Abstract

Kidnapping Detection and its Recovery in
Simultaneous Localization and Mapping

by

TIAN Yang

Doctor of Philosophy in Department of Robotics

Ritsumeikan University, Biwako-Kusatsu Campus

Professor MA Shugen, Chair

Mobile robots that autonomously execute tasks have been applied in various environments, such as factories, hospitals and houses. One of the basic fundamental functions required by the autonomous robot is localization. Localization is the solution to estimate the robot posture (position and orientation) in the world coordinates provided a known map of the environment. In an unknown environment, where both the robot posture and the environment map are unknown, a technique called "Simultaneous Localization and Mapping (SLAM)" was proposed. Therein, a consistent map of the environment is built while simultaneously determining the robot posture within this map with data gathered by proprioceptive and exteroceptive sensors. While the mobile robot performs tasks autonomously, an unexpected movement, such as being moved to another place by a human, may happen due to the interaction with surroundings. This problem is known as "kidnapping" problem, which obstructs the robot to track its desired posture correctly within the environment map. Previous solutions of the kidnapping problem execute global localization within the map directly and periodically to obtain the correct robot posture. However, these solutions are not applicable to the robots kidnapped in an unknown environment due to unique situations of the kidnapping.

In this study, we found that there are two situations after kidnapping happened in an unknown environment. One situation is that the robot is kidnapped to the explored area

during processing SLAM. In this situation, the map of the explored area is available to be utilized to perform global localization as a solution. Another situation is that the robot is kidnapped to a new area lacking an environment map. A new SLAM process should be carried out to realize localization in a new map. Since the solution cannot be determined before ascertaining under which situation the robot is, solutions about global localization cannot be directly applied. We thus propose two algorithms (kidnapping detection and kidnapping recovery) for the kidnapping solution in SLAM. Furthermore, the kidnapping categories are classified for increasing the efficiency of the solution.

A kidnapping detection framework, performing two checks before and after the update process which is called "Double Kidnapping Detection and Recognition (DKDR)" utilizes different metrics to detect the kidnapping and recognize the type of kidnapping in real time. To explain one of the principles of DKDR, we describe a property of filter-based SLAM that corrects the mapping result of the environment using the current observations after the update process. The Extend Kalman Filter SLAM (EKF-SLAM) and the Particle Filter SLAM (PF-SLAM) are modified in simulations and experiments to show that DKDR can be simply and widely applied in existing filter-based SLAM algorithms. Furthermore, an improved detection method called "Probabilistic Double Kidnapping Detection and Recognition (P-DKDR)" is proposed by combining probability of features positions and the robot posture to reduce false alarms in a large-scale environment. Simulations and experiments were conducted for the comparison between DKDR and P-DKDR.

Based on the results from kidnapping detection, two kidnapping recovery methods are proposed to solve different types of kidnapping. If the robot is kidnapped to the explored area, such as being stuck or slip around a spot, an improved Mentor Carol Localization (MCL) method called "MCL with Map Uncertainty (MCL-MU)", which takes the map uncertainty produced by SLAM into the method, is applied to relocalize the robot posture within the environment map. Furthermore, the dispersive area of the particles in MCL is limited in a range to increase the efficiency of the method. In the situation that the robot is moved out of the sensor range, a method called "Known and Unknown Environment's

Simultaneous Localization (KUESL)" is proposed to execute a global localization and a SLAM simultaneously to determine robot posture according to the probability of estimated results. Experiments were conducted to show their validity and performance.

With discussing the simulations and experiments results, we found that several failure cases exist due to the specification of sensors and specific environments, such as the reflection problem of laser range finder in different objects or the robot is kidnapped to a similar environment with the environment before the kidnapping. Besides these specific situations, the proposed kidnapping detection and kidnapping recovery methods can detect and recover the kidnapping events effectively.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to dedicate my sincere gratitude to my supervisor, Prof. Shugen Ma, of the Department of Robotics in Ritsumeikan University, for his patient guidance, continuous support, enthusiastic encouragement on my Ph.D research and other research. His teaching will stay in my mind forever.

Besides my supervisor, I would like to thank Prof. Ryuta Ozawa and Prof. Atsushi Kakogawa for their valuable suggestions and insightful comments on my research in seminars, but also for the questions incented me to perfect my research from various perspectives.

I am grateful to all my colleagues in Ma laboratory, to Dr. Yi Sun, Mr. Dingxin Ge, Mrs. Yongchen Tang, Dr. Yang Yang, Dr. Chao Ren, Dr. Norzalilah Binti Mohamad Nor, Mr. Wenbin Tang, Mr Jie Ma, Mr. Yayi Shen who gave me great help and many useful comments for my research, to Mr.Fabian Eugenio Reyes Pinner, Dr. Guoteng Zhang who helped me greatly in the lab, to Dr. Takahiro Matsuno, Mr. Gomez Aladro Victor Antonio, Mr. Taijyu Yamagami for their assistance and supporting on mechanism design, prototype fabrication and experiments. Studying with them together in Ritsumeikan University leads me a colorful and fruitful campus life. It is my pleasure to thank Dr. Zhongkui Wang, and Mr. Zhe Qiu for their kind help in my research activities and daily life.

My thankfulness goes to all the professors and officers in the Robotics Department, in the Office of Graduate Studies, and in the Office of International Students at Ritsumeikan University for guiding and helping me during my stay in Japan. Without their help, I wouldn't have been able to fully focus in my research.

Finally, I would like to thank my parents for their unconditional love and support without minding what and where I wished to accomplish my dreams; my beloved family and friends for their encouragement, kind words, messages, support, and special understanding during my study. They always kept being attentive to my health and wellbeing, even from the far away distance.

Thank you very much.

Curriculum Vitæ

TIAN Yang

Education

2002-2005	Qingdao 16th High School (Qingdao, Shandong, China) Upper Secondary School Student
2005-2009	Qingdao University of Technology (Qingdao, Shandong, China) Bachelor Degree of Engineering
2010-2013	Ocean University of China (Qingdao, Shandong, China) Master Degree of Engineering
2013-2018	Ritsumeikan University, BKC (Kusatsu, Shiga, Japan) Doctoral Candidate, Advanced Mechanical Engineering and Robotics

Personal

Born	June 19, 1986, Jilin, China.
Research Interests	SLAM, Localization, Mobile Robot, Sensing.

Chapter 1

Introduction

Millions of functional robots exist in the world, and most of them are industrial arm robots [1]. Different with the success of robotics in the industrial field, mobile robots still have unsolved challenges [2]. The robots that can move on the ground, underwater, through air and space are recognized as the mobile robots. The concentration of this study is concerned with mobile robots moving on the ground although it can be applied any of these environments.

1.1 Localization

Localization is core challenging competence required of mobile robots to operate tasks autonomously. From emergency search and rescue [3, 4], to precision agriculture [5, 6] and underwater exploration [7, 8], there are many applications where it is advantageous to deploy robots.

The purpose of robot localization is to find out the robot location relative to the environment [9, 10]. When we talk about location, pose, or posture we mean the position and orientation of a robot in a global coordinate system. It is hard to operate tasks for the autonomous robot if the robot does not know its location relative to the environment. The

robot will most likely need to have at least some idea of where it is to be able to operate and act successfully.

For obtaining the estimated robot pose, two kinds of measurements are available to provide the basic function of localization. One of them is relative measurements which only looking at the robot itself to acquire measurements [11]. Since external information is not included in the measurements, it only provides information relative to the starting measurement point. Instead, observations get from the environment supply information about the robot pose is called absolute measurements [11]. The absolute measurements only looking at the environment that is independent of any previous robot pose estimation.

1.1.1 Relative Measurement

The relative measurements are referred to as dead reckoning [12], which is a process of estimating the pose only based on the estimated speeds over elapsed time by utilizing the last known position with proprioceptive sensors [11]. It is subject to cumulative errors because the pose estimation is based on earlier poses. Odometry and inertial navigation are two basic techniques to implement dead reckoning.

Odometry works by using the robot model and wheel encoders counting rotation angle of each wheel to measure the distance and heading direction the robot travelled [13, 14]. However, errors in both travelled distance and orientation are accumulated over time due to the drift and slippage of the wheel [15]. Another problem of odometry is that it is easily affected by the situation of the terrain [16]. If the robot is moving on an uneven terrain, the irregularities of the terrain will cause considerable errors that it cannot be detected it. Furthermore, error in wheel diameter also can result in odometry errors that cannot be detected. Although many disadvantages of odometry cause accumulated errors, it is the most simple way to obtain pose estimation information, therefore, it is one of the important sources of information for localization.

Inertial navigation measures rotation and acceleration of the robot by gyroscopes and accelerometers separately [17, 18, 19, 20]. Gyroscopes measured the velocity of orientation

detect small accelerations in orientation, and accelerometers detect accelerations along the axis of the robot coordinates. They also suffer from extensive drift and are sensitive to bumpy ground. Since the pose estimation is acquired by integration of the sensors information, errors are accumulated over time.

1.1.2 Absolute Measurement

Absolute measurements provide information from exteroceptive sensors [11] to robot pose estimation, which is independent of previously estimated robot pose. The robot pose is not derived from the integration of a sequence of sensors measurements but from the current measurement only. Therefore, the error of the estimated robot pose does not increase unbounded, which is different with relative measurements. There are two base methods can acquire absolute measurements, which are landmark-based methods and feature-based methods.

Landmark-based methods detect landmarks in the environment from sensors readings, such as markers, doors and trees [21, 22, 23]. Once landmarks are found, matching with prior known information of the environment is applied to estimate the robot pose. Another group of approaches based on features utilizes geometric shapes of the environment, such as the lines, edges and corners, to estimate the robot pose within a map [24, 25, 26]. The disadvantage of the absolute measurements is that it costs large amounts of processing and sensing power to realize matching.

1.1.3 Multiple Sensors Fusion

Fusing relative measurements and absolute measurements with initial information to estimate robot pose at a certain time is called multiple sensors fusion localization [27, 28, 29] or filter-based localization [30, 31, 32]. In this case, measurements readings coming from different sensors are combined with a filter to obtain a more accurate estimated robot pose. The methods consider uncertainty and confidence of robot pose with a probabilistic approach.

1.2 Simultaneous Localization and Mapping

Simultaneous localization and mapping (SLAM) is an important technique used to determine a robot pose in an unknown environment without a prior map [33, 34, 35, 36, 37]. In SLAM, the robot incrementally builds a consistent map of the environment concurrently while simultaneously estimating its pose within this map. It is more difficult than localization in that the map is known since the map needs to be built along the way.

For explaining the form of SLAM, the following quantities are defined at a time step k:

- x_k: the state vector describing the location and orientation of the vehicle
- u_k: the control vector, applied at time step $k-1$ to drive the robot to a state x_k at time step k
- m_i: a vector describing the location of the ith feature whose true location is assumed time invariant
- z_k : an observation taken from the robot of the location of the feature at time step k.

In addition, the following sets are also defined:

- $X_{0:k} = \{x_0, x_1, \cdots, x_k\} = \{X_{0:k-1}, x_k\}$: the history of vehicle locations
- $U_{0:k} = \{u_1, u_2, \cdots, u_k\} = \{U_{0:k-1}, u_k\}$: the history of control inputs
- $m = \{m_1, m_2, \cdots, m_n\}$: the set of features
- $Z_{0:k} = \{z_1, z_2, \cdots, z_k\} = \{Z_{0:k-1}, z_k\}$: the set of feature observations.

The core of SLAM is a state estimation problem. In probabilistic form, the simultaneous localization and mapping (SLAM) problem requires that the probability distribution

$$P(x_k, m | Z_{0:k}, U_{0:k}, x_0) \tag{1.1}$$

be computed for all time steps k. This probability distribution describes the joint posterior density of the landmark locations and vehicle state (at time step k) given the recorded observations and control inputs up to and including time k together with the initial state of the vehicle. In general, a recursive solution to the SLAM problem is desirable. Starting with an estimate for the distribution $P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1})$ at time step $k-1$, the joint posterior, following a control u_k and observation z_k , is computed using Bayes theorem. This computation requires that a state transition model and an observation model are defined describing the effect of the control input and observation respectively.

The observation model describes the probability of making an observation z_k when the vehicle location and landmark locations are known and is generally described in the form

$$P(z_k|x_k, m) \tag{1.2}$$

It is reasonable to assume that once the robot location and map are defined, observations are conditionally independent given the map and the current robot state.

The motion model for the vehicle can be described in terms of a probability distribution on state transitions in the form

$$P(x_k|x_{k-1}, u_k) \tag{1.3}$$

That is, the state transition is assumed to be a Markov process in which the next state x_k depends only on the immediately preceding state x_{k-1} and the applied control u_k and is independent of both the observations and the map.

The SLAM algorithm is now implemented in a standard two-step recursive predict-update form:

Predict

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \int P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0) dx_{k-1} \tag{1.4}$$

Update

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \frac{Pz_k|x_k, mP(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})} \tag{1.5}$$

Equations 1.4 and 1.5 provide a recursive procedure for calculating the joint posterior $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$ for the robot state x_k and map m at a time step k based on all observations $Z_{0:k}$ and all control inputs $U_{0:k}$ up to and including time step k. The recursion is a function of a robot model $P(x_k|x_{k-1}, u_k)$ and an observation model $P(z_k|x_k, m)$.

It is worth noting that the map building problem may be formulated as computing the conditional density $P(m|X_{0:k}, Z_{0:k}, U_{0:k})$. This assumes that the location of the robot x_k is known (or at least deterministic) at all times, subject to knowledge of initial location. A map m is then constructed by fusing observations from different locations. Conversely, the localization problem may be formulated as computing the probability distribution $P(x_k|Z_{0:k}, U_{0:k}, m)$. This assumes that the landmark locations are known with certainty, and the objective is to compute an estimate of robot location with respect to the map.

With supplied relative and absolute sensor information, SLAM estimates most likely configuration of the robot trajectory and map. Various filters were applied to SLAM to address this state estimation problem [38, 39, 35, 37].

1.2.1 Gaussian-based Filter SLAM

Gaussian-based Filter SLAM provides a framework to estimate a Gaussian random variable with mean x and covariance matrix P via readings from sensors. The most basic Gussian-based Filter SLAM is Extended Kalman Filter SLAM (EKF-SLAM) [39, 38, 40, 41] which is comprised of three main processes: predict, observe and update. These processes are periodically executed over a series of time steps.

The predict process propagates the state estimation over time. In predict process, the predicted state $x_{k|k-1}$ and its covariance matrix $P_{k|k-1}$ at time step k are given by

$$
\begin{aligned}
x_{k|k-1} &= f(x_{k-1}, u_k) + w_k \\
P_{k|k-1} &= \nabla f P_{k-1|k-1} \nabla f^T + Q
\end{aligned}
\tag{1.6}
$$

where w_k is the process noise assumed to be white Gaussian with zero mean and a covariance Q, the function f depends on the motion model, ∇f is the Jacobian of f with respect to x_{k-1}, and $P_{k-1|k-1}$ denotes the state covariance matrix at time step $k-1$.

The observations $z_{k|k-1}$ that are obtained from the state $x_{k|k-1}$ and m at the time step k in observe process are given by

$$
z_{k|k-1} = h(x_{k|k-1}) + v_k \tag{1.7}
$$

where h defines the nonlinear coordinates transformation from the state to the observation $z_{k|k-1}$. The noise v_k is assumed to be white Gaussian with zero mean, and z_k is measured from the actual environment and its covariance matrix is denoted by R_k).

In update process, the state x_k, m and the associated covariance matrix P_k are updated by EKF using the observation z.

$$
\begin{aligned}
\begin{bmatrix} x_k \\ m \end{bmatrix} &= \begin{bmatrix} x_{k|k-1} \\ m \end{bmatrix} + W_k[z_k - h(x_{k|k-1}, m)] \\
P_k &= P_{k|k-1} - W_k S_k W_k^T
\end{aligned}
\tag{1.8}
$$

where

$$
\begin{aligned}
W_k &= P_{k|k-1} \nabla h^T S_k^{-1} \\
S_k &= \nabla h P_{k|k-1} \nabla h^T + R_k
\end{aligned}
\tag{1.9}
$$

and ∇h is the Jacobian of h evaluated at $X_{k|k-1}$ and m.

The performance of EKF estimation depends on how similar between linearized model and originals. The quality of EKF estimation is influenced by the degrees of non-linear in original models. For reducing the linearization error in the estimator, Unscented Kalman Filter (UKF) is proposed with higher computation cost.

Several deterministic sigma points are generated from a Gaussian distribution via unscented transform, which can represent the properties of the underlying distribution [42]. Compare with the liberalization of the original models, the sigma points can directly be propagated through the original models. A SLAM method based on UKF is called UKF SLAM which is proposed in [43, 44, 45].

The main computation cost in the EKF-SLAM is involved in the update process which calculates the robot pose and mapping result with the full covariance matrix. This means that the computation cost is changed according to the mapping area. For solving this problem, an alternative solution called Extended Information Filter SLAM (EIF-SLAM) [46, 47, 48] can reduce the computation cost in update process by information form rather than moment form to simplify the update process to an addition operation. However, the mean and covariance cannot be directly got unless processing inversion operation with high computation cost. Basically, keeping the state history in the information matrix is feasible since the information matrix is a relatively sparse matrix.

1.2.2 Particle Filter based SLAM

Basically, EKF-SLAM or EIF-SLAM belonging Gaussion-based Filter SLAM assume that the SLAM distribution should follow the Gaussian distribution to simulate uncertainty of the system. However, the Gaussion-based Filter SLAM cannot get a good performance without the noise following Gaussion distribution. In this case, Particle Filter based SLAM (PF-SLAM) can provide a better solution.

The general form of a particle filter for SLAM is as follows. We assume that, at time step $k-1$, the joint state is represented by $w^i_{k-1}, X^i_{0:k-1}, P(m|X^i_{0:k-1}, Z_{0:k-1})^N_i$

For each particle, compute a proposal distribution, conditioned on the specific particle history, and draw a sample from it

$$x^i_k \sim \pi(x_k|X^i_{0:k-1}, Z_{0:k}, u_k) \tag{1.10}$$

This new sample is joined to the particle history $X^i_{0:k} = X^i_{0:k-1}, x^i_k$.

Weight samples according to the importance function

$$w^i_k = w^i_{k-1} \frac{P(z_k|X^i_{0:k}, Z_{0:k-1})P(x^i_k|x^i_{k-1}, u_k)}{\pi(x^i_k|X^i_{0:k-1}, Z_{0:k}, u_k)} \tag{1.11}$$

The numerator terms of this equation are the observation model and the motion model, respectively. The former differs from 1.2 because particle filter requires dependency on the map be marginalized away.

$$P(z_k|X_{0:k}, Z_{0:k-1}) = \int P(z_k|x_k, m)P(m|X_{0:k-1}, Z_{0:k-1})dm \tag{1.12}$$

If necessary, perform resampling. (When best to instigate resampling is an open problem. Some implementations resample every time-step, others after a fixed number of time-steps, and others once the weight variance exceeds a threshold.) Resampling is accomplished by selecting particles, with replacement, from the set ${X^i_{0:ki}}^N$, including their associated maps, with probability of selection proportional to w^i_k. Selected particles are given uniform weight, $w^i_k = 1/N$.

For each particle, perform an EKF update on the observed landmarks as a simple mapping operation with known robot pose.

With PF-SLAM, a finite number of particles or samples is applied, which is similar to the localization with a known map using particle filter. Since SLAM needs to not only estimate robot pose but also mapping the environment, the computation cost is much higher than the common localization with the known map [24, 30, 32, 49, 8, 50]. For increasing computation efficiency, only the robot trajectory is sampled in Rao-Blackwellized particle filter [37, 51] since the map can be produced according to the estimated robot trajectory. FastSLAM [35] is also affected by this notion to construct each particle representing the hypothesis of the robot pose with EKF associated with its map. According to the motion model, the particles are spread to construct the desired distribution. After that, the particles are weighted and

resembled based on the measurement model. For increase the number of particles in the most likely spot, a new proposal distribution is proposed in FastSLAM 2.0 [36].

1.2.3 Filter-less SLAM

Instead of treating SLAM as a recursive filtering problem, Filter-less SLAM also called Graph-based SLAM casts it as an optimization problem [52]. Since the motion and measurement models usually are non-linear, non-linear least square methods such as Gauss Newton [53, 54, 55] or Levenberg-Marquardt [56, 57, 58] were applied. With the given rotations, square root SAM [59] was improved with matrix factorization techniques to constrict iSAM [60]. The Thin Junction Tree Filter [61, 62] and Treemap [63] algorithms perform this optimization in tree-like topologies.

Filter-less SLAM jointly optimize the entire robot poses during the whole SLAM process with relative and absolute measurements. For achieving the similar results with filtering results, forward-backwards smoothing techniques must be employed. The methods are well suited for rich sensor data application utilizing point clouds or camera image for obtaining the optimized results. However, the amount of sensor data may cause data association problems to lead the wrong results in the optimization. Outlier rejection methods have been proposed to increase the robustness of the filter-less SLAM [63].

1.2.4 Kidnapping problem in SLAM

Kidnapping is a localization problem that occurs when an unexpected movement happened to the robot due to the interaction with its surroundings. It makes localization fail, which means that the robot cannot estimate its posture in the environment correctly. Several methods were proposed to solve this problem in a previous known-map situation; for example, Monte Carlo Localization [30, 32, 64, 50] that performs global localization regularly. Although this method is not efficient from the point of view of the time cost associated with recovering from the kidnapping event, the successful correction of the wrong posture is reliable. Although the solutions of kidnapping in known environment were proposed,

these solutions cannot be applied to the kidnapping in previous unknown situation because of unique situations caused by kidnapping. This problem motivated us to propose a new solution to obtain an accurate result of SLAM while kidnapping happens.

1.3 Outline of this Thesis

To propose a reasonable solution for the kidnapping problem, an analysis is carried out to distinguish different situations of kidnapping. The solution structure including kidnapping detection and kidnapping recovery is determined. After that, a kidnapping detection framework is proposed with simulations verification. Two kidnapping recovery methods are introduced according to different kidnapping types. Experiments were conducted to verify the performance of proposed methods with a robot platform. Conclusions and future works are summarized according to the experimental results.

Chapter 2 introduces a new problem in SLAM which motivates us to construct a set of methods to solve it. The problem called kidnapping in SLAM is introduced with analyzed situations which the robot may encounter. Based on the analyzed situations, we firstly describe a basic structure of the solution including the kidnapping detection and kidnapping recovery, and then present the relationship between the proposed methods in the thesis.

Chapter 3 presents the proposed methods of kidnapping detection called Double Kidnapping Detection and Recognition (DKDR) and Probabilistic DKDR (P-DKDR). P-DKDR follows the framework of DKDR with adding the uncertainty of the estimated state. DKDR framework can judge whether kidnapping has occurred and identify the type of kidnapping with filter-based SLAM. The basic structure, metrics and thresholds of DKDR are described, and simulations on DKDR and P-DKDR with EKF-SLAM are executed. The performance of each method is evaluated by Receiver Operating Characteristic (ROC) [65].

Chapter 4 describes two methods of kidnapping recovery according to the kidnapping situations. One of the methods called Monte Carlo Localization with Map Uncertainty (MCl-MU) is introduced with the problem of basic Monte Carlo Localization method and its

implementation. Another method is based on idea of Known and Unknown Environment's Simultaneous Localization (KUESL) is introduced. Following the proposed idea of KUESL, the implementation with Gmapping and Modified MCL-MU (M-MCL-MU) are utilized.

Chapter 5 introduces the experiments to verify methods of kidnapping detection and kidnapping recovery in an indoor environment. Firstly, the setup of the experiment about the mobile robot platform is introduced. The experiments preforming DKDR and P-DKDR are similar to their simulations. The results of the experiments are described and discussed according to the real situation. The experiments about MCl-MU and KUESL are also introduced with results evaluated by successful rate. The discussion about experiments results shows the reasons of the failure cases.

Chapter 6 concludes this thesis and discusses the possible works in the future.

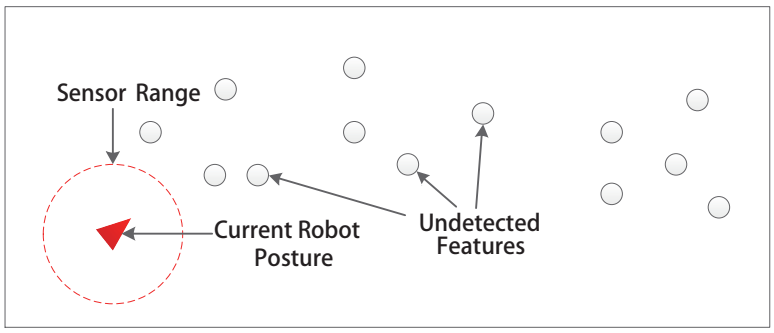
Chapter 2

Problem Statement

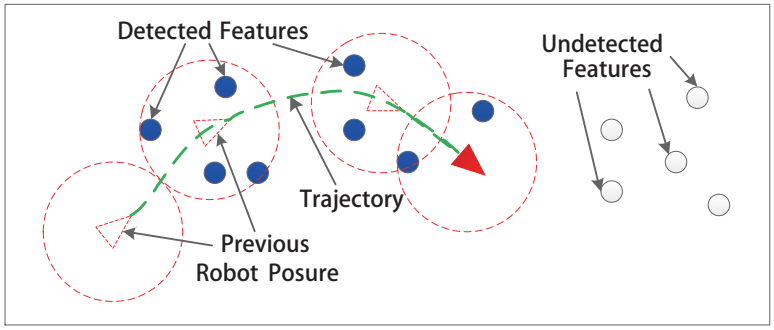
In this chapter, the problem about kidnapping in previous unknown environment is analyzed to divide into different situations. Furthermore, a new solution structure of the problem is introduced based on different kidnapping types.

2.1 Analysis of Kidnapping in SLAM

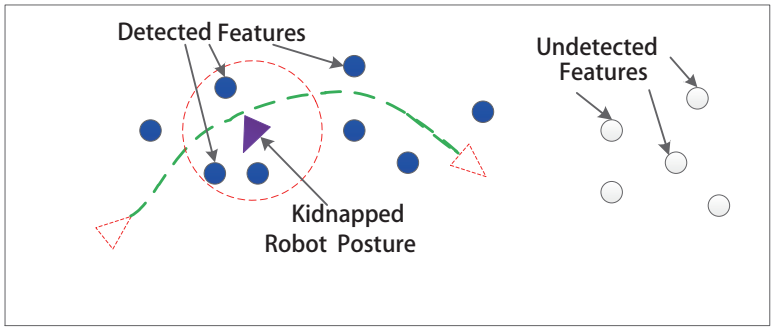
In an unknown environment, the robot needs to explore unknown area with SLAM, as shown in Figure 2.1(a)(b). Therein, white circles represent undetected features in the environment. A mobile robot, carrying an exteroceptive sensor with the limited range represented as a red dashed circle, is shown as a red triangle. During the SLAM, the robot collects useful information (features' positions) about its surroundings while it moves around in the environment. The robot's trajectory is shown in green dashed line. When the robot detects a new feature, the position of the feature is recorded to the map. Blue circles show that the previous undetected features have been detected and included in the map. During the robot performing SLAM, the constructed mapping areas can be classified as explored areas (detected features) or unexplored areas (undetected features). If the robot was kidnapped into an explored area as shown in Figure 2.1(c), the existing kidnapping recovery methods can be applied with reasonable modification to correct the robot posture



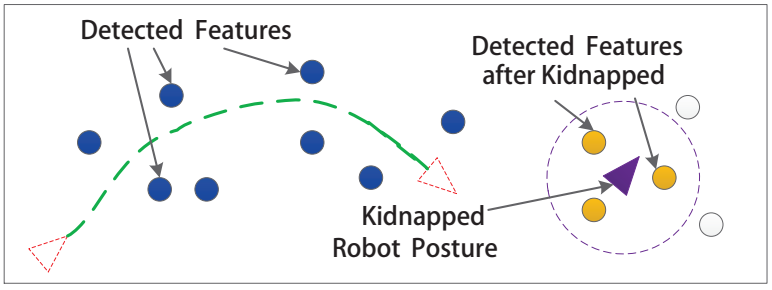
(a)



(b)



(c)



(d)

Figure 2.1. Different situations of kidnapping in SLAM. (a) Initial situation of SLAM. (b) Result of SLAM before kidnapping. (c) The robot is kidnapped to the explored area. (d) The robot is kidnapped to the unexplored area.

with known features positions. However, if the robot was kidnapped into an unexplored area as shown in Figure 2.1(d), directly utilizing the existing methods may result in a wrong estimated posture. Although there is no known-map actually around it, the robot still tries to estimate its posture with the collected information. These two different possible situations make the kidnapping in an unknown environment has its unique problems which need to provide a new solution.

About the influence of the kidnapping in the unknown map situations, it does not only cause an incorrect estimation of the robot posture but also a deformation of the mapping result because of the property of SLAM. With filter-less SLAM algorithms [66, 67], the incorrect collected information is directly added to the explored map after kidnapping, which affects the consistency of the mapping result. In contrast, the information belonging to the explored map is not affected, and the explored map can potentially be utilized in the recovery scheme after kidnapping. However, this property does not exist in filter-based SLAM. In filter-based SLAM, the explored area may also be deformed by the incorrect information, which makes the explored map information difficult to be recovered after kidnapping.

To prevent this scenario, a check about kidnapping is required. With timely detection, the correct information about the explored area will not be deformed by kidnapping. Some of this information can then be utilized to recovery from the kidnapping with sufficient conditions. After that, a suitable kidnapping recovery method should be carried out according to the kidnapping situation to relocalize the robot posture in the environment.

2.2 Solution of kidnapping in SLAM

As we described above, the new situations caused by kidnapping in the unknown environment makes the accurate robot posture cannot be recovered directly. A new structure of the solution need to be proposed to relocalize the robot posture effectively and efficiently. Basically, the solution is divided into two sequential parts, which are kidnapping detection and kidnapping recovery.

The requirement of kidnapping detection is to detect the kidnapping on time to prevent the robot causing hazardous actions. Since the robot performs at least one localization task while processing the kidnapping detection, the computation load for the kidnapping detection should be small to allow the tasks to be executed smoothly. Furthermore, the kidnapping detection should provide sufficient information about kidnapping to make sure a suitable kidnapping recovery method is selected.

Classifying kidnapping into several categories should be helpful to provide basic information to decide what recovery strategy should be employed after kidnapping. Extending the classification given in [68], two main types of kidnapping (types A and B) are proposed in this thesis. Type A kidnapping occurs when the robot does not correctly estimate its posture after its actual posture changed beyond some range. For example, while operating a task autonomously, a robot is moved to another place by a human, and this change in posture is not imported into the SLAM algorithm. In this case, two possible situations (kidnapped in explored areas or kidnapped in unexplored areas) exist. The ordinary recovery methods may cause fault if it is applied directly. Similar situations such as entering elevator, being pushed away by other robots, or falling down from stairs should also be recognized as type A kidnapping. Type B kidnapping occurs when the robot does not actually change its posture, the predicted posture however is significantly changed. For example, the robot is stuck in an area or moving in a slippery area. In this case, since the robot is just kidnapped in explored area, the ordinary recovery methods could be applied directly. Both type A and B kidnapping events are divided into two sub-types based on the range of kidnapping (e.g., slipping belongs to type B.1 kidnapping, while being stuck belongs to type B.2 kidnapping). Providing this detailed information can help to design a suitable recovery algorithm, such as the range of scan-map matching [67, 69, 66].

Existing methods for kidnapping detection can be divided into two types, physical and mathematical methods. Physical methods that use specific sensors (e.g., barometer [70], accelerometer [71] and switch [72]) to measure whether or not kidnapping has occurred. However, these methods have some limitations. First, these methods require additional

sensors to be mounted on the robot. Therefore, the reliability of these methods cannot be ensured when the sensors' states are abnormal. Second, each sensor can detect only a specific type of kidnapping, such as the robot is lifted up or stuck in an area. On the other hand, mathematical methods utilize only inherent sensors such as wheel encoder and laser range finder (LRF) to observe abnormal situations. Compared with physical methods, mathematical methods can be used in robots that have proprioceptive and exteroceptive sensors to locate themselves. Using the entropy of location probabilities [73], the robot can detect kidnapping with the given information. However, it cannot be applied in SLAM when the information of the map is unknown. Metric-based detection [68] needs two independently operating sensor resources, which are required along with pervious test data.

After processing kidnapping detection, different kidnapping recovery methods should relocalize the robot posture according to the current situation. If the robot is kidnapped to the explored area, the kidnapping recovery method should relocalize the robot posture in the map built by SLAM. However, existing global localization methods [64, 30, 32, 50] require the provided map accurately enough, which the map constructed by SLAM with uncertainties is not fulfilled. For obtaining an accurate robot posture estimation, the map uncertainty also should be taken into account in the localization method. In another aspect, the kidnapping recovery method should start a new SLAM to estimate the robot posture with a new map if the robot is kidnapped to the unexplored area since the autonomous mobile robot cannot navigate itself without a map. The robot needs to acquire surrounding information as soon as possible to construct a map for continuing its tasks.

According to the idea of the solution of kidnapping in the previously unknown environment, that methods about kidnapping detection and kidnapping recovery are proposed in this thesis. Their relationship are shown in Figure 2.2. Two kidnapping detection methods called Double Kidnapping Detection and Recognition (DKDR) and Probabilistic DKDR (P-DKDR) are proposed with distinct requirements. DKDR needs less computation expense than P-DKDR, and P-DKDR has less false alarms than DKDR in a large scale environment. The output of the two kidnapping detection methods are same, which is the type of kidnap-

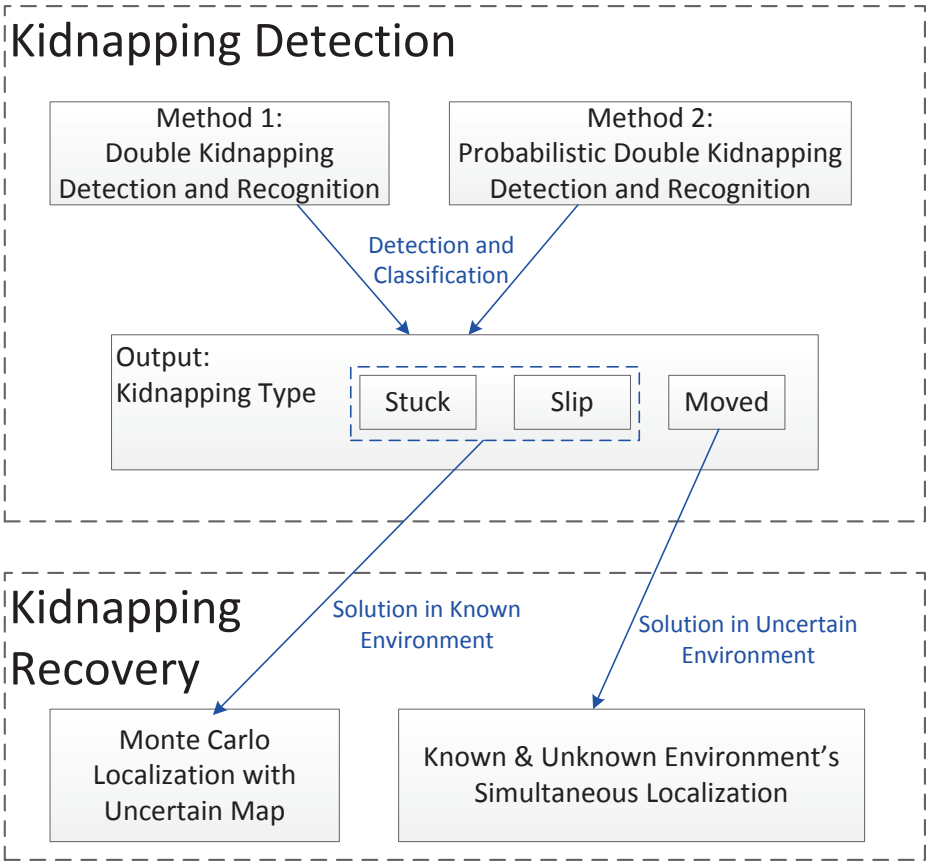


Figure 2.2. Relationship of proposed methods.

ping. It the robot is kidnapped around the place before kidnapping, a method called Monte Carlo Localization with Map Uncertainty (MCL-MU), which realizes no motion kidnapping recover based on Monte Carlo Localization, is applied. In the other situation where the robot is kidnapped out of the range of sensors, a method named Known and Unknown Environment's Simultaneous Localization (KUESL) is preformed to obtain the correct robot posture, no matter the robot is in explored area or not. With these methods, a new complete kidnapping solution is provided in this thesis.

2.3 Summary

This chapter introduced an analysis of kidnapping problem in an unknown environment while processing SLAM. Two different situations about kidnapping are described based on the explored area and the new area. Furthermore, the situations of kidnapping in SLAM without solutions with different types of SLAM are presented.

Based on the kidnapping situations in SLAM, a solution structure including kidnapping detection and kidnapping recovery is proposed. Furthermore, the type of kidnapping is defined to obtain detailed information about the kidnapping. In kidnapping detection, a framework called DKDR, which can detect and distinguished type of kidnapping, is proposed. A method called P-DKDR combining the uncertainty of the estimation based DKDR is also introduced in the kidnapping detection. MCL-MU and KUESL are introduced as kidnapping recover to solve different types of kidnapping. The relationship of the proposed methods is summarized in a relation map.

Chapter 3

Kidnapping Detection

In this chapter, we describe a property of filter-based SLAM that corrects the entire map of the environment with current observations after the update process. Based on this property, a framework using metrics is implemented in ordinary SLAM processes in real time to detect and recognize kidnapping, which is called double kidnapping detection and recognition (DKDR). Furthermore, an improved method called Probabilistic DKDR (P-DKDR) with improved metrics is also introduced. Third, a new classification for kidnapping is proposed to provide more information for the kidnapping recovery strategy. Fourth, a method to determine thresholds for the metrics without previous testing data is applied in the method. To demonstrate the universality and simplicity of the framework, we report on the application of the DKDR framework and P-DKDR method in EKF-SLAM with simulations.

3.1 DKDR Framework

As shown in Figure 3.1, two new processes are embedded into the ordinary SLAM structure to construct the DKDR framework. After the predict and observe processes, one of the new processes called prior-check process is performed. The prior-check process includes the main work for detecting kidnapping and identifying the kidnapping type. Subsequently,

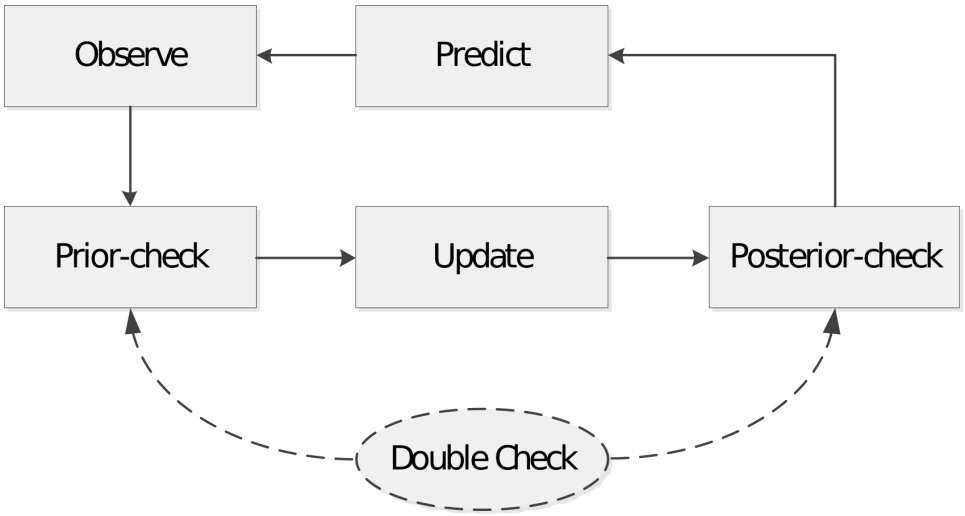


Figure 3.1. Overall DKDR workflow

the update process updates all of the information including the robot's state and the map information. Another new process called the posterior-check process follows the update process. In this process, the change in the entire map is checked to detect kidnapping.

As we introduced EKF-SLAM in Section 1.2.1, comparing observations in sequential time steps can be detect and distinguish kidnapping in the prior-check process. If kidnapping occurred, the difference between observations $z_{k|k-1}$ and z_k would be enlarged. Thus, the value of the component in equation 1.8, $z_k - h(x_{k|k-1}, m)$, would be significantly increased. W_k includes the state covariance matrix $P_{k|k-1}$ multiplying $z_k - h(x_{k|k-1}, m)$, while $P_{k|k-1}$ includes variance of each feature's position. The features' positions with high variance are affected obviously by kidnapping. At the end, the map including features' positions m would be changed obviously after the update process. By comparing the difference between m_k and m_{k-1}, the robot can check whether the information has changed. This work is accomplished in the posterior-check process. If kidnapping occurs and has not been detected by the prior-check process, the posterior-check process could give an alarm. The prior-check process is a prevention mechanism to stop the fault information from corrupting the information as a whole.

Two new processes, the prior-check and the posterior-check processes, are introduced in

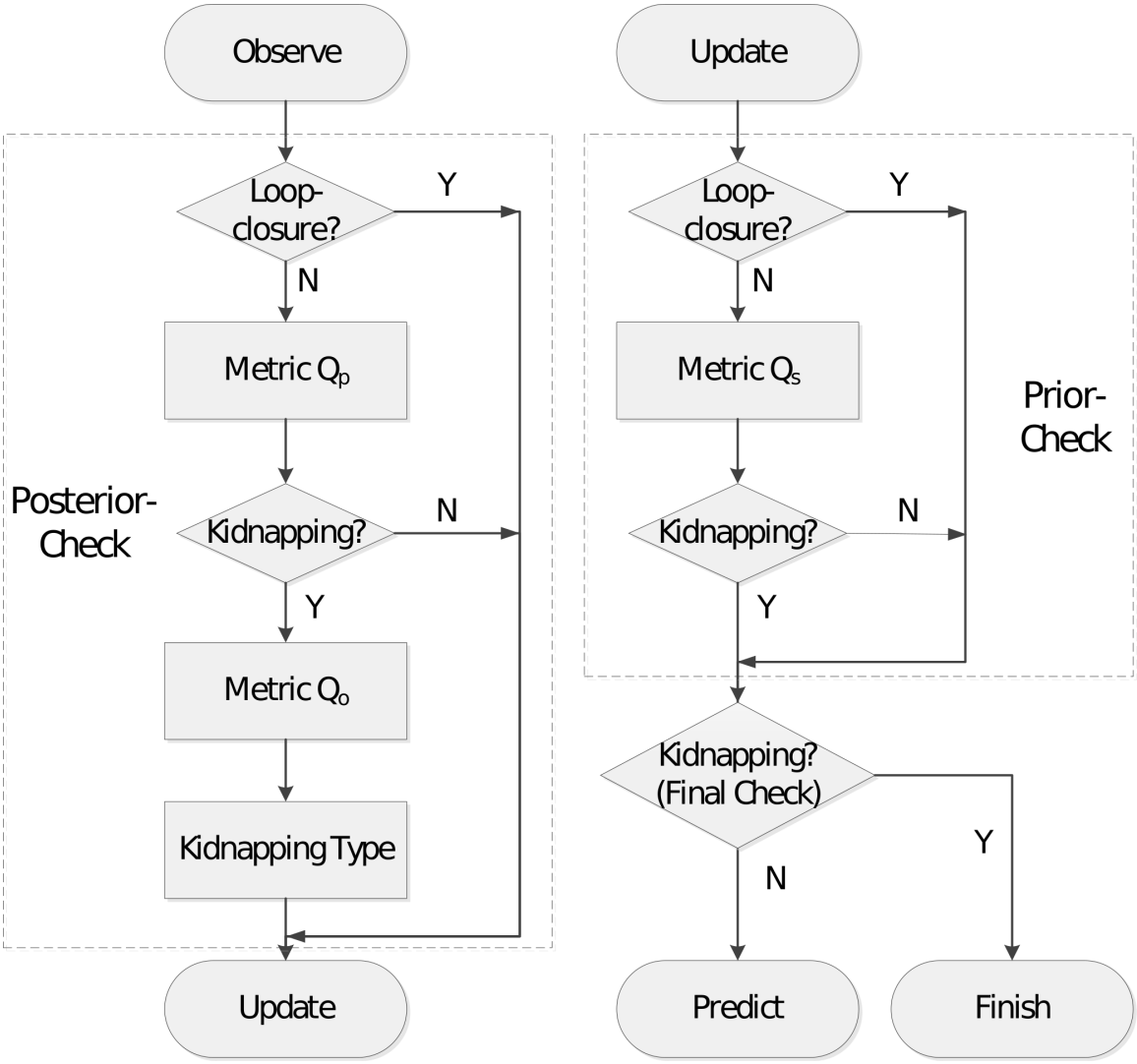


Figure 3.2. Workflows of prior-check and posterior-check processes

this section. The posterior-check process includes two functions : detection and recognition. A complete check-up can be done during the prior-check process. The posterior-check process using one metric is applied to check information changes after the update process. The workflow of each process in every time step is shown in Figure 3.2.

To evaluate whether the robot has moved to the designated position, the metric Q_p and its thresholds are needed. A predicted robot state $x_{k|k-1}$ is generated as an output in the predict process. The predicted observation $z_{k|k-1}$ can thus be calculated using equation 1.7. When the robot actually turns to the predicted state, the actual observation z_k should be similar to the predicted observation $z_{k|k-1}$. If the difference between these two observations exceeds a reasonable threshold, it indicates that kidnapping has happened. Although the kidnapping can be detected by Q_p, the type of kidnapping cannot be distinguished; another metric Q_o is needed to ascertain the type of kidnapping.

There are two main types of kidnapping and two subclasses for each type, as denoted in Section 2.2. The characterization of these kidnapping types is listed in Table 3.1. In sequential time steps, the positions of several overlapped features in the local coordinates can be determined. $\overline{AP}$ denotes the distance between actual and predicted positions of overlapped features in the local coordinates, whereas $\overline{CL}$ represents the distance between the current and last positions of overlapped features in the local coordinates. For dividing four different types of kidnapping, four thresholds are denoted as T_1, T_2, T_3 and T_4. Type A kidnapping occurs when $\overline{AP}$ is larger than T_1 and $\overline{CL}$ is larger than T_3, e.g., when the robot is carried to a new place or pushed into an unexpected area. Compared to type A.1 kidnapping, type A.2 kidnapping is much larger than T_1, indicating that the situation is more critical. In type A kidnapping, the situation of the robot after kidnapping cannot be ascertained. In this case, the algorithm for kidnapping recovery should judge the situation of the robot.

The autonomous robot may mistakenly estimate that it has moved to the predicted position, when it is actually still in the same place. This is an example of type B kidnapping, for which $\overline{CL}$ is smaller than T_3, and it indicates that the robot did not reach the target

Table 3.1. Different Types of Kidnapping

Type		Condition 1	Condition 2
Non-kidnapping		$\overline{AP} \leq T_1$	$\overline{CL} > T_3$
Type A	Type A.1	$T_2 \geq \overline{AP} > T_1$	$\overline{CL} > T_3$
	Type A.2	$\overline{AP} > T_2$	$\overline{CL} > T_3$
Type B	Type B.1	$T_2 \geq \overline{AP} > T_1$	$T_4 < \overline{CL} \leq T_3$
	Type B.2	$\overline{AP} > T_2$	$\overline{CL} \leq T_4$

due to slippage or another external force. Type B.2 kidnapping is recognized as the stuck problem, while other problems are classified as type B.1 kidnapping. For type B kidnapping, the existing methods of kidnapping recovery can be carried out because the robot still remains the explored area.

With Q_p and Q_o, it is easy to implement the complete check-up. However, these two metrics can only serve as pre-tests. If kidnapping cannot be detected with these two metrics, another metric Q_s is required for the post-test in the posterior-check process, which determines whether or not the information as a whole is normal. Moreover, the ability to distinguish normal information from abnormal information is also required.

The final check result (CR) of DKDR is based on the results of each check process, as shown in equation 3.1.

$$
\begin{aligned}
CR &= PrR \vee PoR \\
PrR &= \begin{cases} 1 & \text{if report kidnapping} \\ 0 & \text{otherwise} \end{cases} \\
PoR &= \begin{cases} 1 & \text{if report kidnapping} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{3.1}
$$

where PrR and PoR denote the CRs of the prior-check process and posterior-check process, respectively. CR is calculated by the OR operation with PrR and RoR. If $CR = 1$, DKDR reports that a kidnapping event has occurred and stops SLAM process. Recovery methods are executed after getting the type of kidnapping from detection, such as Monte Carlo Localization or starting new SLAM process. Since this study is focus on the detec-

tion and classification of kidnapping, recovery methods are not discussed in this chapter. For calculating CR, *OR* operation can be replaced by other operations such as the *AND* operation to adapt to different requirements, such as the situation requiring less false alarm and insensitive in detecting kidnapping.

Loop-closure is the task of deciding whether or not a robot has returned to a previously visited area after an excursion of arbitrary length [74, 75]. Both prior-check and posterior-check processes could mistake loop-closure for kidnapping if there is no previous prevention mechanism. To prevent this type of failure, loop-closure is checked before the kidnapping judgment in the prior-check and posterior-check processes.

3.2 Metrics and Thresholds

Three metrics, Q_p, Q_o and Q_s, were briefly introduced in the previous section. In this section, we describe these metrics along with a method to determine their appropriate threshold values.

3.2.1 Metrics in DKDR

Q_p and Q_o compare the difference in observations directly. Without coordinate transformation, these metrics need less computation. In DKDR, the metrics are caculated based on Euclidean distance, which is more efficient than other kind of distance, such as Mahalanobis distance.

With the root mean square, Q_p at time step k is given by

$$Q_p(k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|z_k^i - z_{k|k-1}^i\|^2} \tag{3.2}$$

where N represents the number of overlapped observed features between sequential time steps k and $k+1$. Z_i denotes the observation of the i_{th} overlapped feature. Q_o at time step

k is given by

$$Q_o(k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|z_k^i - z_{k-1}^i\|^2} \tag{3.3}$$

Q_s at the time step k is given by

$$Q_s(k) = \sqrt{\frac{1}{M}\sum_{i=1}^{M} \|m_k - m_{k-1}\|^2} \tag{3.4}$$

where M denotes the number of overlapped features between sequential time steps $k-1$ and k.

3.2.2 Metrics in P-DKDR

For decreasing the effect of the increasing uncertainty in SLAM process in large scale environment, Mahalanobis distance is applied in metrics of P-DKDR.

With the root mean square, Q_p is given by

$$Q_p(k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} V_i(k)^T S_i(k)^{-1} V_i(k)} \tag{3.5}$$

where

$$\begin{aligned} V_i(k) &= z_k^i - z_{k|k-1}^i \\ S_i(k) &= R_k^i + R_{k|k-1}^i \end{aligned} \tag{3.6}$$

and N represents the number of the overlapped observed features between sequential time step k and $k+1$. Z_i denotes the observation of i_{th} overlapped feature. Q_o is given by

$$Q_o(k) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} W_i(k)^T M_i(k)^{-1} W_i(k)} \tag{3.7}$$

where

$$\begin{aligned} W_i(k) &= z_k^i - z_{k-1}^i \\ M_i(k) &= R_k^i + R_{k-1}^i \end{aligned} \tag{3.8}$$

Q_s is given by

$$Q_s(k) = \sqrt{\frac{1}{M}\sum_{i=1}^{M} D_{mi}(k)^T O_{mi}(k)^{-1} D_{mi}(k)} \tag{3.9}$$

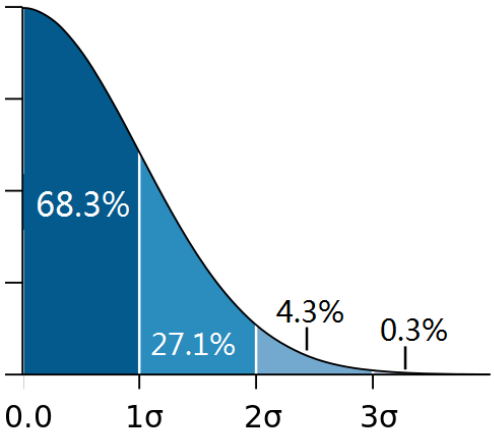


Figure 3.3. Profile of Half-Normal Distribution.

where

$$\begin{aligned} D_{mi}(k) &= m_k - m_{k-1} \\ O_{mi}(k) &= P_k^i + P_{k-1}^i \end{aligned} \tag{3.10}$$

and M denotes the number of the overlapped features between sequential time step $k-1$ and k.

Comparing with DKDR, P-DKDR adds the associated covariance matrix into metrics. It can decrease the rate of false alarm in the whole detection reports. It causes that the whole performance of P-DKDR is better than DKDR in a large scale environment. About the thresholds of metrics, the method proposed in DKDR is used in P-DKDR. Detection and classification condition are the same as that in DKDR. The difference is the composition of metrics.

3.2.3 Thresholds

The accuracy of detection and classification is related to the metrics' thresholds, and a suitable method that works in real time is required to determine reasonable thresholds. A method using the ROC curve to determine the thresholds for a system of detection was proposed; however, this method requires the experimental data in advance, which is not

convenient. A method to determine the thresholds along with the corresponding kidnapping types without previous data is described in this section. This method can be applied in various localization systems.

If there is no noise in the system, Q_p and Q_s should be equal to zero in normal situations, indicating that the actual position of the robot should be the same as the predicted position. However, some errors in the system cannot be avoided, such as the noise of the observation and the uncertainty of the robot model. Hence, the values of the metrics deviate from zero and obey the half-normal distribution (Figure 3.3). Because the standard deviation can reflect the percentage of events, as shown in Figure 3.3, the thresholds can be determined by the percentage of kidnapping events. With different values of noise in the system, this method can provide different suitable thresholds. The different types of kidnapping along with the metrics and their thresholds used in our simulations and experiments are shown in Table 3.2. With these values, the robot can judge its situation and provide a suitable navigation strategy, allowing the robot to avoid explored areas where slippage occurs easily.

The mean μ of these metrics is equal to zero and the covariance σ can be calculated from the data collected before time step k. T_{p1} and T_{p2} of Q_p can be determined from the standard deviation σ_p of Q_p shown in equation (3.11):

$$\sigma_p(k) = \sqrt{\frac{1}{k}\sum_{i=1}^{k} Q_p(i)^2} \tag{3.11}$$

The threshold of Q_o is equal to T_{p1}. The threshold of the metric Q_s can be determined by its standard deviation σ_s shown in equation (3.12):

$$\sigma_s(k) = \sqrt{\frac{1}{k}\sum_{i=1}^{k} Q_s(i)^2} \tag{3.12}$$

Please note that Table 3.1 only describes the conception of kidnapping type which is different with Table 3.2. In

Table 3.2. Thresholds of Each Kidnapping Type

M	**Type A.1**	**Type A.2**	**Type B.1**	**Type B.2**
Q_p	$(T_{p1}, T_{p2}]$	$(T_{p2}, +\infty)$	$(T_{p1}, T_{p2}]$	$(T_{p2}, +\infty)$
Q_o	$(T_{p2}, +\infty)$	$(T_{p2}, +\infty)$	$(T_{p1}, T_{p2}]$	$[0, T_{p1}]$
Q_s	$(T_s, +\infty)$	$(T_s, +\infty)$	$(T_s, +\infty)$	$(T_s, +\infty)$

$T_{p1} = 3\sigma_p$, $T_{p2} = 4\sigma_p$, $T_s = 3\sigma_s$.

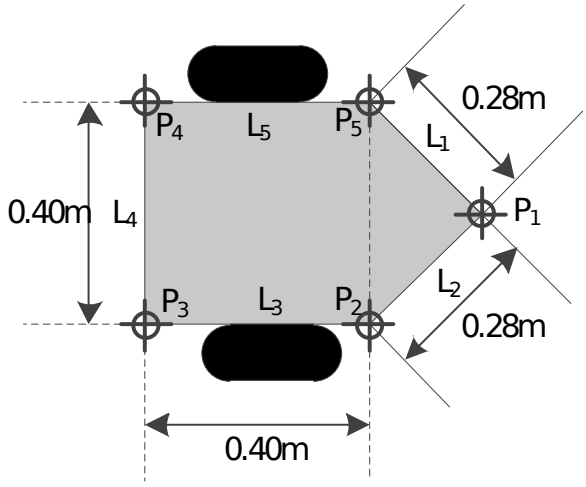


Figure 3.4. Dimensions of robot used in simulations.

3.3 Simulations

Simulations were conducted to investigate the feasibility and accuracy of the proposed method. To obtain the correct response in an ideal situation, the simulations were conducted under following conditions:

1. All sensor uncertainties follow Gaussian distributions.
2. The sensors mounted on the robot work well all of the time without temporary non-operating states.
3. The features are all in static states.
4. Data association is known, and the data association process does not affect the results.

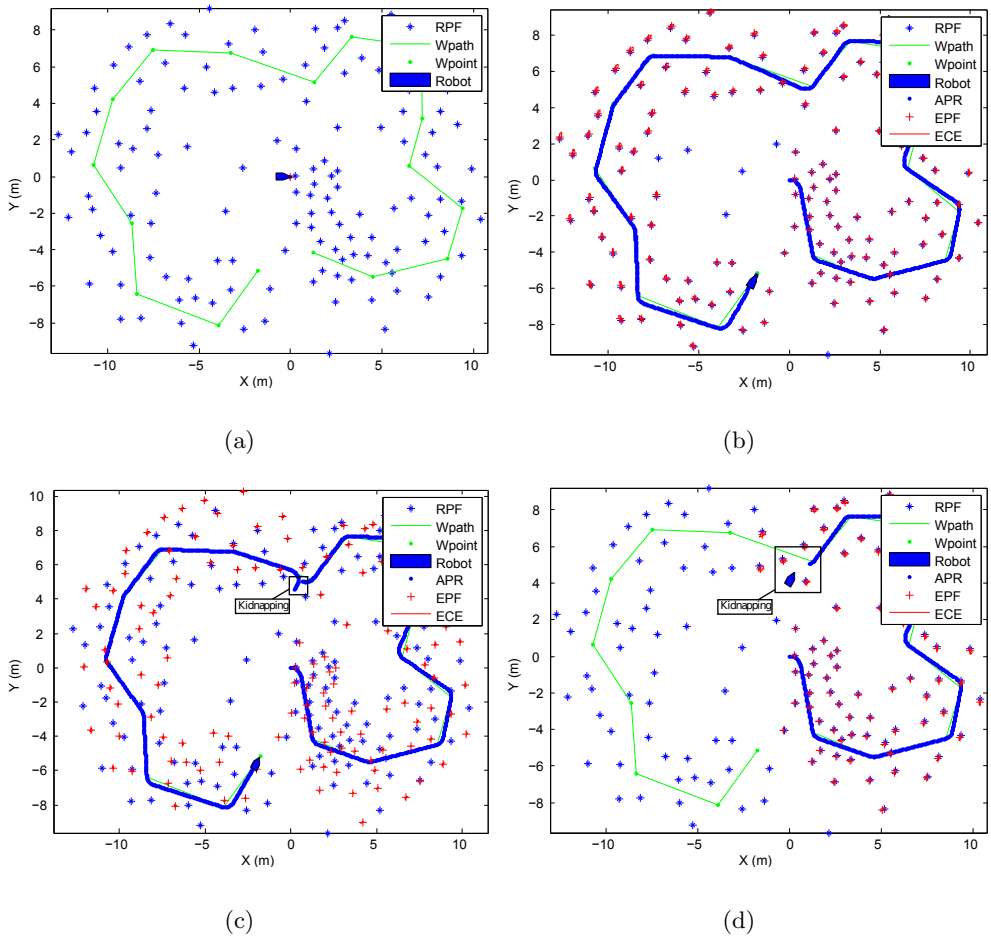
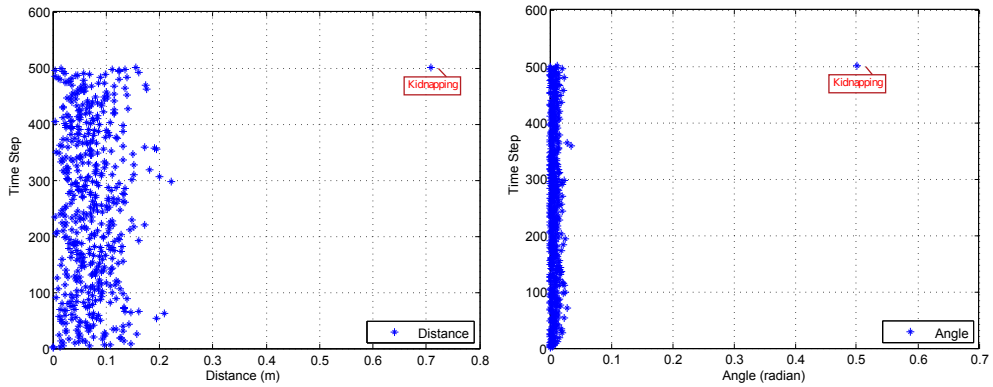


Figure 3.5. Results of the simulation in different situation. (a) Simulation Map. (b) Normal SLAM process. (c) Kidnapping result without detection. (d) Kidnapping result with detection.



(a) (b)

(c) (d)

Figure 3.6. Metrics used to detect kidnapping. (a) Distance. (b) Angle. (c) Q_p. (d) Q_s.

Table 3.3. Simulation Conditions

Condition	**Value**
Robot Speed	$0.3m/s$
Control Cycle	$0.2s$
Observation Cycle	$0.2s$
Max Range of Observation	$3m$
SD of Speed Noise	$0.09m/s$
SD of Orientation Noise	$9°$
SD of Observation Position Noise	$0.01m$
SD of Observation Angle Noise	$1°$

* SD: Standard Deviation

We aim to show the correctness of DKDR in the simulations, including the response of DKDR and the phenomenon after kidnapping without detection. Simulations were executed using MATLAB on a personal computer (CPU: 3.40GHz Intel Core i5, Memory: 8 GB DDR3). The source code was based on the EKF-SLAM algorithm in the SLAM package of Tim Bailey [33]. We modified and implemented our method into this package. The simulation conditions are shown in Table 3.3. The shape of the robot is shown in Figure 3.4.

The map and different situations are depicted in Figure 3.5. As indicated in Figure 3.5(a), 'RPF' denotes the real position of a feature. 'Wpoint' denotes the waypoint and 'Wpath' denotes the path connected with waypoints. The robot needs to drive itself towards each waypoint by the shortest path. Figure 3.5(b) shows the ordinary SLAM progress with the map. 'APR' denotes the actual positions of the robot. 'EPF' and 'ECE' represent the estimated position of the feature and its covariance ellipses. In normal situation, the EPF is near the RPF and distance between them becomes larger as time step passed. In this map, the robot performs the loop-closure before the end because it meets features that have been found before. Since DKDR recognizes the loop-closure event as an exception, even though it is similar to kidnapping. It prevents the erroneous triggering of kidnapping detection. Figure 3.5(c) shows the result for when kidnapping happened without detection. In this case, the distance between EPF and RPF is large after the kidnapping event, and the EPF, which should remain in the same position as before kidnapping, is changed. Figure 3.5(d)

shows the results for the case when kidnapping occurs with DKDR. The kidnapping was successfully detected by DKDR and the original information was correctly retained. The robot can reuse this original information for kidnapping recovery or map joining. These simulated results indicate that the DKDR can successfully detect kidnapping.

The representative data are shown in Figure 3.6. To judge whether kidnapping had actually occurred, we calculated real data without uncertainty at each time step. The distance between the predicted and actual position of the robot is shown in Figure 3.6(a). The difference between the angles of the predicted and actual orientations of the robot is shown in Figure 3.6(b). From Figure 3.6, we could easily determine how the differences in the robot's position and orientation changed due to kidnapping. The values of Q_p for kidnapping and non-kidnapping situations are shown in Figure 3.6(c). In the non-kidnapping situation, Q_p is lower than the first threshold T_{p1}. When Type A.2 kidnapping occurs in $501th$ time step, Q_p is larger than the second threshold T_{p2}. Moreover, the value of Q_s also exceeds T_s, as shown in Figure 3.6(d). Q_s is lower than T_s before kidnapping. Q_o is not shown here since it only be calculated once after Q_p beyond its threshold, which has been denoted in Section 3.1. Since Q_o is only calculated after kidnapping is detected shown in Figure 3.2, it is not shown in Figure 3.6.

We performed several simulations for kidnapping and non-kidnapping situations. If the robot was moved farther than 0.7 m, the situation is recognized as type A.2 kidnapping. Type B.2 is defined as when the robot stops at some specific point until its odometry data exceeded 0.7 m. The ranges of Types A.1 and B.1 are set as 0.2 m. The results processed by ROC are shown in Table 3.4. About the report of DKDR, the true positive rate is the fraction of the detected kidnapping out of the total number of actual kidnapping events, and the false positive rate is the fraction of the incorrectly detected non-kidnapping time steps out of the total number of actual non-kidnapping time steps. Compared to posterior-check process, prior-check process has higher true positive rate and false positive rate. Since OR operation is applied to the system, true positive rate and false positive rate of DKDR are higher than each check process.

Table 3.4. Operating Characteristics of DKDR

True Positive Rate	False Positive Rate
0.9990	0.0228

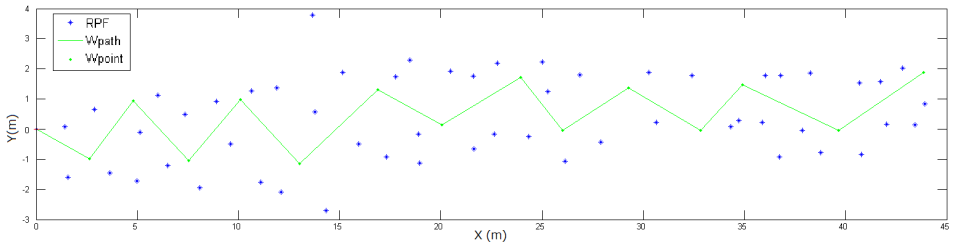
Table 3.5. Operating Characteristics of classification

Kidnapping Type	TPR	FPR
Type A.1	0.8403	0.0437
Type A.2	0.9763	0.0742
Type B.1	0.7823	0.0486
Type B.2	0.9821	0.0634

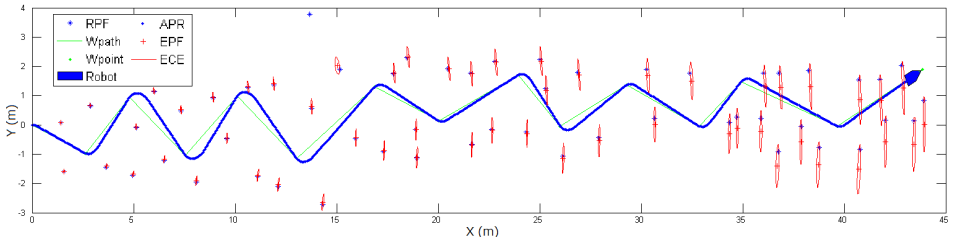
The simulated results for kidnapping type classification are shown in Table 3.5. The true positive rate is the fraction of detected kidnapping events of a certain type out of the total number of actual kidnapping events of that type, and the false positive rate is the fraction of wrongly detected kidnapping events of a certain type out of the total number of actual other types of kidnapping.

For investigating the feasibility and accuracy of P-DKDR, simulations with larger environment were conducted with P-DKDR and DKDR. The map of simulation has been shown in Figure 3.7(a). We did simulations in a no actual kidnapping situation during the whole SLAM process. The results of metrics in P-DKDR of Q_p and Q_s are shown in Figure 3.9. In non-kidnapping situation, the value of metric Q_p is less than the first threshold T_{p1}, and metric Q_s is lower than the threshold T_s. Comparing with the simulation results of DKDR shown in Figure 3.8, the value of both metrics is below the thresholds after time steps increased. This means that, although the uncertainty of the state increased by the SLAM process, influence in kidnapping detection has been decreased.

Several simulations were conducted to show the performance of P-DKDR and DKDR during SLAM process. Different simulations were conducted to verify the performance under the kidnapping situation and the non-kidnapping situation. The kidnapping is a man-made movement to the robot to simulate the real kidnapping that the robot is moved by the human in the environment. Then, we can judge whether the report of P-DKDR is a true alarm or a false alarm or no alarm. In each simulation, one time kidnapping event



(a)



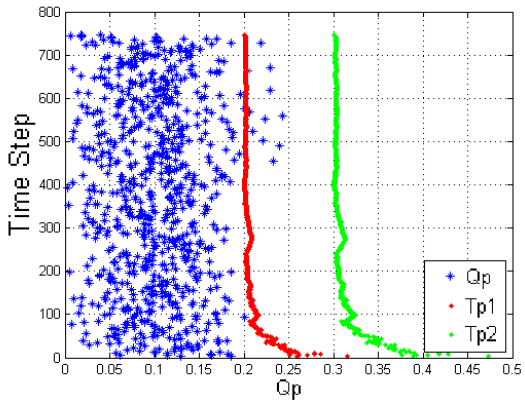
(b)

Figure 3.7. Map and EKF-SLAM. (a) Map for simulation. (b) Result of EKF-SLAM without kidnapping.

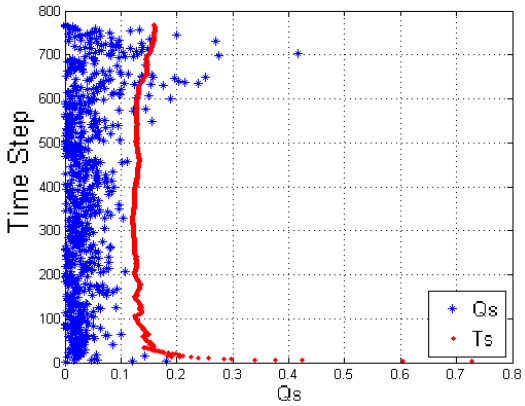
Table 3.6. Operating Characteristics of DKDR and P-DKDR in a large scale environment

Kidnapping Detection	**True Positive Rate**	**False Positive Rate**
DKDR	0.9900	0.1203
P-DKDR	0.9800	0.0431

is set randomly from step 500 to time step 800. The results processed by ROC are shown in Table 3.6. About report of kidnapping in DKDR and P-DKDR, the true positive rate is the fraction of detected kidnapping out of the total number of actual kidnapping events, and the false positive rate is the fraction of non-kidnapping time steps that is incorrectly detected out of the total number of actual non-kidnapping time steps. Comparing with DKDR, the false positive rate is decreased in P-DKDR, which means that the possibility of the false alarm is decreased. At the same time, the true positive rate of P-DKDR is also decreased a little comparing with DKDR. That means although the rate of false alarm is decreased, P-DKDR is not as sensitive as DKDR dealing with the real kidnapping.

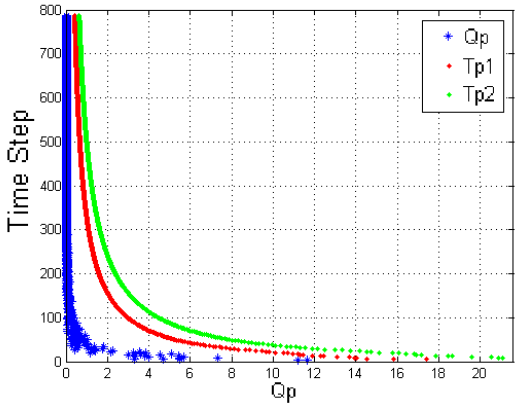


(a)

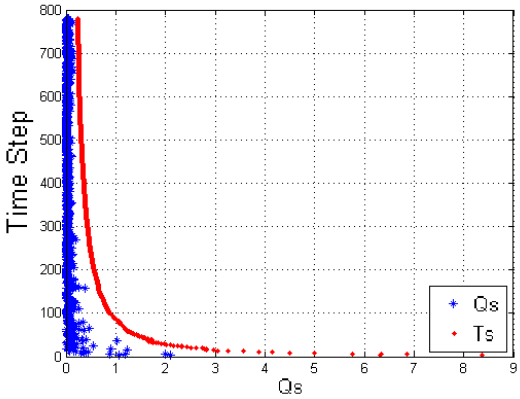


(b)

Figure 3.8. Non-kidnapping with DKDR. (a) Response of the metric Q_p. (b) Response of the metric Q_s.



(a)



(b)

Figure 3.9. Non-kidnapping with P-DKDR. (a) Response of the metric Q_p. (b) Response of the metric Q_s.

3.4 Summary

In this chapter, we have introduced DKDR framework which can detect and recognize kidnapping in real time. Furthermore, a method combing tribalistic estimation in DKDR called P-DKDR for large scale environments has been also described. The method to classify the kidnapping type for providing more detail information to kidnapping recovery has been presented. Distinct metrics and their thresholds have been introduced to realize detection and recognition of kidnapping. Simulation results show that DKDR framework and P-DKDR method can be applied into EKF-SLAM to detect and recognize kidnapping. Experiments for kidnapping detection on a physical platform will be given in Chapter 5.

After proposed kidnapping detection methods, the type of kidnapping is produced as the output of the kidnapping detection. Based on this specific information, it is possible to propose efficient methods for kidnapping recovery.

Chapter 4

Kidnapping Recovery

After performing DKDR process, 2 categories of kidnapping are recognized. Based on different kidnapping type, two kidnapping recovery methods are proposed in this chapter.

For type B kidnapping, since the robot is kidnapped around the robot pose before kidnapping due to a specific reason, a global localization with a map built by SLAM before kidnapping can be applied to perform kidnapping recovery to obtain the correct robot pose. Because the map built by SLAM before kidnapping has unified uncertainty, the existing global localization methods that assume the map provided for localization is accurate enough cannot be directly utilized. To solve this problem, a new global localization method based on Monte Carlo Localization is proposed with taking the map uncertainty into account. Based on the kidnapping detection result that the current robot pose is still around the robot pose before the kidnapping, the range of spreading articles are limited to fulfill the requirement that the robot should not move and to increase the efficiency of the method.

About type A kidnapping, the robot is kidnapped to an uncertain area that could be the explored area or the unexplored area. In this case, a method combining a global localization and a SLAM is proposed to determine the correct robot pose. The method performs a global localization with the map built by SLAM before kidnapping and a new SLAM process simultaneously. After each method obtaining an estimated robot pose, the probabilities of each estimated robot pose are compared to choose the estimated robot pose

with a higher probability as the output of the method. For implementing the idea of the method, a Monte Carlo Localization (MCL) and a particle filter based SLAM are combined as an example to show the performance of the method.

4.1 Monte Carlo Localization with Map Uncertainty

For obtaining the correct robot pose after kidnapping in the explored area, an improved Monte Carlo Localization called Monte Carlo Localization with Map Uncertainty (MCL-MU) is proposed according to the properties of the kidnapping type. First, a problem statement based on the previous particle weight evaluation principle in MCL is described to show the motivation of the study. Second, an improved MCL combing the map uncertainty into particle weight evaluation is described. To prevent the robot from moving out of the explored area, the method should be carried out without the robot motion. The method fulfills these requirements are described at the end of this section.

4.1.1 Problem Statement about Map Uncertainty

Monte Carlo Localization is a method based on particle filter for realizing robot localization with a given environment map. Since the robot cannot perform localization excluding noise including control noise and sensor noise, many hypotheses of robot poses are generated due to the noise of the robot system. Each particle represents each hypothesis of the robot pose, and the distribution of likely robot pose is represented by particles distribution.

The method takes the previous belief of the robot pose $X_{t-1} = \{x_{t-1}^1, x_{t-1}^2 \cdots, x_{t-1}^M\}$, actuation control input u_t, and sensor readings z_t as the input at time step t, where x_{t-1}^m represents the m_{th} particle of the robot pose at time step $t-1$. At time step 0, particles are dispersed randomly all over the free space in the map if the initial robot pose with uncertainties is unknown. If the initial robot state is given, particles are spread according to the given robot pose distribution.

The procedure of the basic MCL in each time step is shown in Table 4.1. Basically,

Algorithm 1: MCL(X_{t-1}, u_t, z_t)

$\bar{X}_t = X_t = \emptyset$

for $m = 1$ *to* M **do**

  $x_t^m = prediction\_phase(u_t, x_{t-1}^m)$

  $w_t^m = update\_phase(z_t, x_t^m)$

  $\bar{X}_t = \bar{X}_t + < x_t^m, w_t^m >$

endfor

for $m = 1$ *to* M **do**

  draw x_t^m from $\bar{X}_t$ with probability $\propto w_t^m$

  $X_t = X_t + x_t^m$

endfor

return X_t

Table 4.1. Monte Carlo Localization

prediction phase and update phase are processed in MCL to obtain each particle x_t^i and its weight w_t^i at time t. In prediction phase, each particle x_{t-1}^i is applied in motion according to the motion control input u_t. The motion model we used is described in Chapter A. Since the motion control input includes noise, as a result, the particles diverge during this phase. By doing so, each particle x_t^i at time t is generated. In this phase, any absolute measurements have been not yet incorporated. The absolute measurement z_k is taken into account in update phase. The weight of each particle w_t^i is generated according to x_t^i and z_t including the noise of the sensor. The measurement model we used is described in Chapter B. After these two phases, a new set of particles X_t is generated according to the weight of each particle w_t^i.

Two problems make the ordinary MCL cannot become the kidnapping recovery method for type B kidnapping. The first problem is the uncertainty of the map built by previously excused SLAM before the kidnapping. Since MCL assumes that the given map is accurate enough, the uncertainty of the map is not considered in the method. In the update phase, the uncertainty of the map is not included as the input that makes output w_t^i cannot

represent actual situation. This assumption makes the method cannot obtain an accurate result with the map built by SLAM because the different part of the map has distinct degree of uncertainty.

The second problem is about the motion of the robot. Since the type B kidnapping shows that the robot is still around the kidnapped place, the robot should not move far before the correct robot pose is recovered. Otherwise, the robot may move out of the mapping area which makes the kidnapping recover fail. However, current MCL cannot estimate robot pose without motion control input u_t due to the particle deprivation problem [76, 51, 37, 49]. If the particles all converge to an erroneous robot pose, the repeat measurement from the same robot pose may cause scarcer of particles on each time step until the particles are decreased to 0.

4.1.2 Implementation of MCL-MU

Since a new set of particles X_t at time step t is draw according to the weight of each particle w_t^i calculated by z_t and x_t^i in previous MCL, the information of the map m is combined to the calculation of the w_t^i in our proposed method. The formula of the update phase is changed to:

$$w_t^i = update\_phase(z_t, x_{t-1}i, m) \tag{4.1}$$

It can be represented by the probability $P(z_t|x_t^i, m)$, which can be divided into following formula:

$$P(z_t|x_t^i, m) = P(z_t|x_t^i)P(z_t|m) \tag{4.2}$$

where $P(z_t|x_t^i)$ represents uncertainty of the sensor reading, and $P(z_t|m)$ represents uncertainty of the map, respectively.

$P(z_t|x_t^i)$ can be calculated by beam range finder model, likelihood field range finder model or other existing sensor model [11]. $P(z_t|m)$ is obtained by the feature likelihood

Algorithm 2: MCL-MU(X_{t-1}, z_t)

$\bar{X}_t = X_t = \emptyset$

for $i = 1$ *to* M **do**

  $w_t^i = update\_phase(z_t, x_{t-1}^i, m)$

  $\bar{X}_t = \bar{X}_t + < x_t^i, w_t^i >$

endfor

for $i = 1$ *to* M **do**

  draw x_t^i from $\bar{X}_t$ with probability $\propto w_t^i$

  $X_t = X_t + x_t^i$

endfor

return X_t

Table 4.2. Monte Carlo Localization with Map Uncertainty

field or occupancy of the grid map [11]. The steps of MCL-MU are shown in Table 4.2. Since the robot has no motion during the kidnapping recovery, the predict phase is not included in the MCL-MU.

For fulfilling the requirement of no motion kidnapping recovery, the particles dispersing way is changed. In the proposed method, initial particles are spread around the kidnapped place. The dispersing range is limited to the range of the absolute sensor. After dispersing particles, motion control input u_t is set to 0 in each time step. Furthermore, the particles are re-spreading when the number of particles is decreased below a threshold. Moreover, the computation cost is reduced since the required number of particles in the proposed method is smaller than the method spreading particles all over the map. With increased efficiency, the method can be performed smoothly in real time. Furthermore, it can also decrease the probability of false kidnapping recovery caused by similar places existing in the map, such as simple corridor, simple corner or simple wall, since the range of checking area is limited.

4.2 Known and Unknown Environment's Simultaneous Localization

A method named Known and Unknown Environment's Simultaneous Localization (KUESL) which simultaneously processes global localization with a given map and SLAM is described in this section. This method can deal with the most critical kidnapping problem that the robot is kidnapped to another place far away. It can recover the robot pose no matter whether the actual robot is in the explored area or not. We will introduce the basic idea of the method firstly to show the reason why the method can solve the problem. After that, an example combing modified MCL-MU (M-MCL-MU) and Gmapping [37] is described to implement our idea of the method. Furthermore, a technique comparing the probability of estimated results between M-MCL-MU and Gmapping is shown.

4.2.1 Basic Idea of KUESL

Since the type A kidnapping cannot provide the information about whether the robot is kidnapped to the explored area, two opposite hypotheses are constructed. One of the hypothesis is that the robot is kidnapped to the explored area. The other hypothesis supposes that the robot is kidnapped to a new area. Based on the two hypotheses, a global localization and a SLAM are processed independently and simultaneously. The global localization is executed based on the hypothesis that the robot is still in the explored area, and the SLAM is performed according to the hypothesis that the robot is kidnapped to a new area.

The reason to set two opposite hypotheses at the same time rather than setting one hypothesis is that the proposed idea can realize kidnapping recovery effectively without information loss. If the robot is actually kidnapped to a new area and only one hypothesis that the robot is still in the explored area exists, the global localization cannot obtain the robot pose on the map theoretically. In the case that the robot is kidnapped to the explored area with the hypothesis that the robot is kidnapped to a new area, a new SLAM

is processed with new data to estimate the robot pose in a new map built by SLAM without using the exist map information. This means that the data for the global localization is lost. The robot needs to cost more time to mapping the environment.

After a specific time period, uncertainties of estimated robot pose in global localization and SLAM are produced. By comparing the uncertainty in Gmapping and M-MCL-MU, the more certain estimated robot pose is determined as the output of the method.

4.2.2 Implementation of KUESL

For showing a typical example of implementation our idea, MCL based method which is a famous global localization method and Gmapping which is a well-known SLAM are applied to our method. Since the map using in kidnapping recovery is built by SLAM before the kidnapping, M-MCL-MU is applied with including map uncertainty. As we denoted in Section 4.1.2, the map uncertainty is added to the weights of particles in MCL-MU. In M-MCL-MU, the same weight evaluation method is applied. The only difference between MCL-MU and M-MCL-MU is the dispersing particles method in the initial situation. At initial situation, particles are spread all over the map in M-MCL-MU replacing the limited dispersing particles range in MCL-MU.

$$H(t) = -\sum_{i=1}^{M} w_t^i \log w_t^i \tag{4.3}$$

During simultaneously processing M-MCL-MU and Gmapping, motion data and measurement data are input to the two methods at the same time, and the estimated result will not be affected each other. After processing KUESL with a specific time period, two estimated robot posed are produced separately with their uncertainties. Although both M-MCL-MU and Gmapping are based on particle filter, the distributions of particles in the two methods are different. M-MCL-MU can form particles as a multimodal distribution which means this non-parametric representation can represent multiple estimate robot pose in the map. However, the particles in Gmapping is basically only tracking one estimated robot

pose. It is difficult to represent the uncertainty of estimated robot pose with the highest particle weight since it cannot be the whole distribution of particles. Instead, the entropy of statistics [77] $H(t)$ at time step t as shown in Equation 4.3 is applied to obtain the uncertainty of estimated robot pose with all particles weights. The higher entropy represents the higher uncertainty of the estimation result. In addition, the minimum number of particles should be same in M-MCL-MU and Gmapping. The estimated robot pose with lower uncertainties is chosen as the output of KUESL. If the entropy of M-MCL-MU and Gmapping is same, the result of Gmapping is picked since choosing the result of M-MCL-MU has risks to obtain a wrong estimation.

4.3 Summary

In this chapter, two kidnapping recovery methods called MCL-MU and KUESL have been introduced. For MCL-MU, the reasons why previous MCL cannot directly be applied for kidnapping recovery in previous unknown environment were described. The improvements based on basic MCL for solving the described problems are introduced. About KUESL, a basic idea combing global localization and SLAM to simultaneously performing localization in known and unknown environment was presented. Furthermore, an example for implementing the proposed idea with M-MCL-MU and Gmapping was described as well as the method utilizing entropy to determine the output of KUESL. The conducted experiments for the proposed kidnapping recovery methods will be described in Chapter 5.

Chapter 5

Experiments

In this chapter, experiments for verifying the performance of kidnapping detection and kidnapping recovery are described. Firstly, a mobile robot platform constructed with Robot Operating System (ROS) is introduced. Then, experiments for verifying the validity of the proposed two kidnapping detection methods were conducted in an indoor environment. The performance of kidnapping detection is evaluated by ROC according to the experimental results. The performance of the two proposed kidnapping recovery methods is described separately based on the experimental results.

5.1 Experimental Setup

Before describing experiments for kidnapping detection and kidnapping recovery, the experimental setup including a mobile robot platform is introduced in this section. All experiments were conducted with the mobile robot platform in this section.

The layered mobile robot platform is shown in Figure 5.1. A Roomba Vacuum Cleaning Robot is used as the mobile base for the whole platform. The Roomba communicates with a laptop is placed on top of it in one of the layers. A Laser Range Finder (LRF), which is also fixed in another layer, is used to get the information of the environment. A gamepad

Table 5.1. Specification of Components

Parts	Specification
Roomba	Roomba 530
LRF	RHokuyo URG-04LX
Laptop	CPU: 2.20GHz Intel Core i3, Memory: 4 GB DDR3
Regulator	OKI-78SR 5V 1.5A
Convertor	FTDI FT232RL 5V cable

is used to manually control the Roomba. All the algorithms are performed in the laptop on the Roomba.

The electronic structure of the platform is shown in Figure 5.2. Roomba supplies power to the LRF through a regulator. The LRF and the laptop are connected with a USB cable. Since the laptop cannot directly transfer the data using UART protocol, we added a UART to USB converter to exchange the information between the Roomba and the laptop. The laptop and the gamepad are linked with the USB as explained. In the laptop, Robot Operating System (ROS) [78] is embedded to process the input information. The components used in the platform are shown in Table 5.1.

The software structure of the whole system is shown in Figure 5.3. The Roomba, LRF and the gamepad exclusively communicate with the laptop. The laptop gets the odometry and the scan data from the Roomba and LRF respectively, using the specific drivers for each component. The gamepad can send motion commands by operator to the laptop with a specific driver. The laptop can configure the LRFs parameters to change its performance. With the Roombas driver, the whole platform can be controlled by sending the speed and direction parameters. The laptop works under Ubuntu 14.04 and Robot Operating System (ROS) Indigo. Some key parameters used in the experiments are shown in Table 5.2.

5.2 Experiments for Kidnapping Detection

Several experiments were performed to verify the performance of DKDR in a real static environment. The proposed methods were applied to Gmapping (FastSLAM 2.0) using

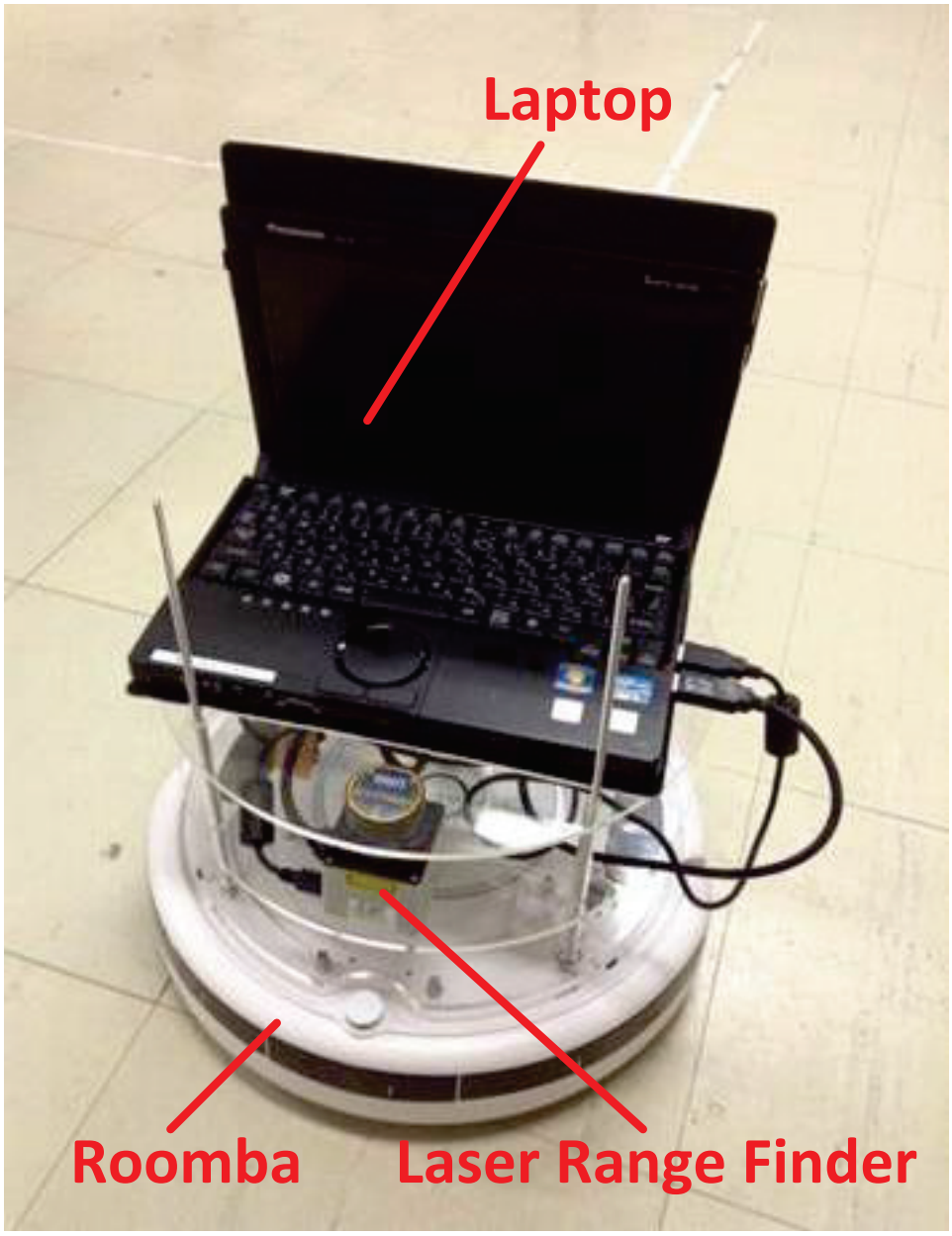


Figure 5.1. Robot Platform.

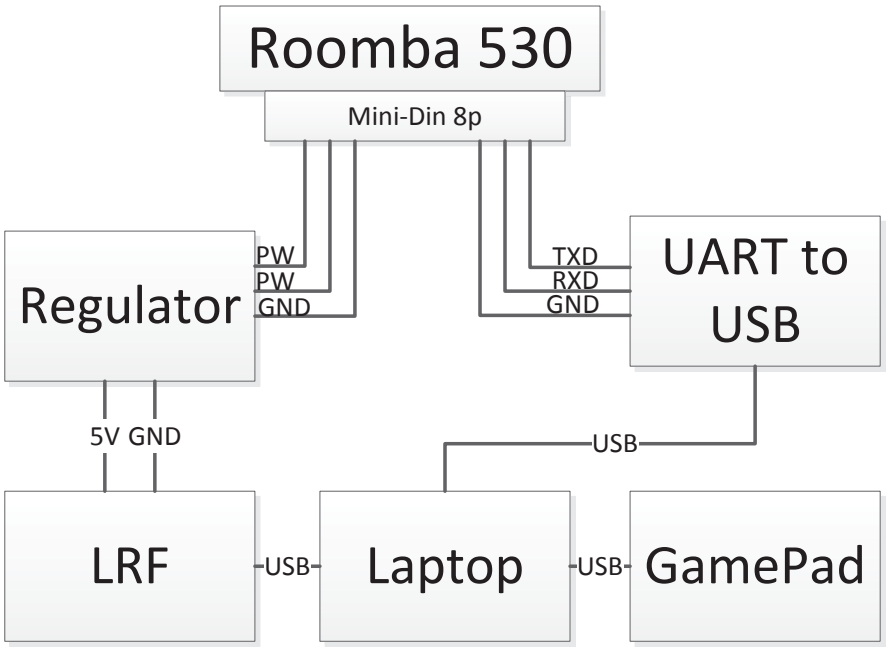


Figure 5.2. Electrical structure of the robot platform.

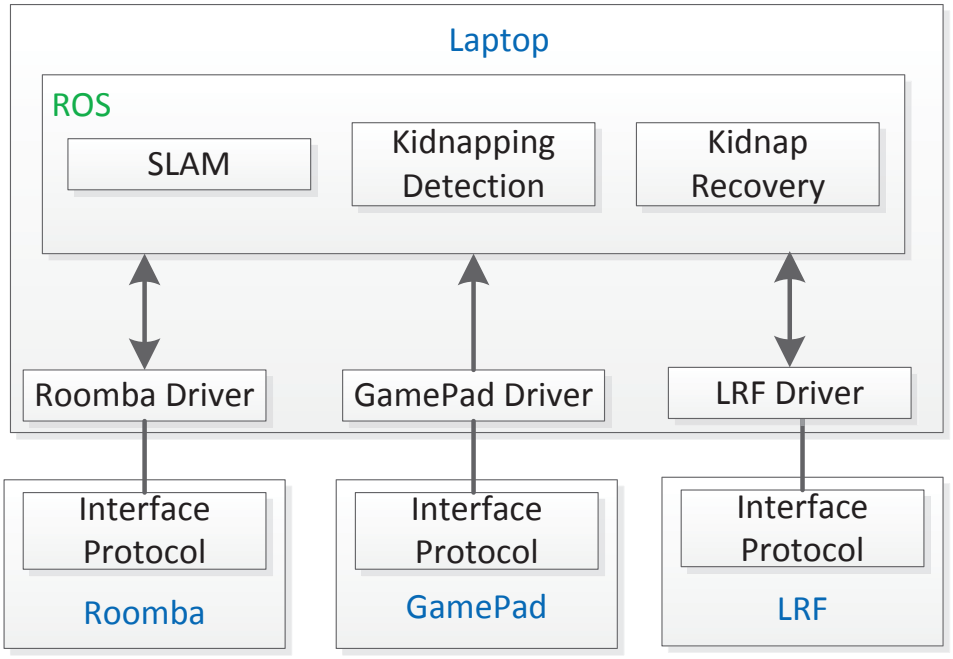


Figure 5.3. Software structure of the robot platform.

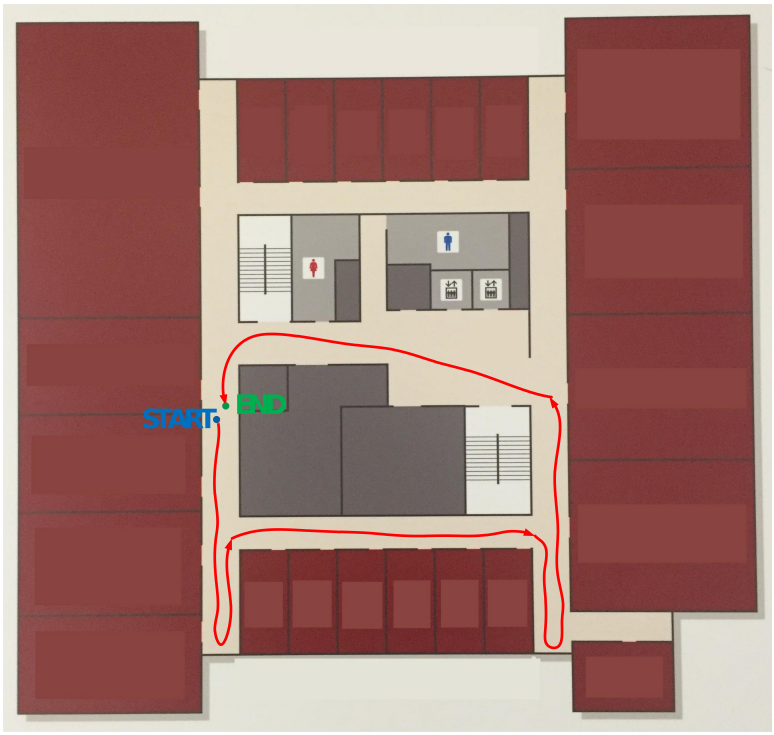
Table 5.2. Experiment Condition

Condition	**Value**
Max Range of Sensor	4m
Scan Angle of Sensor	180°
Variance of Speed Noise	0.05m/s
Variance of Orientation Noise	3°
Variance of Observation Position Noise	0.02m
Variance of Observation Angle Noise	0.36°

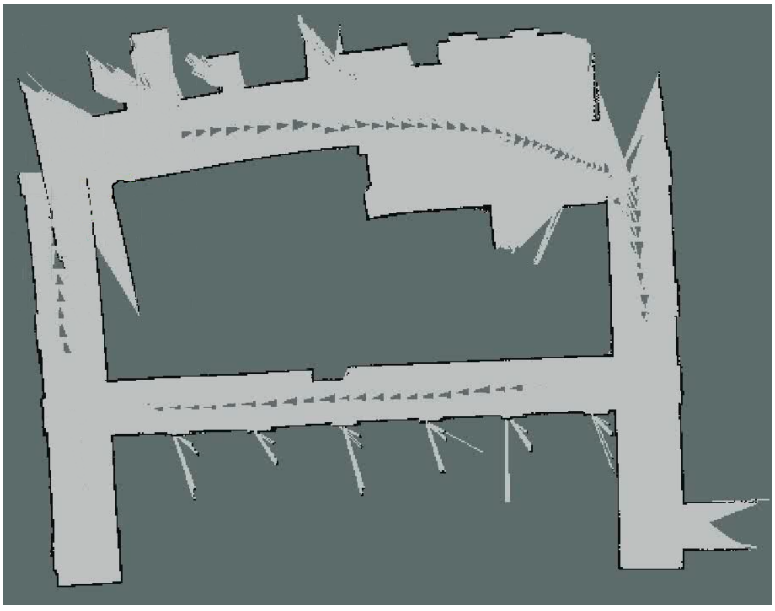
Robot Operating System (ROS) in an indoor environment. For demonstrating our method's universality, we set up a static environment to archive the requirement of Gmapping to show that DKDR can be simply applied to the existing common SLAM algorithm. If the robot is operating task under noisier conditions which makes the data association difficult, such as occlusions of features or existence of dynamic features, some existing methods can be applied to deal with these critical situations, such as a robust data association method for noisy and dynamic environment [69], JPDA (Joint Probabilistic Data Association) [79] and LSDA (Landmark Sequence Data Association) [80]. Since the DKDR just needs associated features to judge the kidnapping, these data association methods can be applied before executing the DKDR under noisy conditions. In contrast to the simulations, the filter was executed according to the movement of the robot based on the odometry data. In the experiments, the occupied grid cells were recognized as the features.

5.2.1 Experiments

The map of the environment treated as a ground truth and an example of a non-kidnapping situation with Gmapping in this environment are shown in Figure 5.4. Figure 5.4(a) depicts a floor plan of a building. Several rooms and corridors exist in this environment. A robot trajectory and its starting and ending points are shown by the a red line and the blue and green dots, respectively. This trajectory is for a non-kidnapping situation. The trajectory only shows the approximate path followed by the robot. The results obtained using Gmapping in the non-kidnapping situation are shown in Figure 5.4(b); Gmapping



(a)



(b)

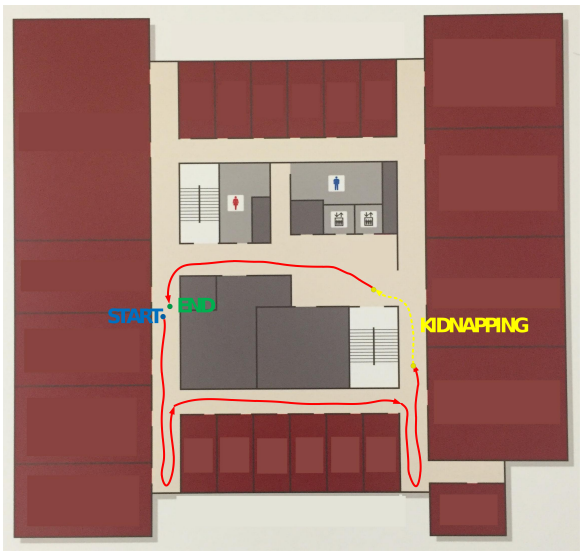
Figure 5.4. The map and the mapping result of a non-kidnapping situation. (a) Map and trajectory. (b) Mapping result.

constructed a consistent map with reasonable accuracy. Since verifying the performance of the SLAM algorithm is not our purpose, we assume that this result is acceptable. Here, there are two reasons for conducting experiments in this non-kidnapping situation: to Provide a comparison with a kidnapping situation to show the difference in the SLAM results with and without kidnapping and to test the DKDR response.

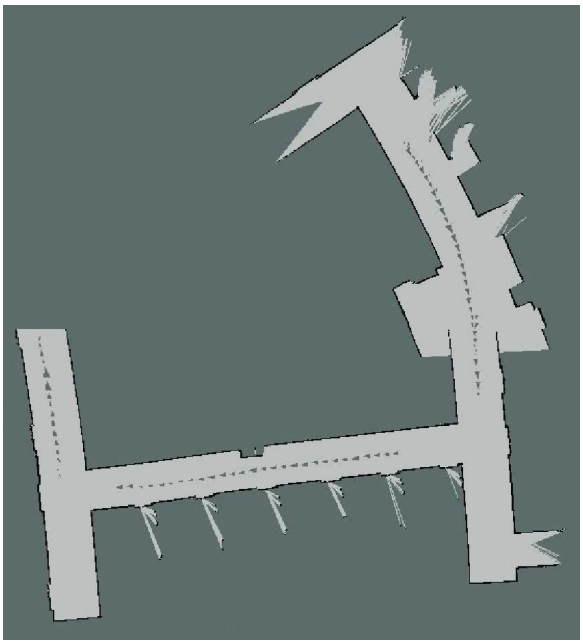
An example of a type A.2 kidnapping situation is shown in Figure 5.5. Figure 5.5(a) shows the starting and ending positions of the kidnapping event. Additionally, the trajectories for the kidnapping and non-kidnapping situations are shown in different colors. First, the robot moves from the START point (blue point) along the red trajectory until the start point (yellow point) at which kidnapping begins. No kidnapping events occur during this process. The robot is then moved by a human to the end point of the kidnapping event along the yellow dashed line. The distance between the start and the end points of the kidnapping event is about 11m. Subsequently, the robot moves to the end point (green point) along the red trajectory. During this process, the kidnapping also is not happened. Without DKDR, the information of the mapping result is directly added to the original map after the kidnapping event (Figure 5.5(b)). After kidnapping, the mapping result created by Gmapping without DKDR does not match the ground truth within a reasonable error. Figure 5.5(c) shows the result with DKDR. While kidnapping happens, DKDR detects this abnormal event and keeps the original information from being deformed by the incorrect information. Comparing the the areas of the mapping results before kidnapping in Figure 5.5(a) and 5.5(b), indicating that the original information was slightly deformed by kidnapping.

The data from the experiment described above are shown in Figure 5.6. Before kidnapping occurs at the time step 168, the values of Q_p and Q_s are lower than their thresholds. After kidnapping at time step 168, the values of Q_p and Q_s increase suddenly and exceed their thresholds. Since Q_o is also beyond its threshold, this kidnapping event is detected and eventually recognized as type A.2 kidnapping.

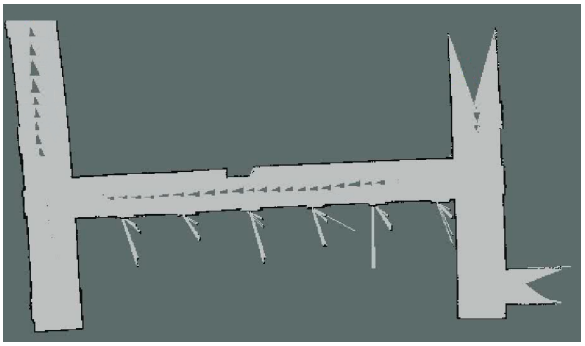
An example of type B.2 kidnapping is shown in Figure 5.7. The map shown in Figure



(a)

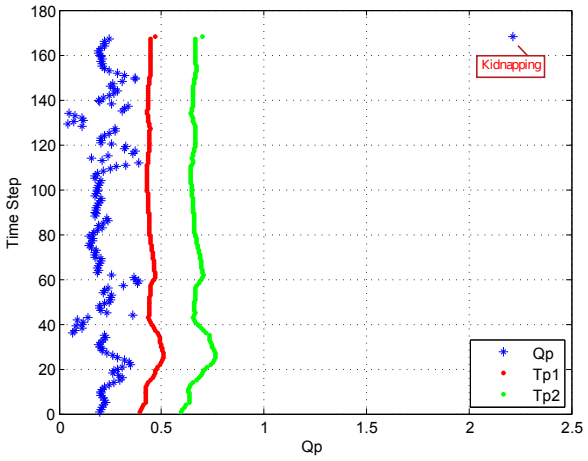


(b)

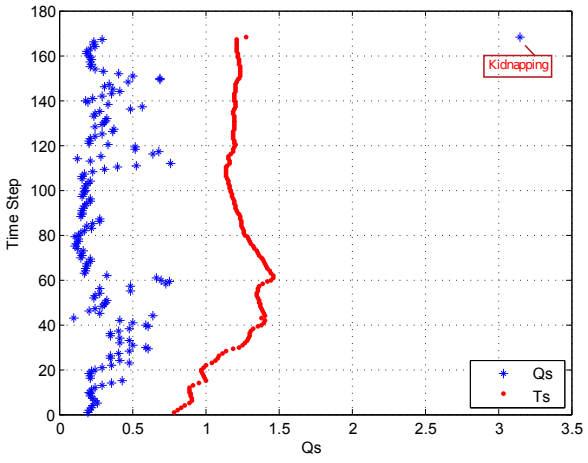


(c)

Figure 5.5. The example of type A.2 kidnapping. (a) Map and trajectory. (b) Mapping result without DKDR. (c) Mapping result with DKDR.

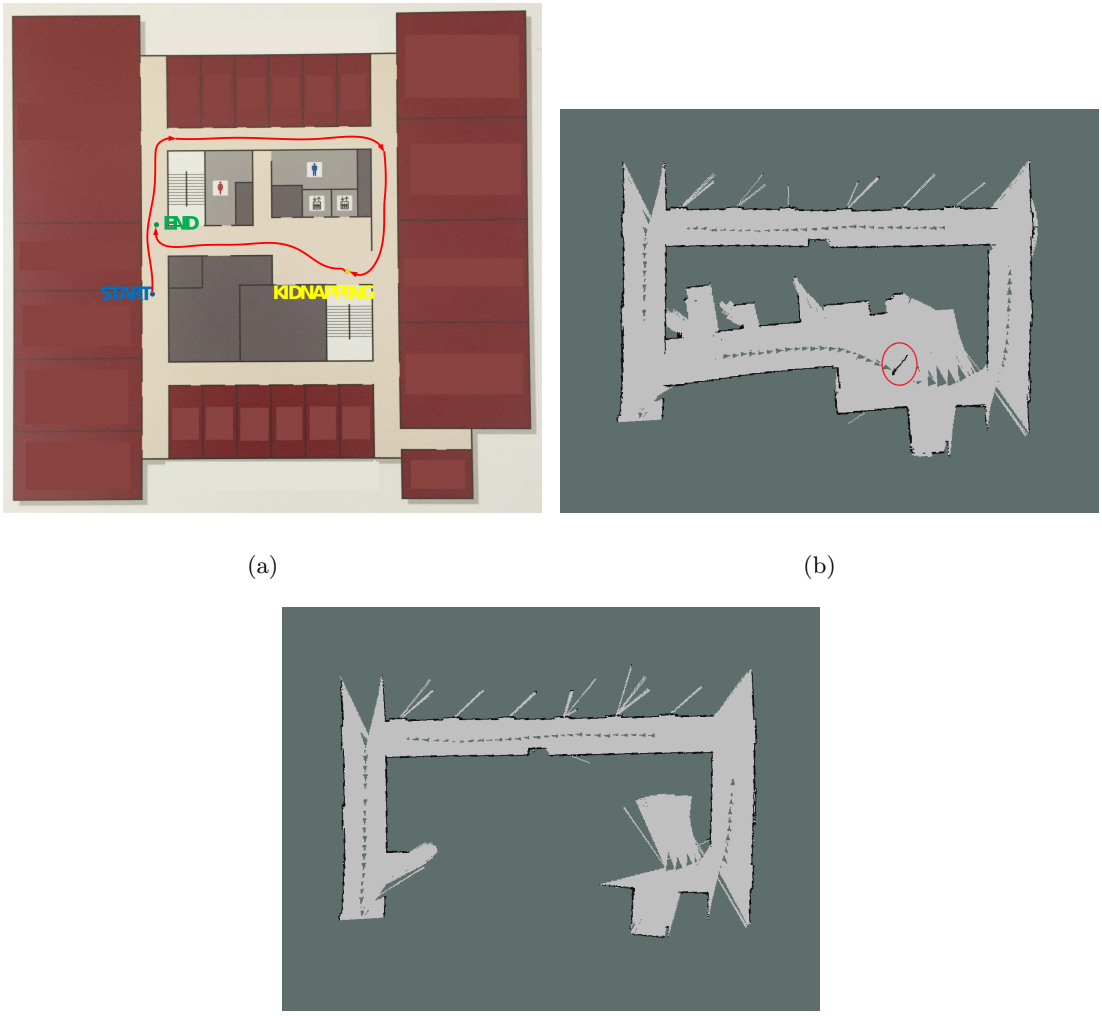


(a)



(b)

Figure 5.6. The response of metrics of type A.2 kidnapping. (a) Q_p. (b) Q_s.



(a) (b)

(c)

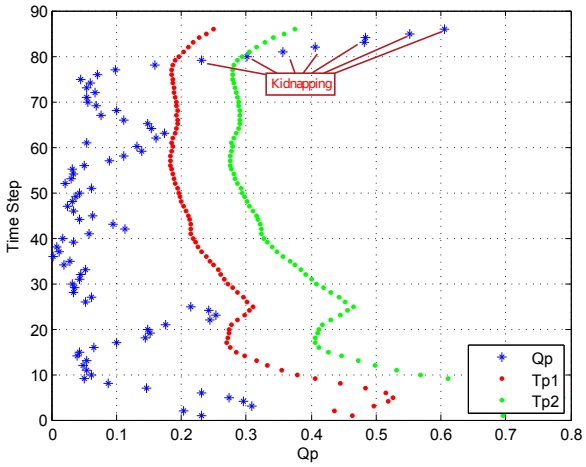
Figure 5.7. The example of type B.2 kidnapping. (a) Map and trajectory. (b) Mapping result without DKDR. (c) Mapping result with DKDR.

Table 5.3. Operating Characteristics of DKDR

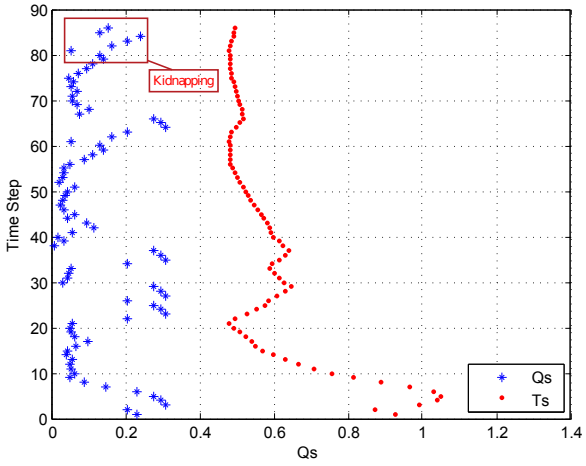
True Positive Rate	False Positive Rate
0.85	0.30

5.7(a) is the same as the map described above; the differences are the trajectory of the robot and the type of kidnapping. First, the robot starts to move along the red trajectory from the start point. When it reaches the kidnapping point, the robot is stuck in that place until the odometry reading reaches 11m. The robot then moves along the red trajectory until the end point. Figure 5.7(b) shows the mapping result without DKDR. Since Gmapping is a hybrid scan-matching and PF-based SLAM, it can correct the misalignment automatically. With the exception of a small incorrectly mapped area shown inside the red circle, the mapping result is acceptable. However, the wrongly mapped area caused by kidnapping could affect the performance of the robot's task. This kidnapping should also be detected efficiently. The mapping result with DKDR is shown in Figure 5.7(c). The original information from before the kidnapping is retained.

In the experiment with DKDR, the kidnapping was successfully detected. However, it was not distinguished correctly by DKDR in this example. Fig 5.8 shows the responses of Q_p and Q_s. In the non-kidnapping situation before time step 79, the value of Q_p is below T_{p1} and T_{p2}. When the robot gets stuck at the kidnapping point during time steps 79-86, the value of Q_p is between T_{p1} and T_{p2}. These values indicate that the prior-check process detected the kidnapping and recognized it type A.1 or type B.1 kidnapping using Q_p. Since the value of Q_o is lower than its threshold, the kidnapping is recognized as type B.1 kidnapping in the prior-check process. The response of Q_s is shown in Figure 5.8(b). Because the overall map is not significantly changed by kidnapping, the value of Q_s is always below T_s indicating that the kidnapping is not detected in the posterior-check process. According to the principle of DKDR, it reports an alarm of kidnapping and recognizes the event as type B.1 kidnapping. To analyze this failure of recognition, the values of the metrics at the time steps after the DKDR alarm are also shown in Figure 5.8 and are further discussed in the following section.



(a)



(b)

Figure 5.8. The response of metrics of type B.2 kidnapping in Figure 5.7. (a) Q_p. (b) Q_s.

Table 5.4. Operating Characteristics of classification

Kidnapping Type	**TPR**	**FPR**
Type A.1	0.8361	0.0428
Type A.2	0.9583	0.0764
Type B.1	0.8248	0.1022
Type B.2	0.8728	0.0403

Table 5.5. Operating Characteristics of DKDR and P-DKDR in a large scale environment

Kidnapping Detection	**True Positive Rate**	**False Positive Rate**
DKDR	0.7500	0.2417
P-DKDR	0.6500	0.1334

The experimental results for detecting kidnapping and distinguishing different types of kidnapping are shown in Table and Table , respectively. If the robot is moved more than 10m, it is recognized as type A.2 kidnapping. Type B.2 kidnapping is defined as that the robot stops at some specific point until its odometry reading exceeds 10m. The ranges of type A.1 and B.1 are set at 2m.

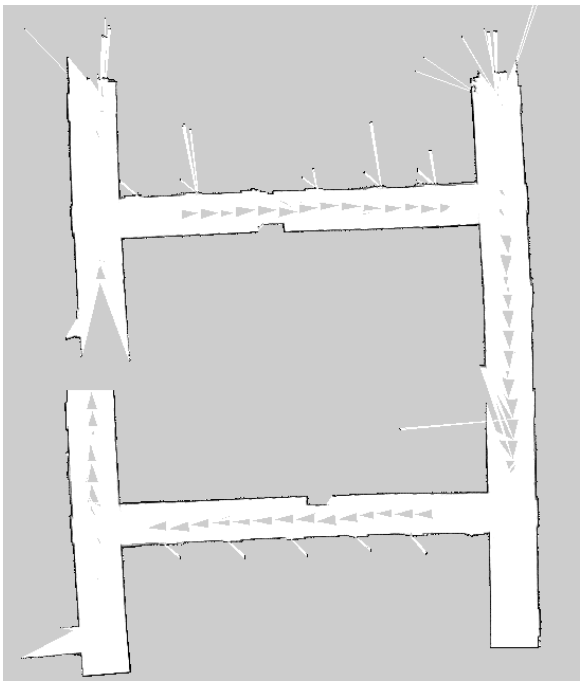
For verifying accuracy of P-DKDR in a real indoor environment, experiments with a larger environment were conducted with P-DKDR and DKDR. The map, robot trajectory for one of the experiments has been shown in Figure 5.9(a). During Gmapping process along the red line from START point to END point, no actual kidnapping happened. The response of each metric in DKDR is shown in 5.10, which is similar with its simulation result. The uncertainty of the estimation affects the estimated result according to the mapping area. P-DKDR can prevent this issue which is shown the response of each metric in Figure 5.11. The tendency of the experiment fits to the simulation result. The results for more experiments processed by ROC are shown in Table 5.5.

5.2.2 Discussion

The ability and performance of DKDR are demonstrated in Section IV. DKDR can detect and classify kidnapping with good performance. However, the thresholds employed are not optimal because the previous data are unknown. Therefore, they are not suitable for applications that require high accuracy. For real-time applications such as convenient and highly accurate detection of kidnapping, DKDR can quickly detect kidnapping that only requires the predicted percentage of kidnapping in the total events with half-normal distribution.

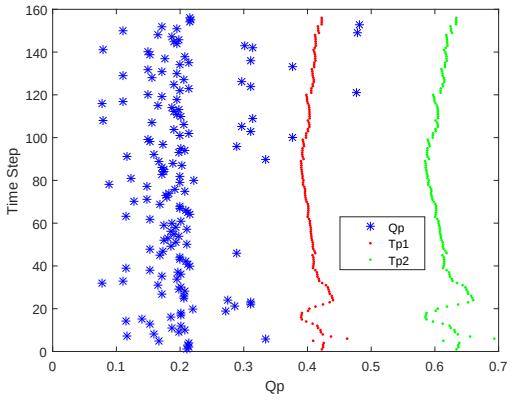
The detection performance of DKDR determined based on the simulations and exper-

(a)

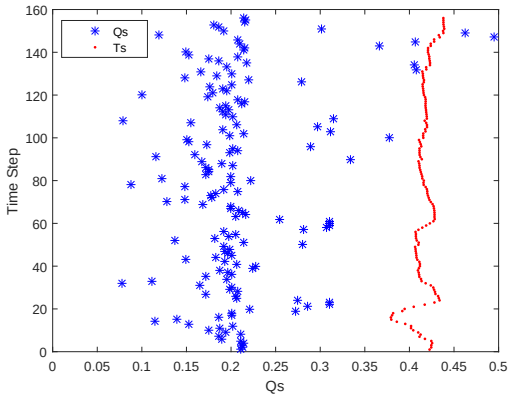


(b)

Figure 5.9. The map, robot trajectory and Gmapping result of a large indoor environment.(a) The map and robot trajectory for the experiment. (b) Result of Gmapping without kidnapping.

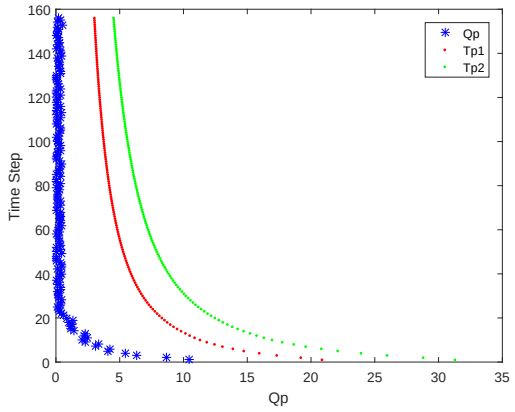


(a)

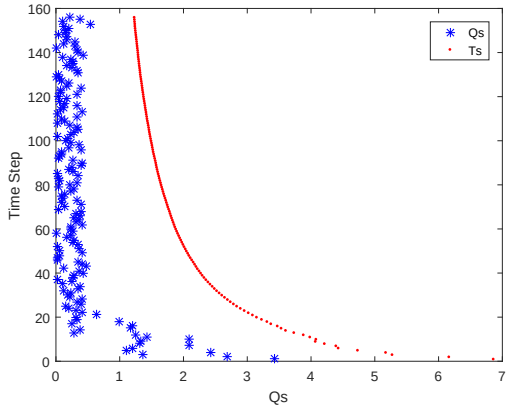


(b)

Figure 5.10. Non-kidnapping with DKDR. (a) Response of the metric Q_p. (b) Response of the metric Q_s.



(a)



(b)

Figure 5.11. Non-kidnapping with P-DKDR. (a) Response of the metric Q_p. (b) Response of the metric Q_s.

iments is clearly not same. First, the laser beams of LRF are not always stable; some of them cannot reflect effectively, although they have projected to the obstacles, such as the grass or the smooth curved surface. In this case, DKDR cannot detect kidnapping. Second, if the robot is moved into a place that is similar to the environment around the kidnapped position, such as in a corridor without any other apparent features, the sensor is difficult to distinguish the differences between the environments before and after kidnapping, which makes DKDR failure.

The performance of recognition is similar in the simulations and experiments with the exception of a couple of points. The true positive rate of type B.2 kidnapping is lower, and the false positive rate of type B.1 is significantly increased because some actual type B.2 kidnapping is recognized as type B.1 kidnapping. In our experiments, the filter is conducted according to the data from the wheel odometry. When the robot is stuck in an area, the odometry data is increasing. Although the robot is stuck until the odometry data get over 10m, the filter was executed many times. The data of the example is shown in Figure 5.8(a). After that, the value of Q_p exceeded T_{p1}, and the value continued to increase beyond T_{p2}. DKDR only handles kidnapping in one time step, the recognition failed in this situation. However, we cannot change the wheel odometry data directly in the simulation, because this change does not fit the real situation of a stuck robot. This special case will be discussed in a future work.

The results as a whole show that type A.1 and type B.1 kidnapping events cannot be detected as accurately as the other types of kidnapping do. This reduced performance is caused because this type of kidnapping needs to satisfy two conditions:

1. The robot needs to be unexpectedly moved within a short distance, which is not easy to detect.

2. The position of the robot is near the position that it moved from, which is difficult to measure. Moreover, T_{p1} and T_{p2}, which are determined by the probability of kidnapping, are not optimal thresholds.

Only three metrics and three thresholds need to be calculated in the DKDR method, therefore, the efficiency of the method is acceptable. More importantly, DKDR can be applied in many SLAM algorithms that contain three basic processes, predict, observe and update. These characteristics allow DKDR to be applied widely and conveniently. Unlike previous methods, DKDR provides kidnapping detection and recognition based on different situations after kidnapping.

About the results of DKDR and P-DKDR in the experiments and the simulations described in Chapter 3, the tendency of the experiments and the simulations is same. Although the whole performance of the experiments results is worse than the simulations results, it is foreseeable because of the reasons described before.

5.3 Experiments for Kidnapping Recovery

For verifying validity and performance of MCL-MU and KUESL, experiments were conducted with the introduced mobile robot platform and the indoor environment. After obtaining the results of experiments, a discussion is presented to provide a analysis of failure cases.

5.3.1 Experiments for MCL-MU

For showing the validity of MCL-MU, experiments were conducted after kidnapping detection. As shown in Figure 5.12, the robot slips around a spot after obtaining a map of partial environment by SLAM with a given trajectory between START point and KIDNAPPING point. This phenomenon belongs to type B kidnapping which the MCL-MU should be carried out to realize kidnapping recovery.

The results of MCL-MU according to the time step is shown in Figure 5.13. At the initial step, particles represented by the red arrows are dispersed around the kidnapping point bounded with the sensor range. The write points represent the current sensor scan from the most likely estimated robot pose. After the time passing with no motion on the

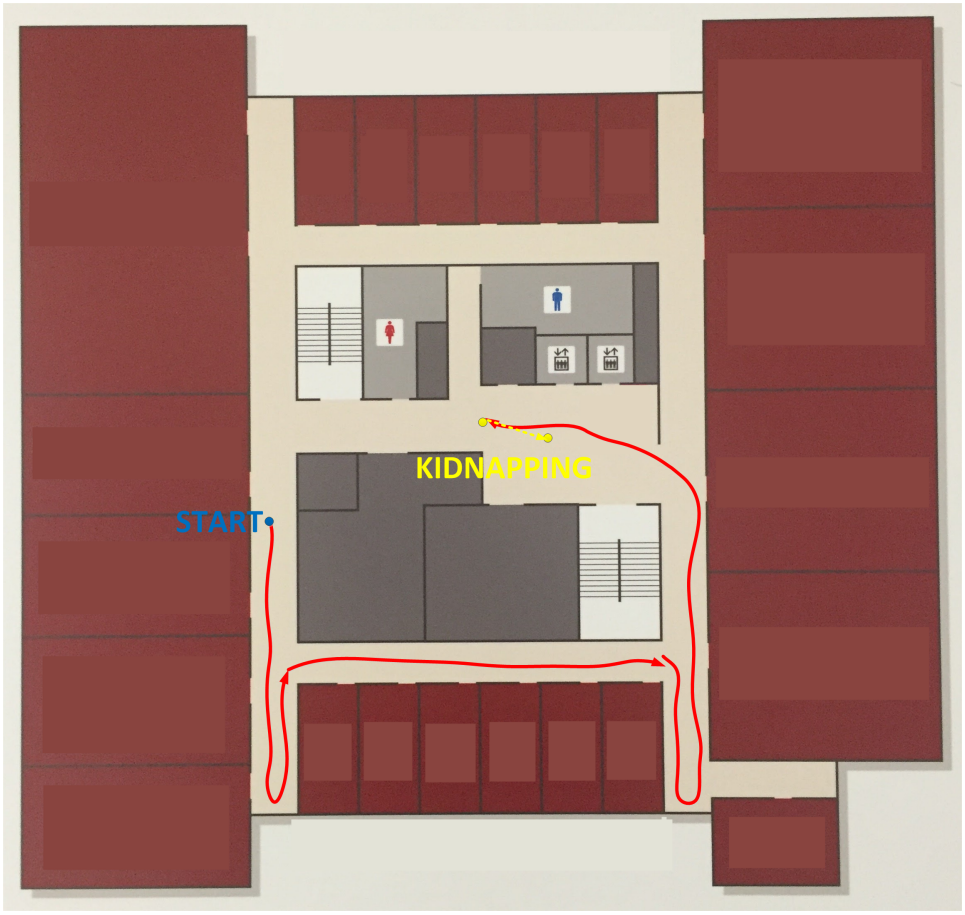
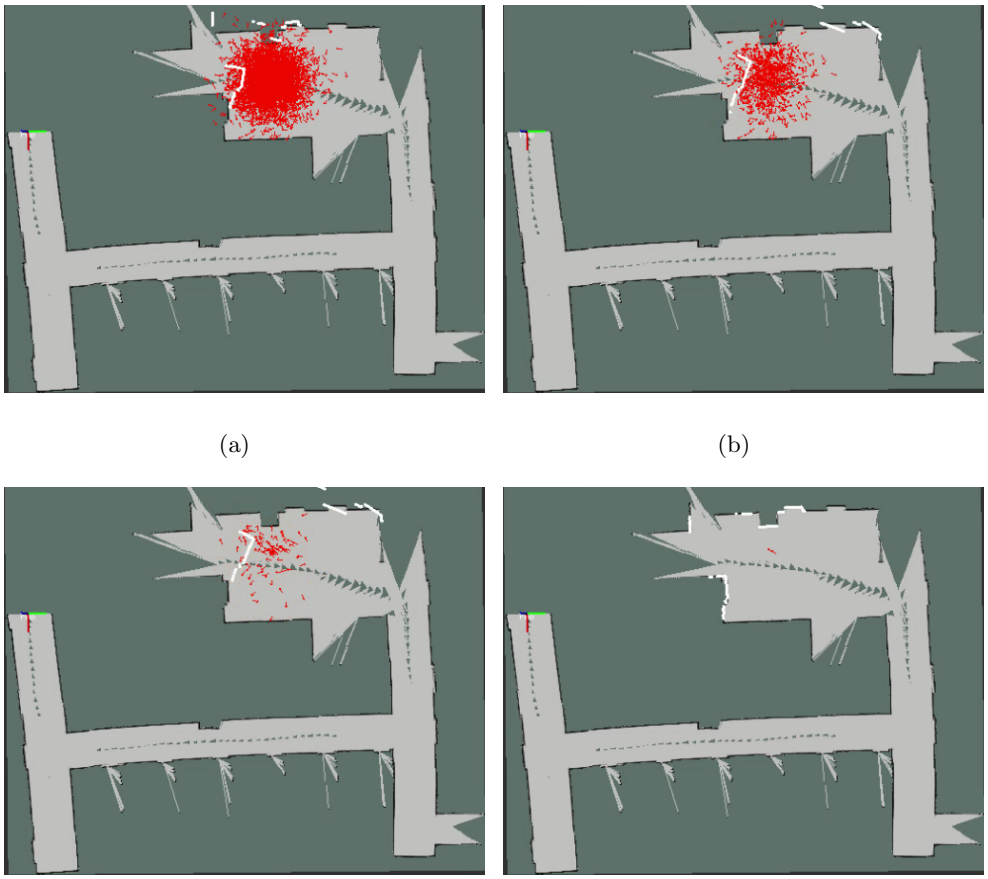


Figure 5.12. The map and robot trajectory of slipping situation.



(a) (b)

(c) (d)

Figure 5.13. The result of MCL-MU at different time. (a) Initial Step (Time = 0s). (b) Time = 21s. (c) T = 32s. (d) T = 48s.

robot, the number of particles is decreasing which means the wrong hypothesises of the robot pose are eliminated according to the principle of MCL-MU. After a certain time, the number of particles keeps stable and the particles are gathered around a specific spot. Furthermore, the current sensor scan matches to the edge of the map properly. The particle with the highest weight is selected as the output of the MCL-MU.

After performing 40 times MCL-MU with distinct kidnapping point, the successful rate is 0.85. The succussed recover is defined as the distance between the estimate robot position and the actual robot position is under 0.1 m, and the variation of the estimated robot orientation and the actual estimated robot orientation is below the 10 degree. The failure cases and their reason are discussed in the Section 5.3.3.

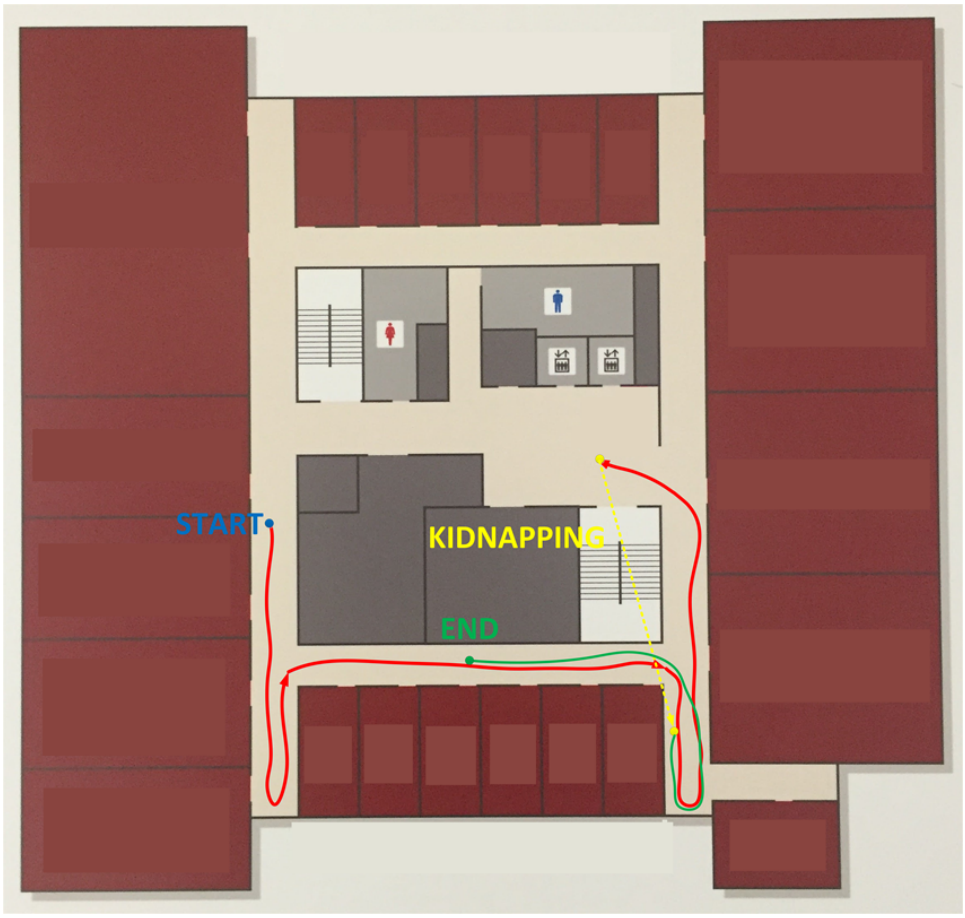


Figure 5.14. The map and robot trajectory of kidnapping to the explored area.

5.3.2 Experiments for KUESL

Since there are two situations in type A kidnapping, experiments are divided into two groups. In one of the groups, the robot was kidnapped into the explored area. In the other group, the robot was kidnapped into a new area. The performance of KUESL is evaluated according to these two groups experiments separately.

As shown in Figure 5.14, the robot follows desired trajectory along the red line from START point. During the robot following the desired trajectory, Gmapping is performed to obtain the estimated robot pose and the map of the environment. After reaching the KIDNAPPING point, the robot is kidnapped to a point in the explored area beyond the range of the sensor. KUESL was excused after the end of kidnapping to recover the robot

pose since this unexpected event belongs to type A kidnapping. During performing KUESL, the robot explored the surroundings along the green line until reaching the END point.

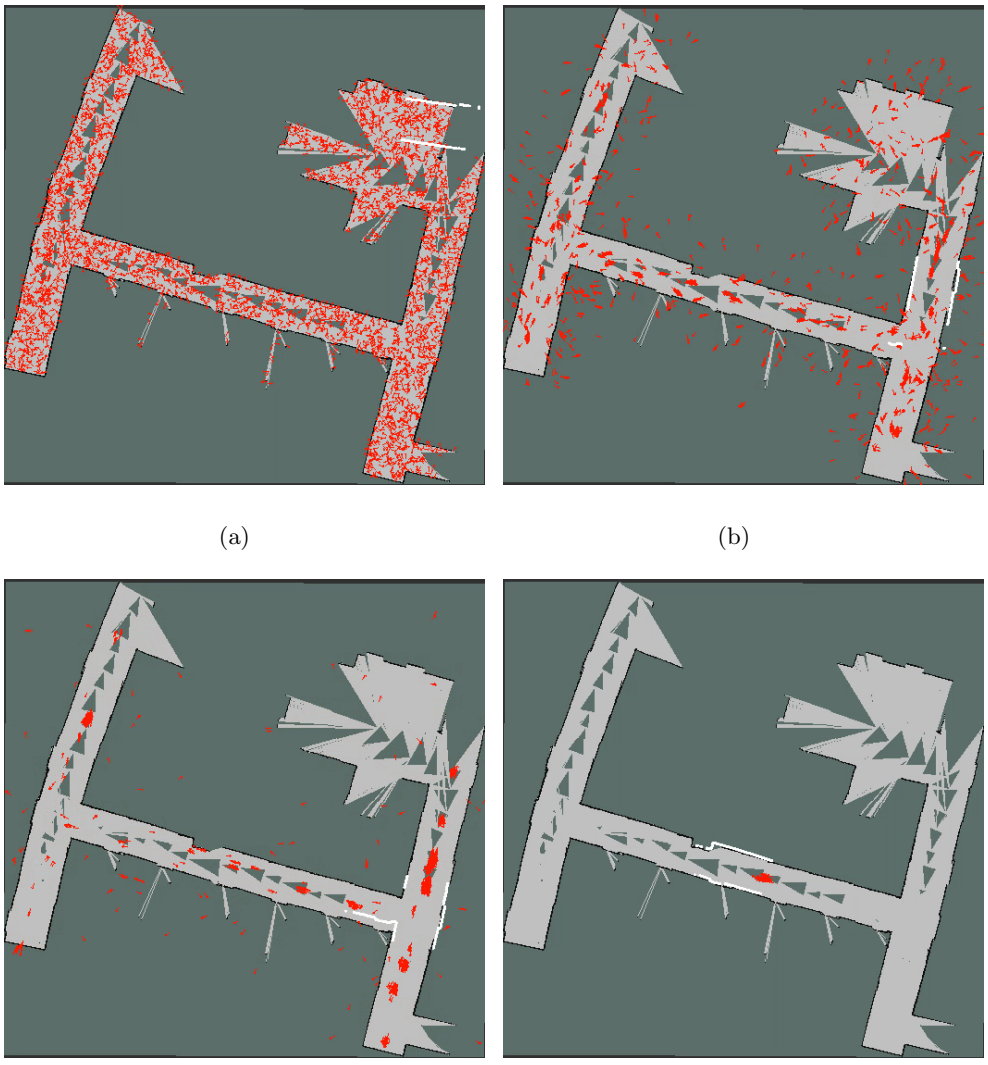
The results of KUESL in M-MCL-MU part according to the passing time are shown in Figure 5.15. Firstly, the particles represent as red arrows are spread all over the map. After time flowing, particles were gathered into several spots. At the end, particles are only around a specific spot. Moreover, the current sensor scan matches to the edge of the map.

In the same time period, Gmapping in KUESL was also processed. Figure 5.16 shows the results of the initial situation and the final mapping result. In the initial situation, the Gmapping only obtained a small map with current sensor readings. At the end of the KUESL, the map of the partial environment from SLAM is obtained which is similar to the ground truth.

The entropy from M-MCL-MU and Gmapping according to time step is shown in Figure 5.17. Before time step 41, the entropy of M-MCL-MU is larger than the entropy of Gmapping, which means the estimated robot pose of MCL has higher uncertainty than the result produced by Gmapping. After that, the M-MCL-MU entropy is under the Gmapping entropy which means the estimated robot pose from M-MCL-MU is selected as the output of KUESL.

The example of the other group experiments is shown in Figure 5.18. The difference between the previous experiment is that the robot is kidnapped into a room which is a totally new area. After kidnapping event, the robot moved around surroundings until the END point. The result of Gmapping in KUESL shown in Figure 5.19, which means the map of the room is constructed.

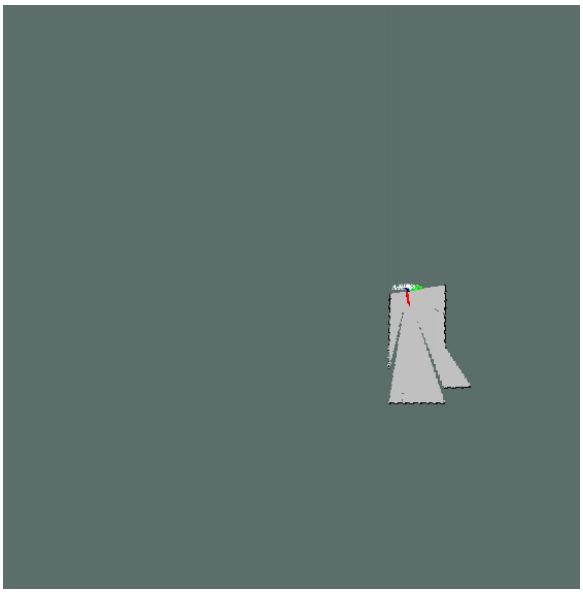
The result of M-MCL-MU in KUESL is shown in Figure 5.20. After spreading particles in the initial time step, the number of particles is decreased a little. However, the particles did not aggregate into one spot during a certain time. Furthermore, the sensor scan does not match to the edge of the map during this time period. This means M-MCL-MU did not



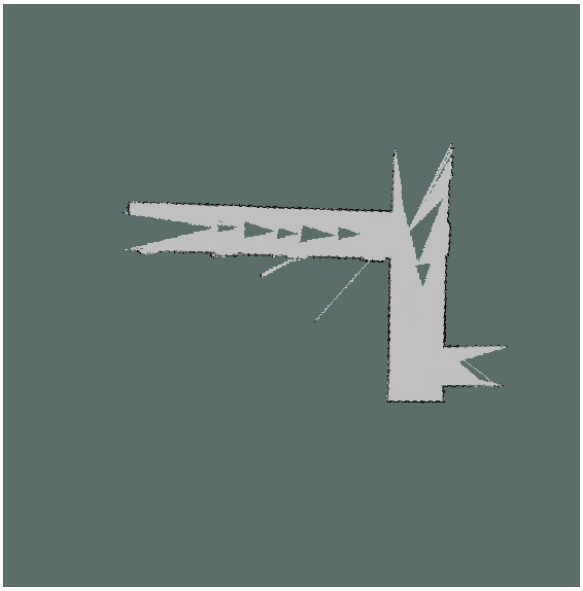
(a) (b)

(c) (d)

Figure 5.15. The result of M-MCL-MU in KUESL with kidnapping into the explored area at different time. (a) Initial Step (Time = 0s). (b) Time = 19s. (c) T = 35s. (d) T = 63s.



(a)



(b)

Figure 5.16. The result of Gmapping in KUESL when the robot is kidnapped to the explored area. (a)Initial situation. (b) Final map.

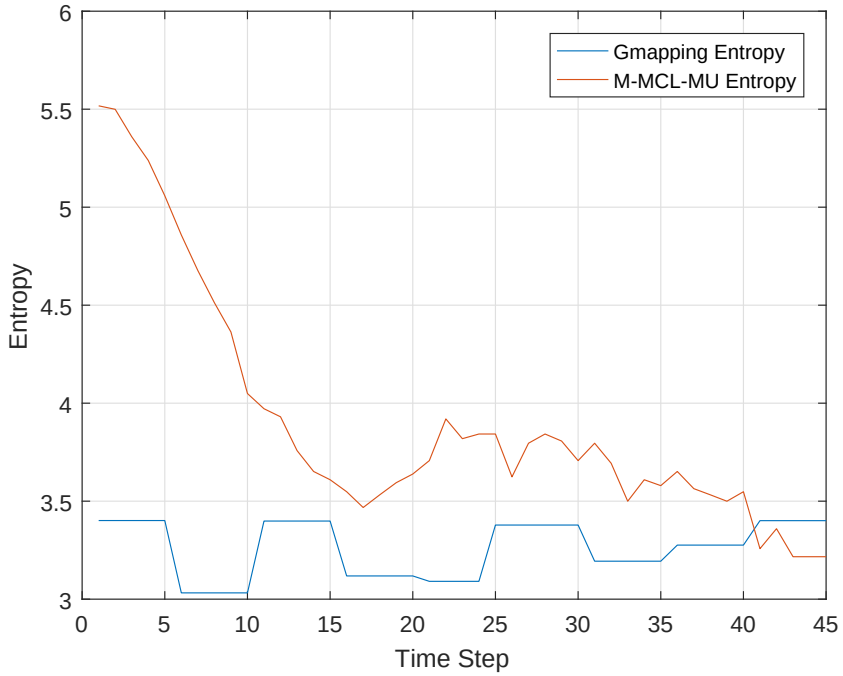


Figure 5.17. The entropy of M-MCL-MU and Gmapping during the experiment in the explored area.

obtain a good estimated result. The state of entropy during this period from Figure 5.21 shows that the estimated robot pose from M-MCL-MU has higher uncertainty all along.

For evaluating the performance of the KUESL, a performance index is defined. The event E is defined as the estimated robot pose is in the explored area. On the contrary, the estimated robot pose in a new area is denoted as event N. Based on E situation, true E which means the actual robot pose is also in the explored area, and false E is the opposite situation which the robot pose is in the new area. Similar to true E and false E, the difference between true N and false N is whether the actual robot pose is same as the N situation.

With randomly choosing kidnapping spot in the environment 20 times in each group, the performance of KUESL is shown in Table 5.6. Further analysis is described in Section **??**.

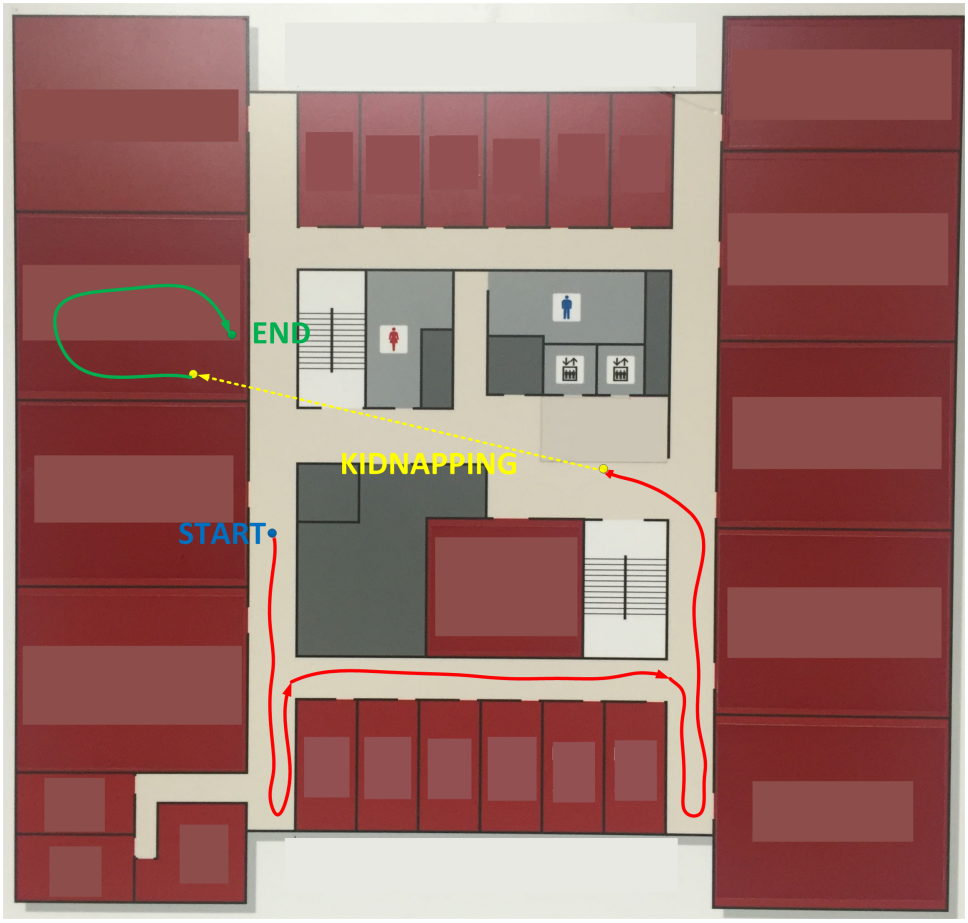
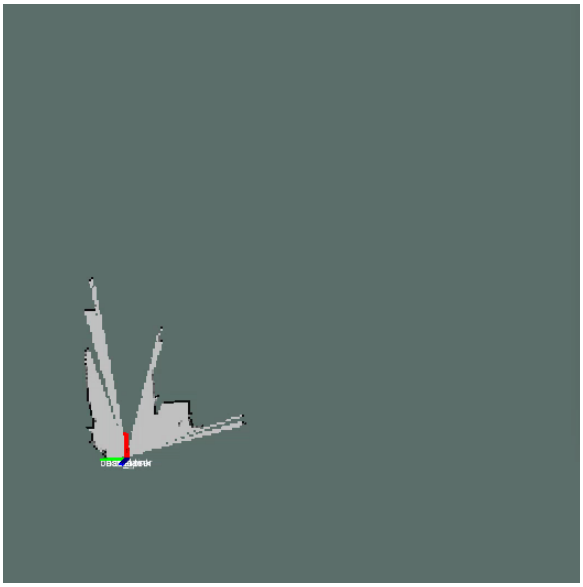


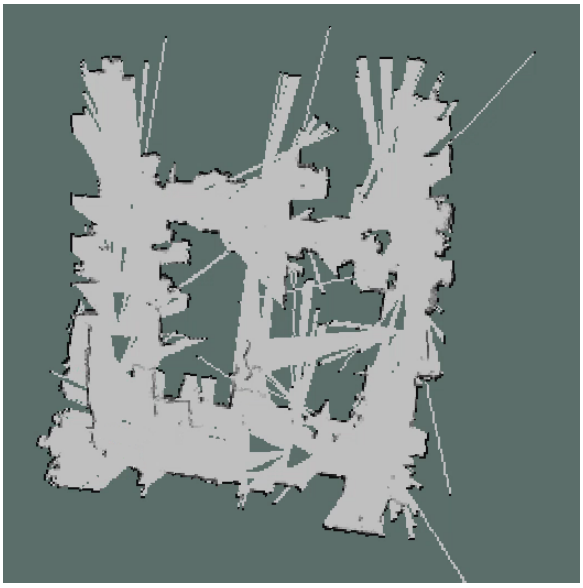
Figure 5.18. The map and robot trajectory of kidnapping to a room (new area).

Table 5.6. Performance of KUESL

True E Rate	**True N Rate**
0.85	0.70

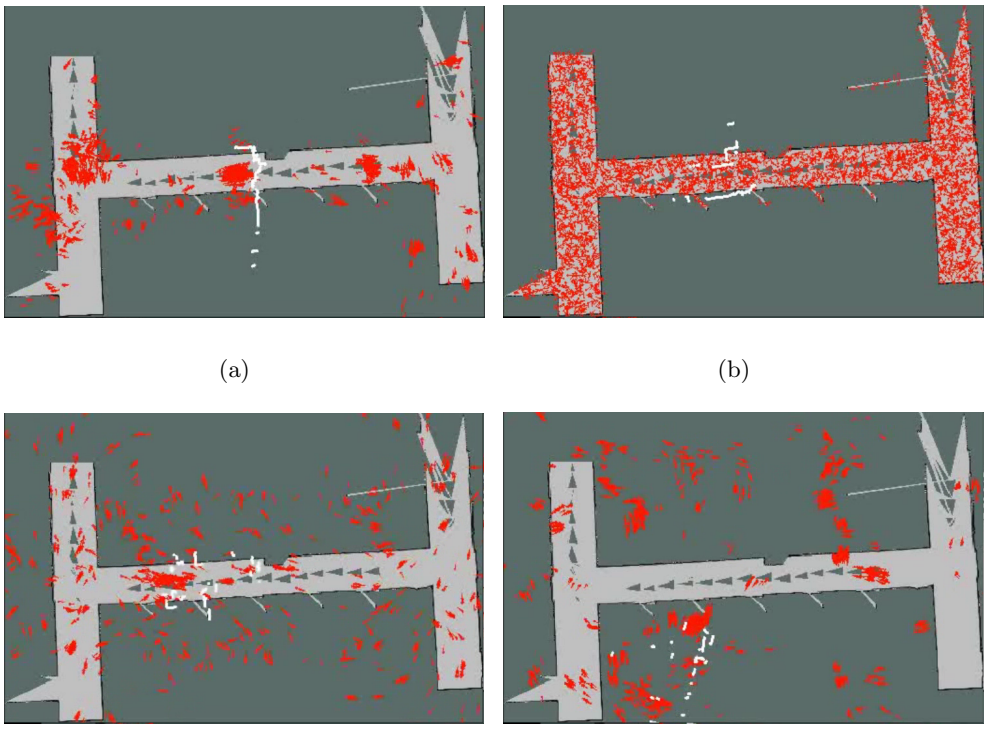


(a)



(b)

Figure 5.19. The result of Gmapping in KUESL when the robot is kidnapped to a room (new area). (a)Initial situation. (b) Final map.



(a) (b)

(c) (d)

Figure 5.20. The result of M-MCL-MU in KUESL with kidnapping into a new area at different time. (a) Initial Step (Time = 0s). (b) Time = 20s. (c) T = 30s. (d) T = 38s.

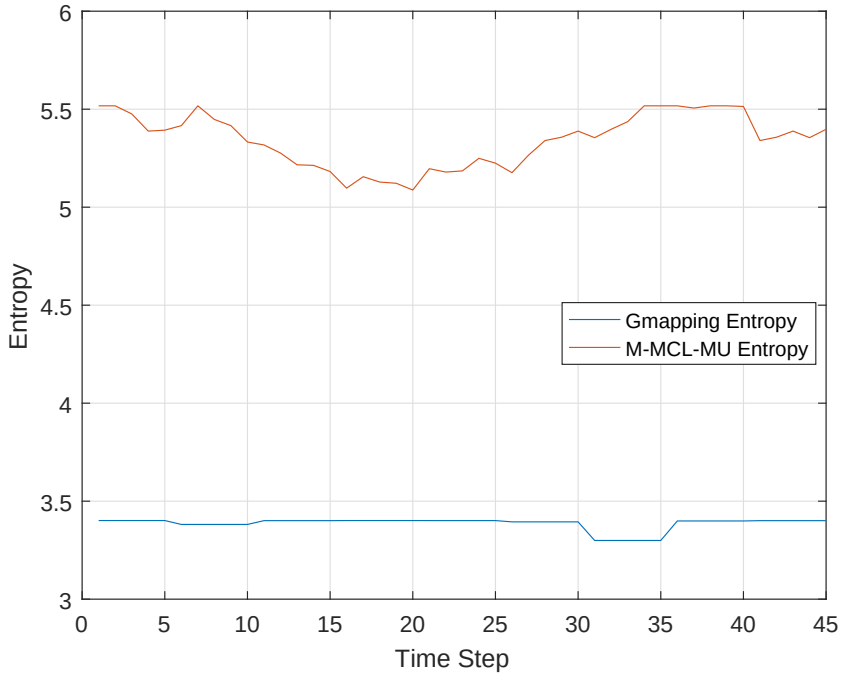


Figure 5.21. The entropy of M-MCL-MU and Gmapping during the experiment in the new area.

5.3.3 Discussion

About the failure cases of kidnapping recovery in MCL-MU, an analysis was carried out to find out the reason. After checking the environment map and sensor reading after the kidnapping, we found that MCL-MU may cause confusion if several similar scenes exist in the map. For example, if there is a corridor with only two detectable lines represent as walls in two side, MCL-MU will produce multiple hypotheses of the robot pose, as shown in Figure 5.22. Red arrows represented as particles are gathered on two lines. This result shows that this kind of rare situation causes the failure of the kidnapping recovery. However, this problem can be solved by adding more sensors on the robot system, such as camera sensors, to obtain more information to distinguish the similar scenes.

For discussing the nonsuccess result of KUESL, reasons are described in two aspects which are false E and false N. The reason of false E is that the particles were not perfectly dispersed all over the map. In this case, there are no particles around the actual robot pose

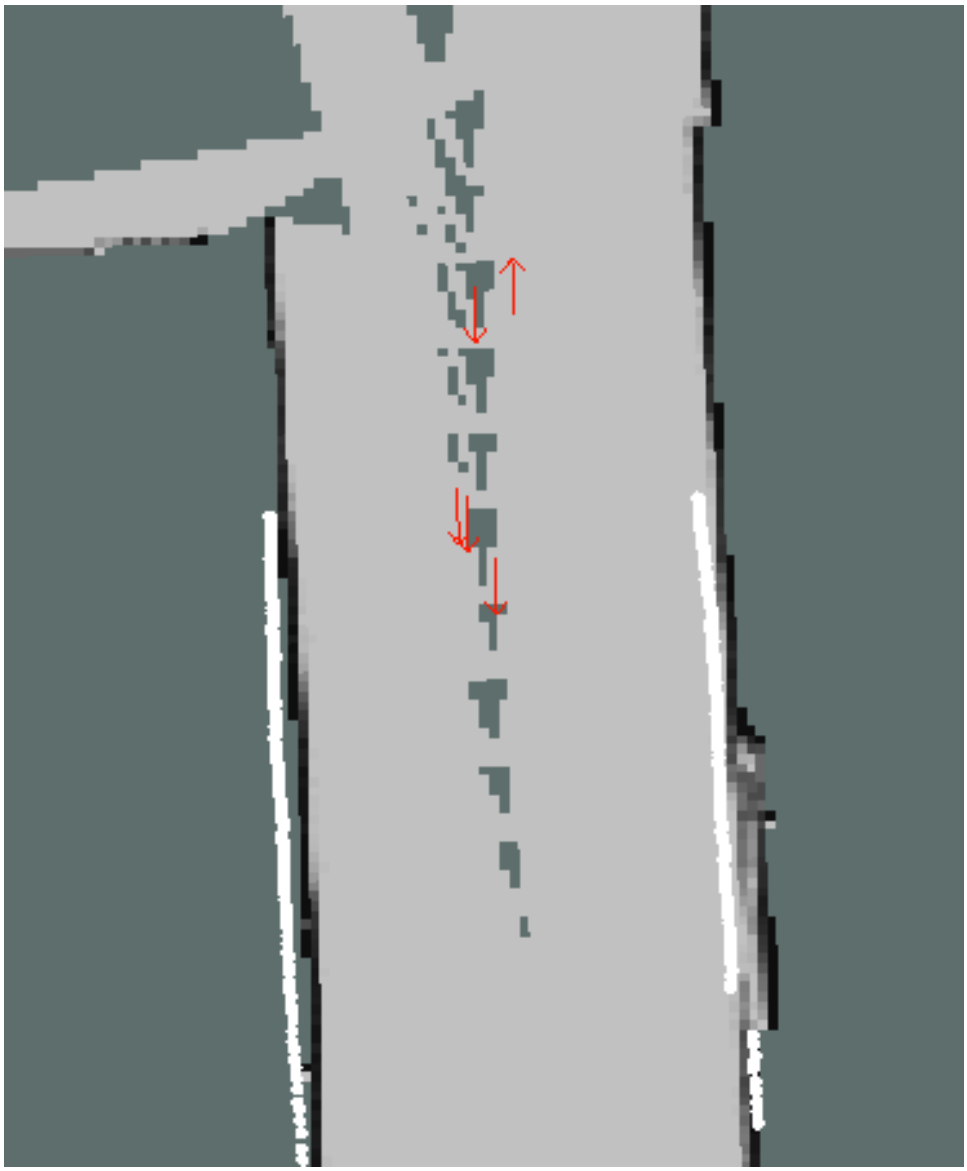


Figure 5.22. Several robot pose hypothesis in a corridor environment.

which makes KUESL cannot obtain a good result. Most false N cases are caused by similar scenes in the map which is the same failure reason of MCL-MU. Since the map of KUESL generally bigger than the map of MCL-MU, the possibility of existing similar scenes in the map of KUESL becomes bigger. Furthermore, in true E, the similar scene also affects the results. If we set the distance and angle of orientations between actual robot pose and estimated robot pose under 0.1m and 10 degrees as a successful recovery, the success rate in true E rate is 0.824 (14/17).

5.4 Summary

In this chapter, the experiments of kidnapping detection and kidnapping recovery are introduced separately. About the kidnapping detection, the validity of DKDR is verified by man-made Type A kidnapping and Type B kidnapping separately. More experiments were conducted to show the performance of DKDR with ROC. The experiments of DKDR and P-DKDR in a whole floor of the environment were conducted to verify the different performance of DKDR and P-DKDR in a large scale environment. The results of the experiments about DKDR and P-DKDR have the similar tendency of their simulations.

The experiments of kidnapping recovery were conducted according to the different situations. The results of MCU-ML and KUESL show that each method can work in the desired situation with acceptable performance. However, in some specific cases, the proposed methods will go fail because of the unique feature of the environment. The reasons for the failure cases have been discussed.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This study has presented a solution for kidnapping in SLAM, which is a new challenge in localization topic. With our analysis, we have found that two basic situations exist in kidnapping problem during SLAM process. One situation is that the robot is kidnapped to the explored area, while another is that the robot is kidnapped to a new area without mapping before. Based on these two unique situations, a basic solution including kidnapping detection and its recovery have been proposed.

A kidnapping detection framework called DKDR that can detect kidnapping and distinguish the type of kidnapping were proposed. DKDR framework comprises two processes embedded in SLAM algorithms with three metrics and their thresholds. The results of the simulations and experiments demonstrated the validity and feasibility of the proposed framework. The experimental mean execution time of the proposed method is $27\mu s$. The application of DKDR to different filter-based SLAM algorithms shows the simplicity and university of our framework. For solving the problem that the increased uncertainty of estimation according to the mapping area produces more false alarms in a large environment, the estimation uncertainty included a method based on DKDR called P-DKDR were introduced. The experiments results and simulations results show that P-DKDR has less false

alarms than DKDR in the large environment. Furthermore, P-DKDR has a little lower true positive rate and needs more experimental mean computation time ($47\mu s$) than DKDR.

Based on the different kidnapping types, a suitable kidnaping recovery method was carried out. If the robot is still in the explored area, MCL-MU will estimate the robot pose in the map built by SLAM. Since the ordinary MCL just works with a provided accurate map, the uncertainty of the map is not included in the algorithm. Furthermore, the robot motion is required during MCL which cannot apply to kidnapping such as stuck or slip on a spot. MCL-MU solved these problems with the modification of ordinary MCL. The experiment results show that validity and performance of MCL-MU. Moreover, the failure cases were also discussed with the analyzed reasons. About the kidnapping that the robot is moved out of the range of the sensor, KUESL which simultaneously processing SLAM and global localization to realize kidnapping recover have been described. An example of KUESL combing Gmapping and M-MCL-MU was introduced and utilized in the experiments. Two groups of experiments were conducted to verify the performance of KUESL with the two situations separately. The success rate with true E rate and true N rate are calculated based on the results of the experiment. The failure cases were analyzed in the discussion.

6.2 Future Work

With the analyzing the experiments results of the all proposed methods, several points could be the future work of the study.

1. The proposed DKDR framework can solve the problem of a short-time kidnapping events. If the kidnapping occurs over a long time, such as when the robot slips all the time in a specific area, the introduced method could fail. Thus, we need to improve on our method to solve such a problem.

2. To distinguish the similar scene in the environment, more sensors should be mounted on the robot to acquire more information of the environment. Based on multiple sensor reading, the difference between the similar scene should be recognized successfully.

3. For the problem that the dispersed particles cannot cover the actual robot pose well in MCL, a new method should be proposed. With re-spreading the particles in likely spots during each time step during MCL, the problem should be solved.

Once the existing problems can be solved with these future work, it can increase the robustness of the solution of kidnapping in SLAM to apply in the more complex environments.

Bibliography

[1] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[3] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, "Distributed search and rescue with robot and sensor teams," in *Field and Service Robotics*. Springer, 2003, pp. 529–538.

[4] S. Zorn, R. Rose, A. Goetz, and R. Weigel, "A novel technique for mobile phone localization for search and rescue applications," in *2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, 2010, pp. 1–4.

[5] A. Baggio, "Wireless sensor networks in precision agriculture," in *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005)*, 2005, pp. 1567–1576.

[6] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture," in *20th International Parallel and Distributed Processing Symposium*. IEEE, 2006, pp. 8–pp.

[7] W. Cheng, A. Y. Teymorian, L. Ma, X. Cheng, X. Lu, and Z. Lu, "Underwater localization in sparse 3d acoustic sensor networks," in *The 27th Conference on Computer Communications*. IEEE, 2008, pp. 236–240.

[8] H.-P. Tan, R. Diamant, W. K. Seah, and M. Waldmeyer, "A survey of techniques and challenges in underwater localization," *Ocean Engineering*, vol. 38, no. 14, pp. 1663–1676, 2011.

[9] 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. 15, 1996.

[10] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer Science & Business Media, 2008.

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

[12] J. Borenstein, L. Feng, and H. Everett, *Navigating mobile robots: Systems and techniques*. AK Peters, Ltd., 1996.

[13] E. Papadopoulos and M. Misailidis, "On differential drive robot odometry with application to path planning," in *2007 European Control Conference (ECC)*. IEEE, 2007, pp. 5492–5499.

[14] F. Chenavier and J. L. Crowley, "Position estimation for a mobile robot using vision and odometry," in *1992 IEEE International Conference on Robotics and Automation*. IEEE, 1992, pp. 2588–2593.

[15] J. Borenstein and L. Feng, "Measurement and correction of systematic odometry errors in mobile robots," *IEEE Transactions on robotics and automation*, vol. 12, no. 6, pp. 869–880, 1996.

[16] M. Reinstein, V. Kubelka, and K. Zimmermann, "Terrain adaptive odometry for mobile skid-steer robots," in *2013 ieee international conference on Robotics and automation (icra)*. IEEE, 2013, pp. 4706–4711.

[17] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 3, pp. 328–342, 1995.

[18] D. Titterton and J. L. Weston, *Strapdown inertial navigation technology*. IET, 2004, vol. 17.

[19] M. S. Grewal, L. R. Weill, and A. P. Andrews, *Global positioning systems, inertial navigation, and integration*. John Wiley & Sons, 2007.

[20] O. J. Woodman, "An introduction to inertial navigation," University of Cambridge, Computer Laboratory, Tech. Rep., 2007.

[21] M. G. Cavalcanti, J. W. Haller, and M. W. Vannier, "Three-dimensional computed tomography landmark measurement in craniofacial surgical planning: experimental validation in vitro," *Journal of oral and maxillofacial surgery*, vol. 57, no. 6, pp. 690–694, 1999.

[22] S. Bhattacharya, R. Murrieta-Cid, and S. Hutchinson, "Optimal paths for landmark-based navigation by differential-drive vehicles with field-of-view constraints," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 47–59, 2007.

[23] H. Hile, R. Vedantham, G. Cuellar, A. Liu, N. Gelfand, R. Grzeszczuk, and G. Borriello, "Landmark-based pedestrian navigation from collections of geotagged photos," in *Proceedings of the 7th international conference on mobile and ubiquitous multimedia*. ACM, 2008, pp. 145–152.

[24] K. O. Arras, J. A. Castellanos, M. Schilt, and R. Siegwart, "Feature-based multi-hypothesis localization and tracking using geometric constraints," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 41–53, 2003.

[25] P. Jensfelt, D. J. Austin, O. Wijk, and M. Andersson, "Feature based condensation for mobile robot localization," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 2531–2537.

[26] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[27] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.

[28] B.-S. Choi, J.-W. Lee, J.-J. Lee, and K.-T. Park, "A hierarchical algorithm for indoor mobile robot localization using rfid sensor fusion," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 6, pp. 2226–2235, 2011.

[29] G. Rigatos and S. Tzafestas, "Extended kalman filtering for fuzzy modelling and multi-sensor fusion," *Mathematical and computer modelling of dynamical systems*, vol. 13, no. 3, pp. 251–266, 2007.

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

[31] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 5, pp. 564–577, 2003.

[32] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[33] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[34] M. Csorba, "Simultaneous localisation and map building," Ph.D. dissertation, University of Oxford, 1997.

[35] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit *et al.*, "Fastslam: A factored solution to the simultaneous localization and mapping problem," in *Aaai/iaai*, 2002, pp. 593–598.

[36] M. Montemerlo and S. Thrun, "Fastslam 2.0," *FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, pp. 63–90, 2007.

[37] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on.* IEEE, 2005, pp. 2432–2437.

[38] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the ekf-slam algorithm," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.* IEEE, 2006, pp. 3562–3568.

[39] S. Ahn, M. Choi, J. Choi, and W. K. Chung, "Data association using visual object recognition for ekf-slam in home environment," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on.* IEEE, 2006, pp. 2588–2594.

[40] 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.

[41] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *IEEE Transactions on robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.

[42] Q. Pan, F. Yang, L. Ye, Y. Liang, and Y.-m. Cheng, "Survey of a kind of nonlinear filters-ukf," *Control and Decision*, vol. 20, no. 5, p. 481, 2005.

[43] J. Zhu, N. Zheng, Z. Yuan, Q. Zhang, X. Zhang, and Y. He, "A slam algorithm based on the central difference kalman filter," in *2009 IEEE Intelligent Vehicles Symposium*. IEEE, 2009, pp. 123–128.

[44] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "On the complexity and consistency of ukf-based slam," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 4401–4408.

[45] J. Andrade-Cetto, T. Vidal-Calleja, and A. Sanfeliu, "Unscented transformation of vehicle states in slam," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 323–328.

[46] F. A. Cheein, G. Steiner, G. P. Paina, and R. Carelli, "Optimized eif-slam algorithm for precision agriculture mapping based on stems detection," *Computers and electronics in agriculture*, vol. 78, no. 2, pp. 195–207, 2011.

[47] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based slam," *The International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, 2007.

[48] S. Huang, Z. Wang, and G. Dissanayake, "Sparse local submap joining filter for building large-scale maps," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1121–1130, 2008.

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

[50] L. Zhang, R. Zapata, and P. Lépinay, "Self-adaptive monte carlo localization for mobile robots using range sensors," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1541–1546.

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

[52] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

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

[54] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," *Robotics: Science and Systems VI*, vol. 2, 2010.

[55] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2011, pp. 155–160.

[56] D. M. Cole and P. M. Newman, "Using laser range data for 3d slam in outdoor environments," in *2006 IEEE International Conference on Robotics and Automation*. IEEE, 2006, pp. 1556–1563.

[57] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2006, pp. 363–370.

[58] E. Eade and T. Drummond, "Monocular slam as a graph of coalesced observations," in *IEEE 11th International Conference on Computer Vision*. IEEE, 2007, pp. 1–8.

[59] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[60] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[61] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based slam," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006.

[62] S. Thrun and Y. Liu, "Multi-robot slam with sparse extended information filers," *Robotics Research*, pp. 254–266, 2005.

[63] U. Frese, "Treemap: An o (log n) algorithm for indoor simultaneous localization and mapping," *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[64] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro, "Image-based monte carlo localisation with omnidirectional images," *Robotics and Autonomous Systems*, vol. 48, no. 1, pp. 17–30, 2004.

[65] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve." *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.

[66] E. Tsardoulias and L. Petrou, "Critical rays scan match slam," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 3-4, pp. 441–462, 2013.

[67] J. Nieto, T. Bailey, and E. Nebot, "Recursive scan-matching slam," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 39–49, 2007.

[68] D. Campbell and M. Whitty, "Metric-based detection of robot kidnapping," in *Mobile Robots (ECMR), 2013 European Conference on*. IEEE, 2013, pp. 192–197.

[69] D. Rodriguez-Losada and J. Minguez, "Improved data association for icp-based scan matching in noisy and dynamic environments," in *2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3161–3166.

[70] A. G. Ozkil, Z. Fan, J. Xiao, S. Dawids, J. K. Kristensen, and K. H. Christensen, "Mapping of multi-floor buildings: A barometric approach," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 847–852.

[71] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard, "Temporally scalable visual slam using a reduced pose graph," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 54–61.

[72] S. Lee, S. Lee, and S. Baek, "Vision-based kidnap recovery with slam for home cleaning robots," *Journal of Intelligent & Robotic Systems*, vol. 67, no. 1, pp. 7–24, 2012.

[73] J. Choi, M. Choi, and W. K. Chung, "Topological localization with kidnap recovery using sonar grid map matching in a home environment," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 3, pp. 366–374, 2012.

[74] K. L. Ho and P. Newman, "Loop closure detection in slam by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740–749, 2006.

[75] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[76] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.(IROS 2003)*, vol. 1. IEEE, 2003, pp. 206–211.

[77] K. Frenken *et al.*, "Entropy statistics and information theory," *Chapters*, 2007.

[78] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[79] R. H. Wong, J. Xiao, and S. L. Joseph, "A robust data association for simultaneous localization and mapping in dynamic environments," in *2010 IEEE International Conference on Information and Automation (ICIA)*, 2010, pp. 470–475.

[80] Y. Yi and Y. Huang, "Landmark sequence data association for simultaneous localization and mapping of robots," *Cybernetics and Information Technologies*, vol. 14, no. 3, pp. 86–95, 2014.

Appendix A

Motion Model

Odometry is commonly obtained by integrating wheel encoders information; most commercial robots make such integrated pose estimation available in periodic time intervals. Practical experience suggests that odometry, while still erroneous, is usually more accurate than velocity. Both suffer from drift and slippage, but velocity additionally suffers from the mismatch between the actual motion controllers and its (crude) mathematical model. However, odometry is only available in retrospect, after the robot moved. This poses no problem for filter algorithms, but makes this information unusable for accurate motion planning and control.

Technically, odometry are sensor measurements, not controls. To model odometry as measurements, the resulting Bayes filter would have to include the actual velocity as state variableswhich increases the dimension of the state space. To keep the state space small, it is therefore common to simply consider the odometry as if it was a control signal. The resulting model is at the core of many of todays best probabilistic robot systems.

Let us define the format of our control information. At time step t, the correct pose of the robot is modeled by the random variable x_t. The robot odometry estimates this pose; however, due to drift and slippage there is no fixed coordinate transformation between the coordinates used by the robots internal odometry and the physical world coordinates.

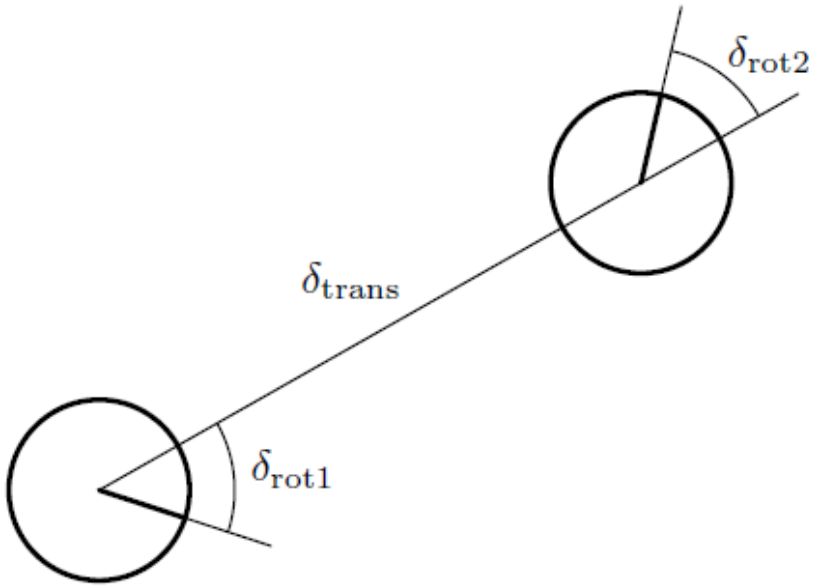


Figure A.1. Odometry model: The robot motion in the time interval $(t-1, t]$ is approximated by a rotation δ_{rot1}, followed by a translation δ_{trans} and a second rotationδ_{rot2}. The turns and translation are noisy.

The odometry model uses the relative information of the robots internal odometry. More specifically, In the time interval $(t-1, t]$, the robot advances from a pose x_{t-1} to pose x_t. The odometry reports back to us a related advance from $\bar{x}_{t-1} = (\bar{x}, \bar{y}, \bar{\theta})$ to $\bar{x}_t = (\bar{x}', \bar{y}', \bar{\theta}')$. Here the bar indicates that these are odometry measurements, embedded in a robot-internal coordinate whose relation to the global world coordinates is unknown. The key insight for utilizing this information in state estimation is that the relative difference between $\bar{x}_{t-1}$ and $\bar{x}_t$, under an appropriate definition of the term difference, is a good estimator for the difference of the true poses x_{t-1} and x_t. The motion information u_t is, thus, given by the pair

$$u_t = (\bar{x}_{t-1}, \bar{x}_t)^T \tag{A.1}$$

To extract relative odometry, u_t is transformed into a sequence of three steps: a rotation, followed by a straight line motion and another rotation, as shown in Figure A.1. The initial

Algorithm 3: sample odometry motion model(X_{t-1},u_t,z_t)

1 $\delta_{rot1} = atan2(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$

2 $\delta_{trans} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$

3 $\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$

4 $\hat{\delta}_{rot1} = atan2(y' - y, x' - x) - \theta$

5 $\hat{\delta}_{trans} = \sqrt{(x - x')^2 + (y - y')^2}$

6 $\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$

7 $p_1 = prob(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1\delta_{rot1} + \alpha_2\delta_{trans})$

8 $p_2 = prob(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3\delta_{trans} + alpha_4(\delta_{rot1} + \delta_{rot2}))$

9 $p_3 = prob(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1\delta_{rot2} + \alpha_2\delta_{trans})$

10 **return** p_1, p_2, p_3

Table A.1. Algorithm for sampling from $P(x_t|u_t, x_{t-1})$ based on odometry information.

turn is called δ_{rot1}, the translation δ_{trans}, and the second rotation δ_{rot2}. Each pair of positions $(\bar{s}, \bar{s}')$ has a unique parameter vector $(\delta_{rot1}, \delta_{trans}, \delta_{rot2})^T$, and these parameters are sufficient to reconstruct the relative motion between $\bar{s}$ and $\bar{s}'$. Thus, $\delta_{rot1}, \delta_{trans}, \delta_{rot2}$ is a sufficient statistics of the relative motion encoded by the odometry. The motion model assumes that these three parameters are corrupted by independent noise.

Table A.1 depicts the algorithm for computing $P(x_t|u_t, x_{t-1})$ from odometry. This algorithm accepts as an input an initial pose x_{t-1}, a pair of poses $u_t = (\bar{x}_{t-1}, \bar{x}_t)^T$ obtained from the robots odometry, and a hypothesized final pose x_t. It outputs the numerical probability $P(x_t|u_t, x_{t-1})$ with p_1, p_2, p_3.

Lines 2 to 4 recover relative motion parameters $\delta_{rot1}, \delta_{trans}, \delta_{rot2}$ from the odometry readings. They implement an inverse motion model. The corresponding relative motion parameters $\hat{\delta}_{rot1}, \hat{\delta}_{trans}, \hat{\delta}_{rot2}$ for the given poses x_{t-1} and x_t are calculated in Lines 5 through 7 of this algorithm. Lines 8 to 10 compute the error probabilities for the individual motion parameters. As above, the function $prob(a, b^2)$ implements an error distribution over a with zero mean a and variance b^2. Here the implementer must observe that all angular

differences must lie in $[-\pi, \pi]$. Hence the outcome of $\delta_{rot2} - \hat{\delta}_{rot2}$ has to be truncated correspondingly a common error that tends to yield occasional divergence of software based on this model. Finally, Line 11 returns the combined error probability, obtained by multiplying the individual error probabilities p1, p2, and p3. This last step assumes independence between the different error sources. The variables α_1 through α_4 are robot-specific parameters that specify the noise in robot motion.

Appendix B

Measurement Model

Measurement models comprise the second domain-specific model in probabilistic robotics, next to motion models. Measurement models describe the formation process by which sensor measurements are generated in the physical world. Todays robots use a variety of different sensor modalities, such as tactile sensors, range sensors, or cameras. The specifics of the model depends on the sensor: Imaging sensors are best modeled by projective geometry, whereas sonar sensors are best modeled by describing the sound wave and its reflection on surfaces in the environment.

Probabilistic robotics explicitly models the noise in sensor measurements. Such models account for the inherent uncertainty in the robots sensors. Formally, the measurement model is defined as a conditional probability distribution $P(z_t|x_t, m)$, where x_t is the robot pose, z_t is the measurement at time step t, and m is the map of the environment. Instead the basic principle can be applied to any kind of sensor, such as a camera or a bar-code operated landmark detector.

The model incorporates four types of measurement errors, all of which are essential to making this model work: small measurement noise, errors due to unexpected objects, errors due to failures to detect objects, and random unexplained noise. The desired model $P(z_t|x_t, m)$ is therefore a mixture of four densities, each of which corresponds to a particular type of error:

1. Correct range with local measurement noise. In an ideal world, a range finder would always measure the correct range to the nearest object in its measurement field. Let us use z_t^{k*} to denote the true range of the object measured by z_t^k. In location-based maps, the range z_t^{k*} can be determined using ray casting; in feature-based maps, it is usually obtained by searching for the closest feature within a measurement cone. However, even if the sensor correctly measures the range to the nearest object, the value it returns is subject to error. This error arises from the limited resolution of range sensors, atmospheric effect on the measurement signal, and so on. This noise $p_{hit}(z_t^k|x_t, m)$ is usually modeled by a narrow Gaussian with mean z_t^k and standard deviation σ_{hit}.

2. Unexpected objects. Environments of mobile robots are dynamic, whereas maps m are static. As a result, objects not contained in the map can cause range finders to produce surprisingly short rangesat least when compared to the map. A typical example of moving objects are people that share the operational space of the robot. One way to deal with such objects is to treat them as part of the state vector and estimate their location; another, much simpler approach, is to treat them as sensor noise. Treated as sensor noise, unmodeled objects have the property that they cause ranges to be shorter than z_t^k, not longer. More generally, the likelihood of sensing unexpected objects decreases with range. To see, imagine there are two people that independently and with the same, fixed likelihood show up in the perceptual field of a proximity sensor. One persons range is z_1, and the second persons range is z_2. Let us further assume that $z_1 < z_2$, without loss of generality. Then we are more likely to measure z_1 than z_2. Whenever the first person is present, our sensor measures z_1. However, for it to measure z_2, the second person must be present and the first must be absent. Mathematically, the probability of range measurements $p_{short}(z_t^k|x_t, m)$ in such situations is described by an exponential distribution.

3. Failures. Sometimes, obstacles are missed altogether. For example, this happens frequently with sonar sensors when measuring a surface at a steep angle. Failures also

occur with laser range finders when sensing black, light-absorbing objects, or when measuring objects in bright light. A typical result of sensor failures are max-range measurements: the sensor returns its maximum allowable value . Since such events are quite frequent, it is necessary to explicitly model max-range measurements in the measurement model. A point-mass distribution is applied as the noise $p_{max}(z_t^k|x_t, m)$.

4. Random measurements. Finally, range finders occasionally produce entirely unexplained measurements. For example, sonars often generate phantom readings when they bounce off walls, or when they are subject to cross-talk between different sensors. To keep things simple, such measurements probability $p_{rand}(z_t^k|x_t, m)$ will be modeled using a uniform distribution spread over the entire sensor measurement range $[0, z_{max}]$.

These four different distributions are now mixed by a weighted average, defined by the parameters z_{hit}, z_{short}, z_{max}, and z_{rand} with $z_{hit} + z_{short} + z_{max} + z_{rand} = 1$. With the equation (B.1), the basic characteristics of all four basic models are combined.

$$p(z_t^k|x_t, m) = \begin{pmatrix} z_{hit} & z_{short} & z_{max} & z_{rand} \end{pmatrix} \cdot \begin{pmatrix} p_{hit}(z_t^k|x_t, m) \\ p_{short}(z_t^k|x_t, m) \\ p_{max}(z_t^k|x_t, m) \\ p_{rand}(z_t^k|x_t, m) \end{pmatrix} \tag{B.1}$$

Published Papers During Doctoral Course

Journal Papers:

1. **Yang Tian** and Shugen Ma, Kidnapping Detection and Recognition in Previous Unknown Environment, ***Journal of Sensors***, vol.2017, pp. 1-15, 2017.

2. **Yang Tian** and Shugen Ma, Probabilistic Double Guarantee Kidnapping Detection in SLAM, ***Robotics and Biomimetics***, vol.3, no. 1, pp. 1-7, 2016.

International Conference Papers:

1. **Yang Tian**, Takahiro Matsuno and Shugen Ma, Development of Remote Robot Control System for Snake-like Robot based on SSH Protocol and iOS System, In *Proc. of the 2016 IEEE International Conference on Robotics and Biomimetics (**ROBIO2016**)*, Qingdao, China, pp. 100-105, Dec. 2016.

2. **Yang Tian**, Victor Gomez and Shugen Ma, Influence of Two SLAM Algorithms using Serpentine Locomotion in a Featureless Environment, In *Proc. of the 2015 IEEE International Conference on Robotics and Biomimetics (**ROBIO2015**)*, Zhuhai, China, pp. 182-187, Dec. 2015.

3. **Yang Tian** and Shugen Ma, A Double Guarantee Kidnapping Detection in Simultaneous Localization and Mapping, In *Proc. of the 2014 IEEE International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (**MFI2014**)*, Beijing, China, pp. 1-6, Sep. 2014.