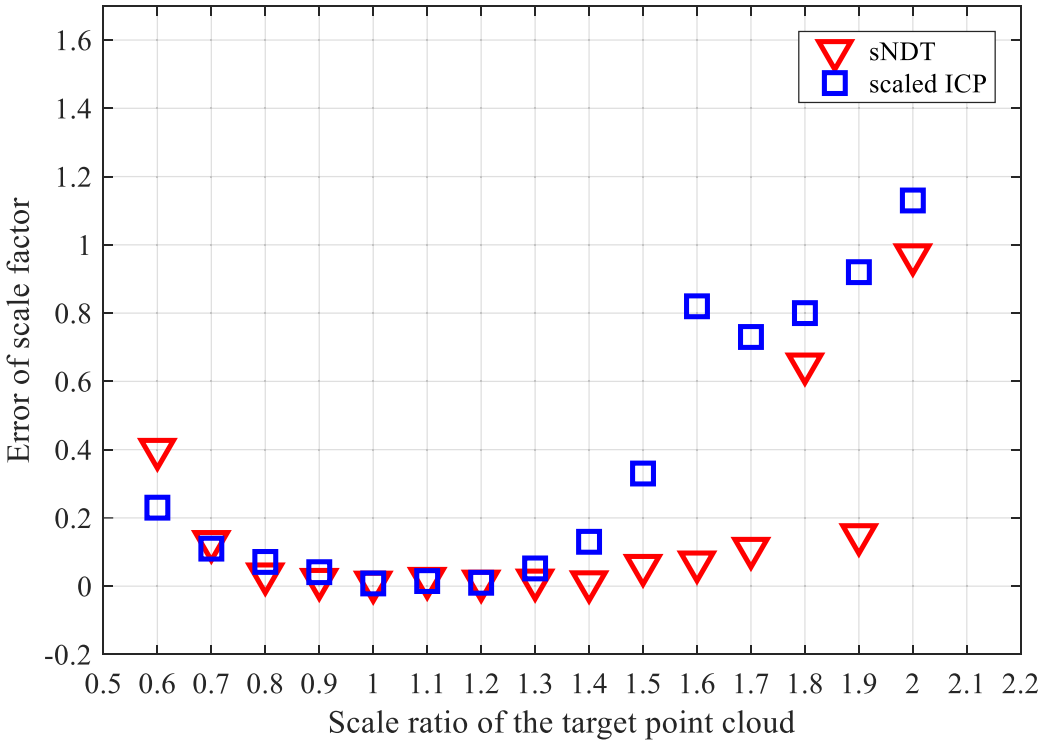


Figure 3.6: Scale error of the tests on *cloud-office*.

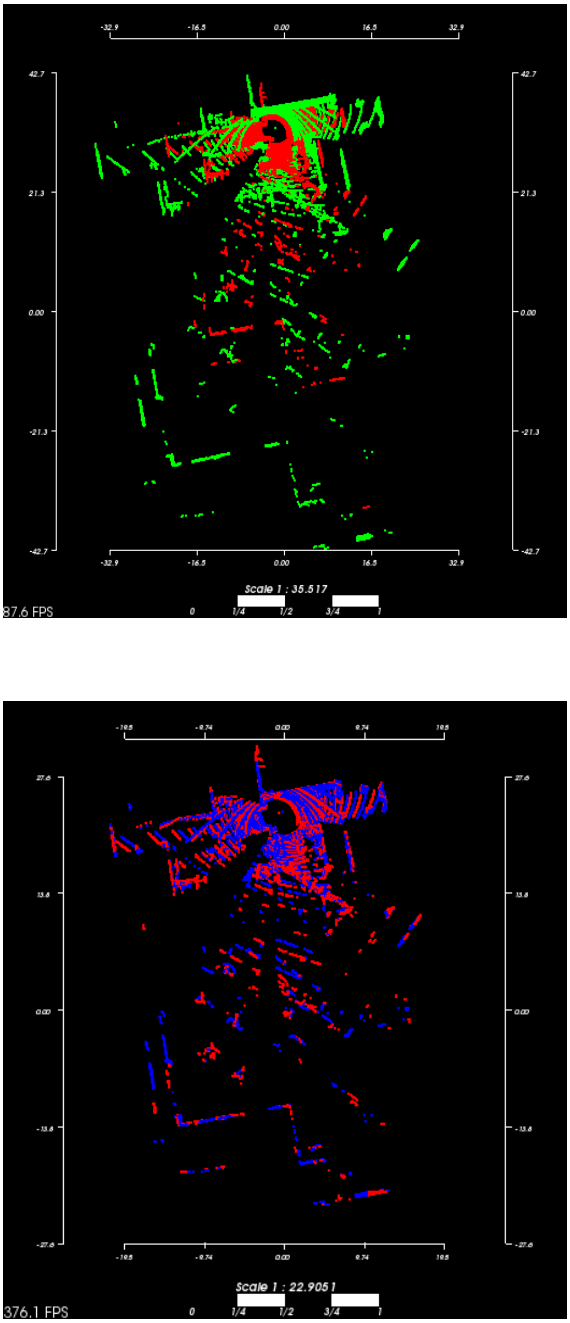


Figure 3.7: The scale registration result of *cloud-office* when the re-scale ratio of the source point cloud is 1.6. Red: reference scan. Green: source scan. Blue: aligned scan.

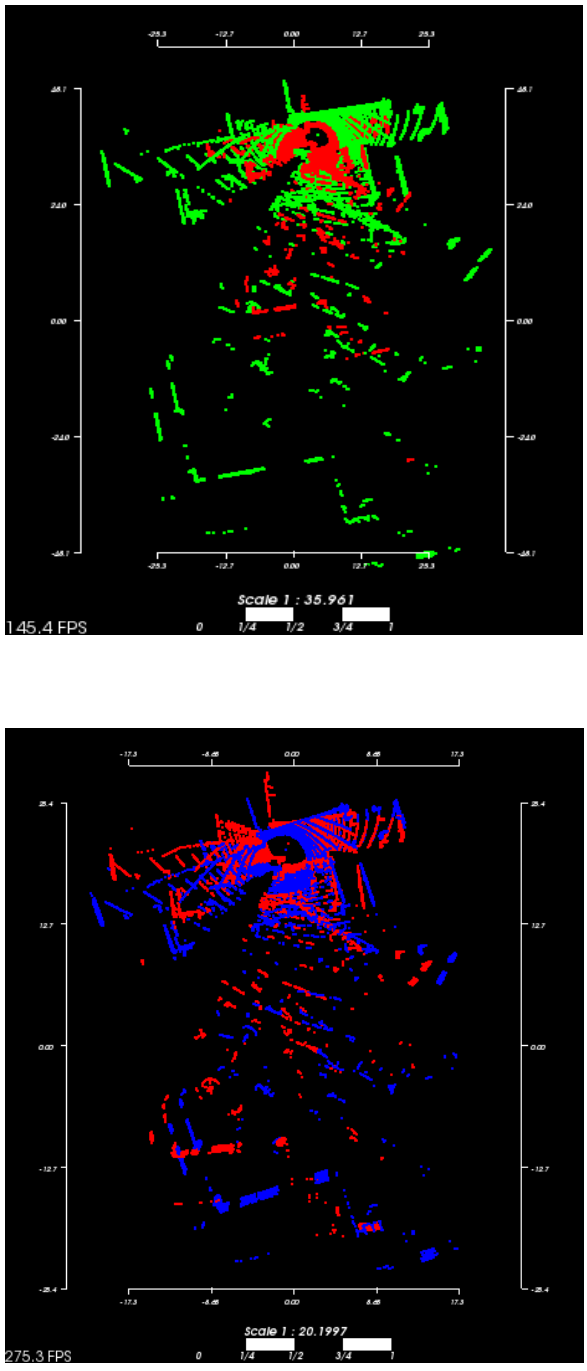


Figure 3.8: The scale registration result of *cloud-office* when the re-scale ratio of the source point cloud is 1.9. Red: reference scan. Green: source scan. Blue: aligned scan. It can be seen that the estimated scale is roughly accurate, however, there exists a notable error in the rotation estimation.

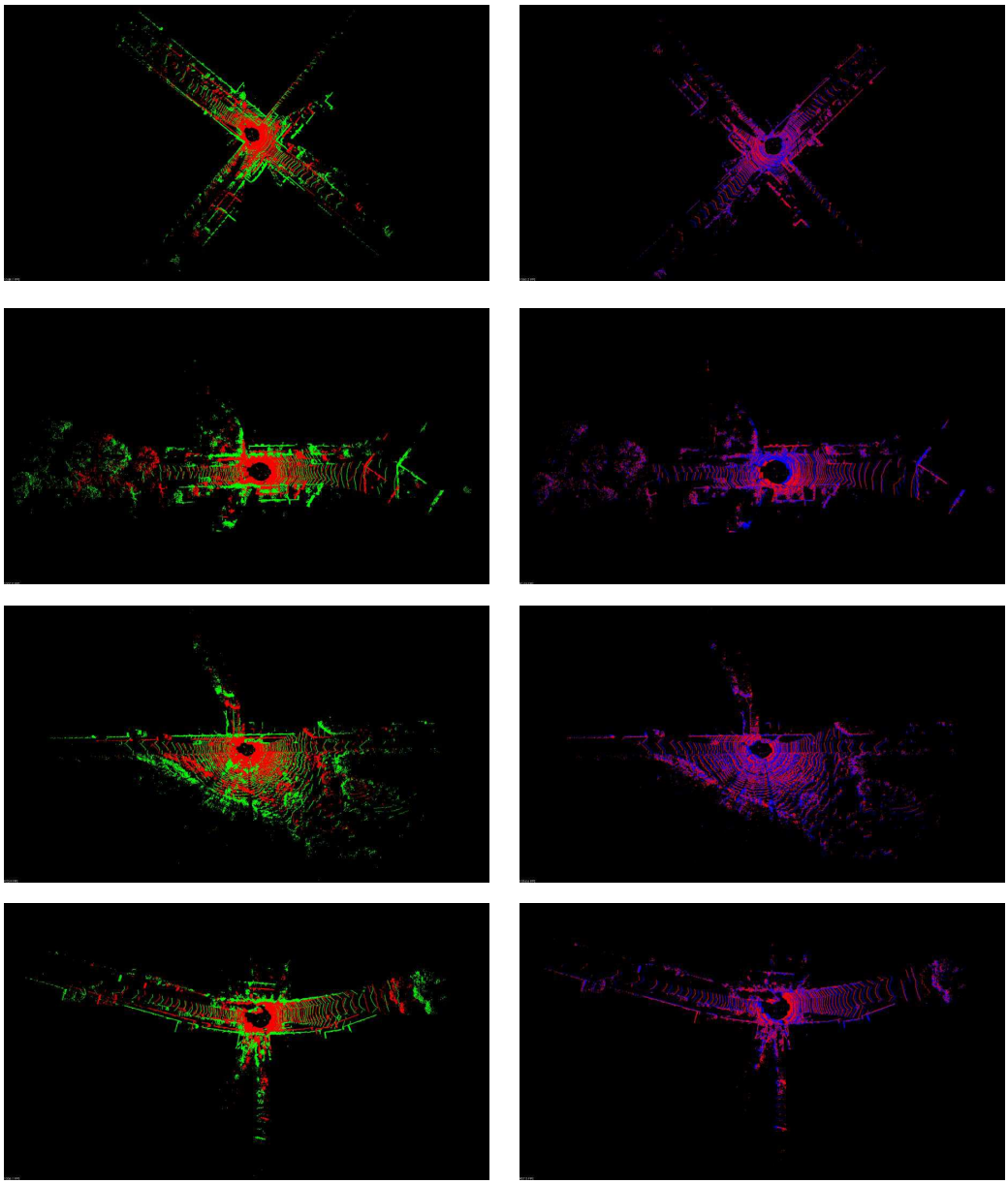


Figure 3.9: Some registration results on the *KITTI-07*. The left column includes the point pairs before aligning, while the point clouds after aligning are shown in the right column. The reference point clouds, the source point cloud with incorrect scale factor, and the aligned source point cloud are represented by red, green, and blue, respectively.

3.3.2 Evaluation of Localization in Scale Drifted Point Cloud Map

In the previous subsection 3.2, all the selected scan pairs are extracted from individual scans. However, in a robot localization procedure, the point cloud is likely to be Scan-to-Map alignment instead of Scan-to-scan matching. The problem becomes more complex since the reference point cloud is a map with much larger size and there may be plenty of outliers. To test whether the sNDT is able to handle the localization in the scale drifted map. In this subsection, a very simple localization system is built based on sNDT and tested on a public dataset.

Even though the 3D registration is often reliable to provide a good estimate, however, it may still fails when large rotations between frames happen. Therefore, when the robot navigates in the map, the uncertainty of the robot state needs to be approximated. Kalman filter (KF) is a powerful tool for modelling the uncertainty of robot and calculate the robot state with the maximum probability. Since the Kalman filter is a linear filter while the transformation of robot states is frequently highly non-linear, the extended Kalman filter (EKF) and the unscented Kalman filter (UKF) are the most widely used methods in the practical application [72].

In this test, the back end of the proposed localization method is based on the UKF, which was first proposed in [73]. In the UKF algorithm, the state distribution is propagated through selecting a set of sampling points (sigma points) around the linearization point. Consider the robot pose at time t as $\hat{x}_t$ and the observation as z_t, the non-linear procedure of robot pose estimation can be described as:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_t \\ z_{t+1} &= h(x_t, m) + v_t, \end{aligned} \tag{3.16}$$

where u_t is the input of the system. f and h are motion and observation functions, respectively. v_t and w_t are zero-mean Gaussian noises. The localization system is initialized by a estimated initial state vector x_0 and its covariance matrix P_0.

In order to update the statics, a set of sigma points $\mathcal{X}_t$ are selected by:

$$\mathcal{X}_t^+ = [\hat{x}_t^+, \hat{x}_t^+ \pm \sqrt{(n_x) + \kappa)P_t^+}], \tag{3.17}$$

where κ is a design parameter for scaling the sigma points. n_x is the dimension of the state vector.

Via updating the sigma points based on the non-linear odometry model:

$$\mathcal{X}_{t+1}^- = f(\mathcal{X}_t^+, u_t), \tag{3.18}$$

the mean and covariance of the prior is updated:

$$\begin{aligned} \hat{x}_{t+1}^- &= \sum_{i=0}^{2n_x} W_{i,t+1}\mathcal{X}_{i,t+1}^- \\ P_{t+1}^- &= \sum_{i=0}^{2n_x} W_{i,t+1}(\mathcal{X}_{i,t+1}^- - \hat{x}_{t+1}^-)(\mathcal{X}_{i,t+1}^- - \hat{x}_{t+1}^-)^T + Q_t, \end{aligned} \tag{3.19}$$

where Q_t is the covariance matrix of w_t, $W_{i,t+1}$ is the weight of the sigma point $\mathcal{X}_{i,t+1}^-$, which is computed as:

$$W_i = \frac{1}{n_x + \kappa}(\kappa, \frac{1}{2}, ..., \frac{1}{2}) \tag{3.20}$$

Then, in the correction step, the sigma points generated previously are reused in order to measure the observation noise:

$$\hat{z}_{t+1}^- = h(\mathcal{X}_{i,t+1}^-, m)P_z = \sum_{i=0}^{2n_x} W_i(z_{t+1}^- - \sum_{i=0}^{2n_x} W_i z_{t+1}^-)(z_{t+1}^- - \sum_{i=0}^{2n_x} W_i z_{t+1}^-)^T + R_{t+1} \tag{3.21}$$

R_t is the covariance matrix of v_t.

The cross covariance between the state propagation and observation is obtained by:

$$P_{xz} = \sum_{i=0}^{2n_x} W_i(\mathcal{X}^-_{i,t+1} - \hat{x}^-_{t+1})(z^-_{t+1} - \sum_{i=0}^{2n_x} W_i z^-_{t+1})^T + R_{t+1} \tag{3.22}$$

Then the state estimation goes into a normal Kalman filter process, in which the Kalman gain is calculated by:

$$K_{t+1} = P_{xz} p_z^{-1} \tag{3.23}$$

The estimation of the mean and covariance is given by:

$$\begin{aligned} \hat{x}^+_{t+1} &= \hat{x}^-_{t+1} + K_{t+1}(z_{t+1} - \hat{z}^-_{t+1}) \\ P^+_{t+1} &= P^-_{t+1} - K_{t+1} P_z K^T_{t+1} \end{aligned} \tag{3.24}$$

As a pure laser-based localization framework, extra sensors, such as IMU and encoders, are not employed in this test. Two point cloud registration threads are performed at the same time for different purposes. The first one is used to calculate the rigid transformation between two joint laser frames in order to provide a rough prediction of the robot's motion. The second one is the proposed sNDT, which corrects the updated statics, including the mean and variance of estimated robot pose. The structure of the localization method is as shown in the Figure 3.10.

The dataset used in this test is the KITTI 00 [74]. First, a point cloud map is reconstructed through ORB-SLAM2 [1] based on the monocular images, then the Velodyne Lidar data are used for performing localization. Unlike the extended MCL in the previous chapter, the sNDT-based localization is sensitive to the initial

value. Therefore, the localization algorithm is initialized with a comparatively accurate robot state.

The estimated trajectory is shown in Fig. 3.12, two groups of alignment examples are depicted in Fig. 3.11. It may be noticed that, though there are great disagreements between the scan and the point cloud map, the sNDT can still provide a good estimation of the similarity transformation.

Note that, the motion model can not provide an estimation of the scale factor that the value of s_t only depends on the observation. Therefore, this method is not robust enough against the failure of scan registration. However, the main focus of this test is to exam the behaviour of the sNDT rather than designing a localization system and the result also shows that the sNDT is able to handle the point cloud alignment between Lidar scan and the sparse point cloud map. Furthermore, this motion model is also compatible with other types of odometries to improve the accuracy.

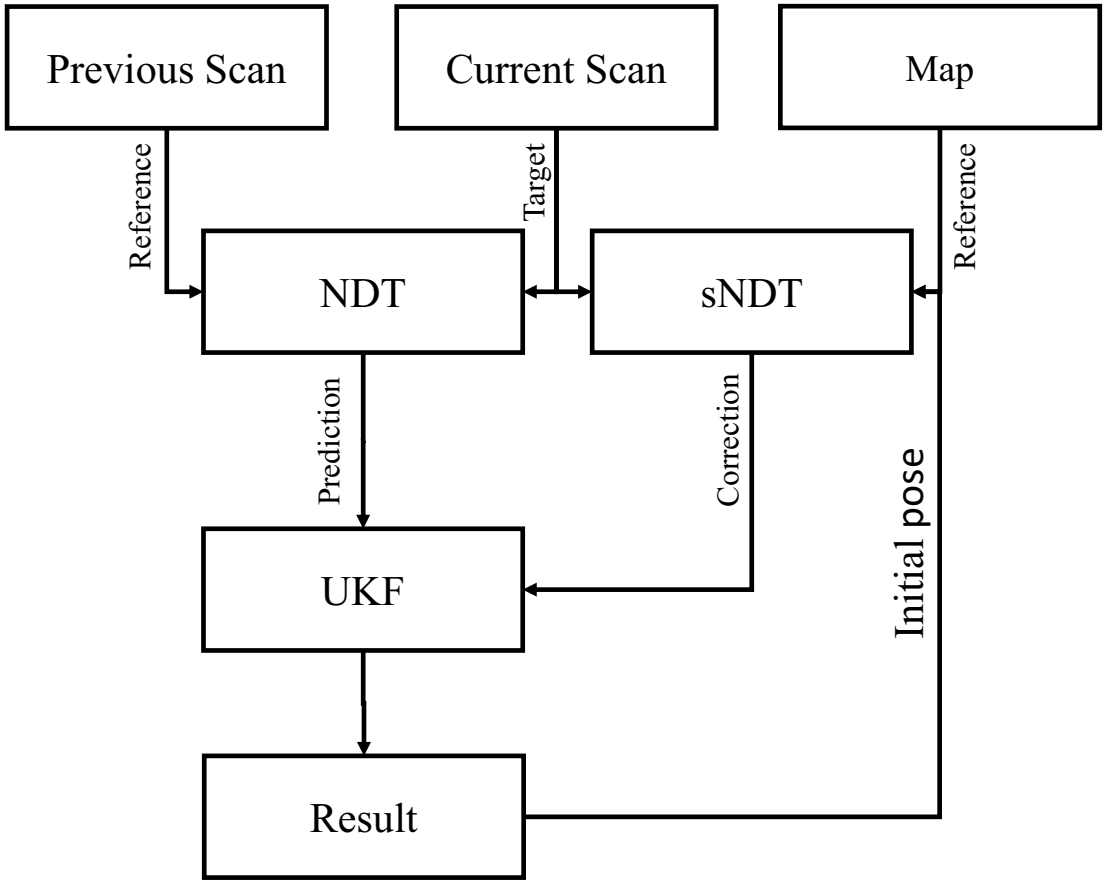


Figure 3.10: Diagram of the localization approach operating with pure Lidar scan. The previous and reference scans are captured by the Lidar sensor. The map is represented by a point cloud built by monocular SLAM.

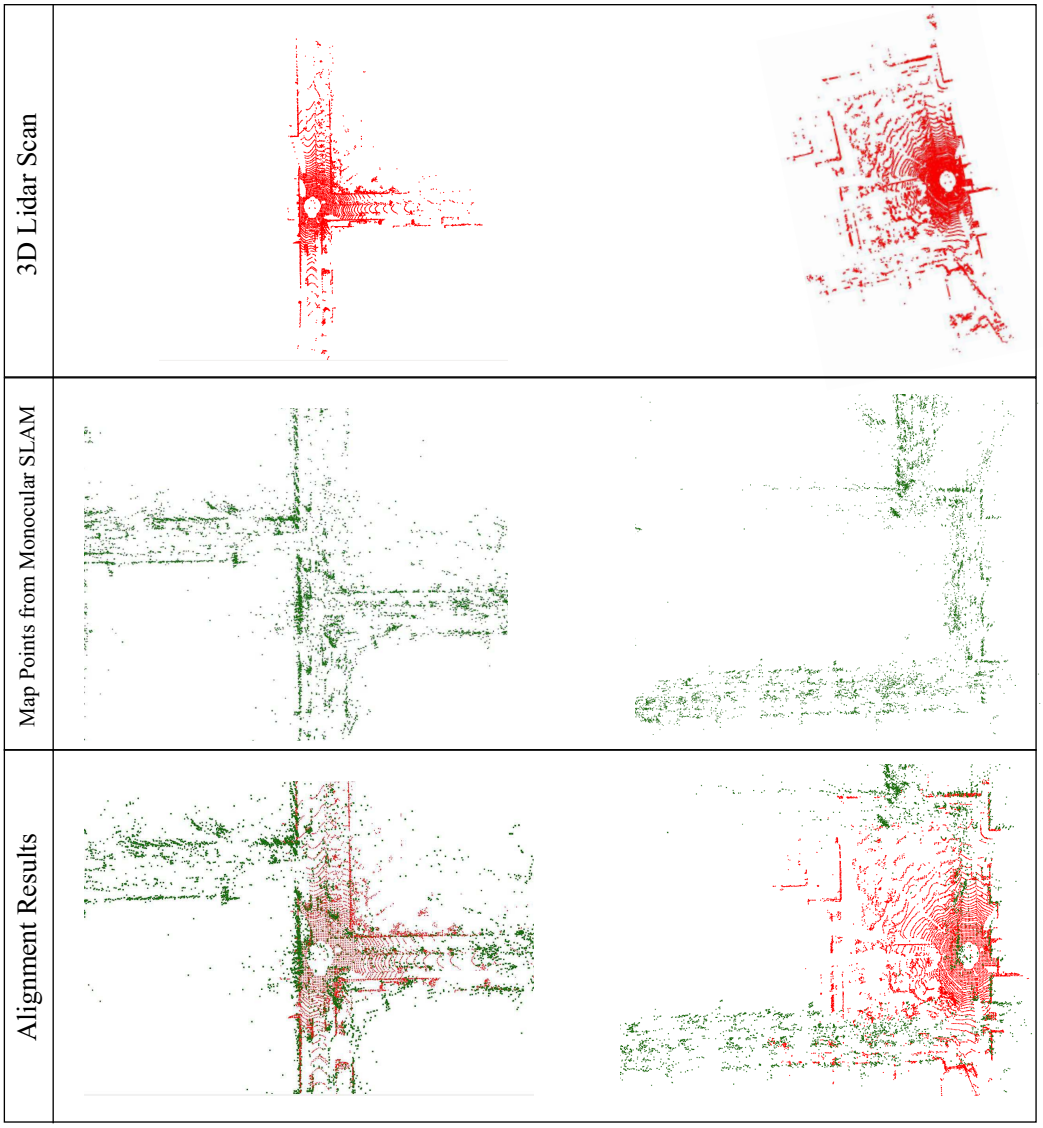


Figure 3.11: Alignment between the Lidar scan (first row) and the map from monocular SLAM (second row). It can be seen that, there are significant differences in the shape of point cloud between the scan and the map, namely, the Lidar scan can describe the environment more completely than the monocular camera due to its larger FOV. However, the proposed point cloud registration can still handle this situation even the scale is unknown as well.

Figure 3.12: Trajectory of the pure-Lidar-based localization result on KITTI 00 dataset. The map (green) is built by ORB-SLAM2 [1].

3.4 Conclusions

In this section, a new similarity transformation registration method called sNDT is introduced. The proposed method is improved on the basis of the the standard 3D NDT algorithm. Newton's method is used to optimized both the relative scale factor between the source and target point cloud. The performance of sNDT is evaluated based on several group of point pairs compared with the scaled ICP. If the scale of the source point set only drifts slightly, the sNDT and scaled ICP is conducted with the similar performance. When the points are sparse and the scale ratio of source point cloud is very closed to 1.0, the scaled ICP slightly

outperforms the sNDT. Otherwise, the sNDT shows better performance than the scaled ICP in term of the time cost and convergence.

In conclusion, the reason why sNDT is a more suitable method to calculate the relative pose of the 3D Lidar scans in the map in contrast with ICP is listed as follow:

- **The ability of outlier rejection.** In a ICP framework, every point makes contribution to the sum of squared distances. If the source and reference point cloud can perfectly match each other, the ICP would be able to be well converged. However, for localization, the map and the laser scan always disagree with each other. Especially in the situation where the map is built by the visual SLAM using a camera whose FOV is less than 150 degrees, and the sensor used for localization is a 360-degree Lidar. Therefore, the map can only partially depict the environment. The paper [75] provides an evaluation and comparison of the performance of standard ICP and standard NDT for autonomous vehicle localization under various environments with changes, and the conclusion is that NDT possesses better ability to handle the dynamic environmental changes.

- **The algorithm running speed.** On account that a slow localization method will lead to data loss incidents that decrease the robustness of localization system, another concern is real-time requirement. The time complexity of the original ICP is $O(N^2)$. In the fast ICP proposed in [76], the time complexity of the ICP method can be reduced to $O(logN)$ by associating the closest point pairs using KD-tree. As a contrast, since the normal distribution of the reference point cloud only need to be calculated once, the NDT method can be performed with $O(N)$. Consequently, if the sensor is a 2D laser scanner or a tilted one, there is no obvious difference between the

ICP and NDT methods in real time performance. However, at present, the main solution of 3D sensor is the multi-line Lidar, with which the sampling number of points can be up to 2.8 million. The speed of NDT is far superior to the ICP method that the ICP-based laser odometry can hardly catch up the speed of laser refresh

– **The convergence performance of scale estimation.** Since we need to recover the scale factor as well as the 6DoF transformation, the method should be able to handle the scale stretching. Convergence analysis of different error conditions is provided in this chapter, in which the convergence performance of the proposed sNDT outperforms the ICP when the source point cloud is with a large re-scaled ratio.

In order to verify the performance of sNDF for map-scan alignment, a test on the map built by monocular SLAM over the KITTI 00 dataset is conducted. The UKF method is utilized as the back-end for estimating the inconsistency of the state estimation. The result shows that the sNDT is able to handle the alignment between the point cloud map and a single scan.

Chapter 4
Conclusion and Future Works

Currently, monocular SLAM is a popular solution for the real-time environment reconstruction. However, on account of the absence of a fixed baseline between frames, the scale of the map built by monocular SLAM is unknown and inconsistent. In this study, the main focus is on solving the problem of Lidar localization on such a metrically inconsistent map. A modified Monte Carlo localization method (extended MCL) to track the pose and local scale of robot in this monocular-SLAM-built map using the 2D Lidar. The proposed method has been tested on a public dataset. Experiment result shows that good localization can be achieved while the scale can also be estimated correctly most of the time. Thus the proposed method can be regarded as an extension to the cooperative robot systems equipped with different types of sensors.

Since the map built by monocular SLAM is a 3D one and the 2D lidar utilized in the extended MCL can only provide 2D range data, the extended MCL can hardly obtain the results with high resolution due to the lost information. Nowadays, there is a trend that the 3D Lidar is becoming the mainstream sensor in many fields, in this work, a 3D scan registration method named sNDT is proposed in order to perform high quality scale-aware scan matching. Using the normal distribution transform representation for point cloud alignment and scale estimation, stretching the scale of the point cloud can be achieved as well as the

rigid transformation alignment.

The usage scenarios of the two proposed methods (extended MCL and sNDT) are similar, in which a Lidar is used for environment perception while a monocular-SLAM-built map is utilized as the prior map. However, they have their own advantages and drawbacks under different working conditions, respectively. Firstly, the sensor used for the extended MCL method is a 2D laser scanner, while the sNDT can deal with the 6DoF pose estimation with a 3D omnidirectional Lidar. The extended MCL method is able to handle the global localization and shows robust performance against initial error. Whereas the sNDT is more accurate than the motion model in MCL method, but also more sensitive to the initial value of robot state, hence the global localization function is also absent in the sNDT-based localization method.

There are still multiple directions to be explored to improve the methods in this study. For example, a more robust back-end can be combined with the sNDT to enhance the localization performance. Additionally, given the position of Lidar sensor in the map built by monocular SLAM, the map built by the Lidar-based SLAM and monocular SLAM can be linked. One of the ideas worth trying is to combine the proposed sNDT method with the MCL method on account that the work [77] shows that transforming the map into the NDT space can improve the accuracy of the MCL method.

References

[1] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular slam," in *European Conference on Computer Vision*, pp. 834–849, Springer, 2014.

[3] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[4] M. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard, "The mit stata center dataset," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1695–1699, 2013.

[5] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3, pp. 2743–2748, IEEE, 2003.

[6] S. Ahn, J. Choi, N. L. Doh, and W. K. Chung, "A practical approach for ekf-slam in an indoor environment: fusing ultrasonic sensors and stereo camera," *Autonomous robots*, vol. 24, no. 3, pp. 315–335, 2008.

[7] A. A. Ravankar, Y. Hoshino, A. Ravankar, L. Jixin, T. Emaru, and Y. Kobayashi, "Algorithms and a framework for indoor robot mapping in a noisy environment using clustering in spatial and hough domains," *International Journal of Advanced Robotic Systems*, vol. 12, no. 3, p. 27, 2015.

[8] J. Lv, Y. Kobayashi, T. Emaru, and A. A. Ravankar, "Indoor slope and edge detection by using two-dimensional ekf-slam with orthogonal assumption," *International Journal of Advanced Robotic Systems*, vol. 12, no. 4, p. 44, 2015.

[9] Y.-T. Wang, C.-C. Peng, A. Ravankar, and A. Ravankar, "A single lidar-based feature fusion indoor localization algorithm," *Sensors*, vol. 18, no. 4, p. 1294, 2018.

[10] A. Gawel, R. Dubé, H. Surmann, J. Nieto, R. Siegwart, and C. Cadena, "3D registration of aerial and ground robots for disaster response: An evaluation of features, descriptors, and transformation estimation," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pp. 27–34, IEEE, 2017.

[11] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The international journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[12] F. Dellaert, S. M. Seitz, C. E. Thorpe, and S. Thrun, "Structure from motion without correspondence," in *Proceedings IEEE Conference on Computer*

Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662), vol. 2, pp. 557–564, IEEE, 2000.

[13] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 1052–1067, 2007.

[14] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 3565–3572, IEEE, 2007.

[15] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pp. 83–86, IEEE, 2009.

[16] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[17] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[18] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*, pp. 2320–2327, IEEE, 2011.

[19] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[20] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2174–2181, IEEE, 2015.

[21] Z. Zhang, R. Zhao, E. Liu, K. Yan, and Y. Ma, "Scale estimation and correction of the monocular simultaneous localization and mapping (slam) based on fusion of 1d laser range finder and vision data," *Sensors*, vol. 18, no. 6, p. 1948, 2018.

[22] S. Shen and N. Michael, "State estimation for indoor and outdoor operation with a micro-aerial vehicle," in *Experimental Robotics*, pp. 273–288, Springer, 2013.

[23] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[24] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, no. 1, pp. 3–20, 2006.

[25] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth to depth conversion for monocular slam," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pp. 2778–2783, IEEE, 2007.

[26] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," *Robotics: Science and Systems VI*, vol. 2, no. 3, p. 7, 2010.

[27] H. Strasdat, *Local accuracy and global consistency for efficient visual SLAM*. PhD thesis, Department of Computing, Imperial College London, 2012.

[28] S. Song, M. Chandraker, and C. C. Guest, "Parallel, real-time monocular

visual odometry," in *2013 ieee international conference on robotics and automation*, pp. 4698–4705, IEEE, 2013.

[29] D. P. Frost, O. Kähler, and D. W. Murray, "Object-aware bundle adjustment for correcting monocular scale drift," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4770–4776, IEEE, 2016.

[30] A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid, "Dense reconstruction using 3d object shape priors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1288–1295, 2013.

[31] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Autonomous Robots*, vol. 19, no. 1, pp. 53–65, 2005.

[32] F. Martinelli, "Robot localization: comparable performance of ekf and ukf in some interesting indoor settings," in *2008 16th Mediterranean Conference on Control and Automation*, pp. 499–504, IEEE, 2008.

[33] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings*, vol. 2, pp. 1322–1328, IEEE, 1999.

[34] G. Chen, J. Kua, S. Shum, N. Naikal, M. Carlberg, and A. Zakhor, "Indoor localization algorithms for a human-operated backpack system," in *3D Data Processing, Visualization, and Transmission*, p. 3, 2010.

[35] J. Borenstein, H. Everett, L. Feng, *et al.*, "Where am i? sensors and methods for mobile robot positioning," *University of Michigan*, vol. 119, no. 120, p. 27, 1996.

[36] D. Fernandez and A. Price, "Visual odometry for an outdoor mobile robot," in *IEEE Conference on Robotics, Automation and Mechatronics, 2004.*, vol. 2, pp. 816–821, IEEE, 2004.

[37] S. Koenig and R. G. Simmons, "Passive distance learning for robot navigation," in *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, pp. 266–274, Morgan Kaufmann Publishers Inc., 1996.

[38] R. Kümmerle, B. Steder, C. Dornhege, A. Kleiner, G. Grisetti, and W. Burgard, "Large scale graph-based SLAM using aerial images as prior information," *Autonomous Robots*, vol. 30, no. 1, pp. 25–39, 2011.

[39] M. Mielle, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "SLAM autocomplete: completing a robot map using an emergency map," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, pp. 35–40, IEEE, 2017.

[40] F. Boniardi, A. Valada, W. Burgard, and G. D. Tipaldi, "Autonomous indoor robot navigation using a sketch interface for drawing maps and routes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2896–2901, IEEE, 2016.

[41] F. Boniardi, B. Behzadian, W. Burgard, and G. D. Tipaldi, "Robot navigation in hand-drawn sketched maps," in *2015 European Conference on Mobile Robots (ECMR)*, pp. 1–6, IEEE, 2015.

[42] T. Korah, S. Medasani, and Y. Owechko, "Strip histogram grid for efficient lidar segmentation from urban environments," in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 74–81, IEEE, 2011.

[43] R. D. Morton and E. Olson, "Positive and negative obstacle detection using the HLD classifier," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1579–1584, IEEE, 2011.

[44] R. Goeddel, C. Kershaw, J. Serafin, and E. Olson, "FLAT2D: Fast localization from approximate transformation into 2d," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1932–1939, IEEE, 2016.

[45] R. W. Wolcott and R. M. Eustice, "Fast lidar localization using multiresolution gaussian mixture maps," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 2814–2821, IEEE, 2015.

[46] R. W. Wolcott and R. M. Eustice, "Robust lidar localization using multiresolution gaussian mixture maps for autonomous driving," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 292–319, 2017.

[47] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[48] K.-H. Lin, C.-H. Chang, A. Dopfer, and C.-C. Wang, "Mapping and localization in 3d environments using a 2d laser scanner and a stereo camera.," *Journal of information science and engineering*, vol. 28, no. 1, pp. 131–144, 2012.

[49] A. Ravankar, A. Ravankar, Y. Kobayashi, and T. Emaru, "Symbiotic navigation in multi-robot systems with remote obstacle knowledge sharing," *Sensors*, vol. 17, no. 7, p. 1581, 2017.

[50] A. Ravankar, A. Ravankar, Y. Kobayashi, and T. Emaru, "Hitchhiking robots: A collaborative approach for efficient multi-robot navigation in indoor environments," *Sensors*, vol. 17, no. 8, p. 1878, 2017.

[51] B. Xu, W. Jiang, J. Shan, J. Zhang, and L. Li, "Investigation on the weighted ransac approaches for building roof plane segmentation from lidar point clouds," *Remote Sensing*, vol. 8, no. 1, p. 5, 2015.

[52] B. Behzadian, P. Agarwal, W. Burgard, and G. D. Tipaldi, "Monte carlo localization in hand-drawn maps," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4291–4296, IEEE, 2015.

[53] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, pp. 713–720, 2002.

[54] N. L. Johnson, A. W. Kemp, and S. Kotz, *Univariate discrete distributions*, vol. 444. John Wiley & Sons, 2005.

[55] A. Milstein, J. N. Sánchez, and E. T. Williamson, "Robust global localization using clustered particle filtering," in *AAAI/IAAI*, pp. 581–586, 2002.

[56] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[57] S. Wang, Y. Kobayashi, A. Ravankar, A. A. Ravankar, and T. Emaru, "Localization with laser range finder in a metrically inconsistent map from monocular slam," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 796–801, IEEE, 2018.

[58] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "Robust robot localization in a complex oil and gas industrial environment," *Journal of Field Robotics*, vol. 35, no. 2, pp. 213–230, 2018.

[59] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "Normal distributions transform monte-carlo localization (ndt-mcl)," in *2013 IEEE/RSJ*

International Conference on Intelligent Robots and Systems, pp. 382–389, IEEE, 2013.

[60] A. Ravankar, A. Ravankar, Y. Kobayashi, Y. Hoshino, and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, p. 3170, 2018.

[61] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," *Proc. Int. Joint Con on Artificzul Intelligence*, 1981.

[62] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611, pp. 586–607, International Society for Optics and Photonics, 1992.

[63] S. Du, N. Zheng, S. Ying, Q. You, and Y. Wu, "An extension of the icp algorithm considering scale factor," in *2007 IEEE International Conference on Image Processing*, vol. 5, pp. V–193, IEEE, 2007.

[64] S. Ying, J. Peng, S. Du, and H. Qiao, "A scale stretch method based on icp for 3d data registration," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 3, pp. 559–565, 2009.

[65] J. Dong, Y. Peng, S. Ying, and Z. Hu, "LieTrICP: An improvement of trimmed iterative closest point algorithm," *Neurocomputing*, vol. 140, pp. 67–76, 2014.

[66] A. Makovetskii, S. Voronin, V. Kober, and D. Tihonkih, "Affine registration of point clouds based on point-to-plane approach," *Procedia engineering*, vol. 201, pp. 322–330, 2017.

[67] M. Magnusson, A. Nuchter, C. Lorken, A. J. Lilienthal, and J. Hertzberg, "Evaluation of 3d registration reliability and speed-a comparison of icp and ndt," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3907–3912, IEEE, 2009.

[68] P. Biber, S. Fleck, and W. Strasser, "A probabilistic framework for robust and accurate matching of point clouds," in *Joint Pattern Recognition Symposium*, pp. 480–487, Springer, 2004.

[69] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: A general framework for (hyper) graph optimization," *Tech. Rep.*, 2011.

[70] R. B. Rusu and S. Cousins, "Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*, pp. 1–4, 2011.

[71] H. Merten, "The three-dimensional normal-distributions transform," *threshold*, vol. 10, p. 3, 2008.

[72] A. Giannitrapani, N. Ceccarelli, F. Scortecci, and A. Garulli, "Comparison of ekf and ukf for spacecraft localization via angle measurements," *IEEE Transactions on aerospace and electronic systems*, vol. 47, no. 1, pp. 75–84, 2011.

[73] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[74] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[75] S. Pang, D. Kent, X. Cai, H. Al-Qassab, D. Morris, and H. Radha, "3d scan registration based localization for autonomous vehicles-a comparison of ndt

and icp under realistic conditions," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pp. 1–5, IEEE, 2018.

[76] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-icp: A globally optimal solution to 3d icp point-set registration," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 11, pp. 2241–2254, 2015.

[77] R. Valencia, J. Saarinen, H. Andreasson, J. Vallvé, J. Andrade-Cetto, and A. J. Lilienthal, "Localization in highly dynamic environments using dual-timescale ndt-mcl," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3956–3962, IEEE, 2014.