Research Article

Kidnapping Detection and Recognition in Previous Unknown Environment

Yang Tian and Shugen Ma

Department of Robotics, Ritsumeikan University, Shiga 525-8577, Japan

Correspondence should be addressed to Yang Tian; piaocaoty@gmail.com

Received 5 April 2017; Revised 23 June 2017; Accepted 3 July 2017; Published 22 August 2017

Academic Editor: Hana Vaisocherova

Copyright © 2017 Yang Tian and Shugen Ma. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

An unaware event referred to as kidnapping makes the estimation result of localization incorrect. In a previous unknown environment, incorrect localization result causes incorrect mapping result in Simultaneous Localization and Mapping (SLAM) by kidnapping. In this situation, the explored area and unexplored area are divided to make the kidnapping recovery difficult. To provide sufficient information on kidnapping, a framework to judge whether kidnapping has occurred and to identify the type of kidnapping with filter-based SLAM is proposed. The framework is called double kidnapping detection and recognition (DKDR) by performing two checks before and after the “update” process with different metrics 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. Two classical filter-based SLAM algorithms, Extend Kalman Filter (EKF) SLAM and Particle Filter (PF) SLAM, are modified to show that DKDR can be simply and widely applied in existing filter-based SLAM algorithms. Furthermore, a technique to determine the adapted thresholds of metrics in real time without previous data is presented. Both simulated and experimental results demonstrate the validity and accuracy of the proposed method.

1. Introduction

Localization, which provides the estimated pose (position and orientation) of a target based on the world coordinates, is a fundamental capability in various fields. Over the past few decades, several advancements in localization have been realized [1, 2].

In a previous unknown environment, Simultaneous Localization and Mapping (SLAM) is necessary to determine the pose of a target [3–7]. In SLAM, the target incrementally builds a consistent map of the environment while simultaneously determining its pose within this map. Filter-based SLAM [5, 7–9] processes the information from proprioceptive and exteroceptive sensors with filters, such as Extend Kalman Filter (EKF) and Particle Filter (PF), allowing an optimized estimated result to be obtained.

Kidnapping is a localization problem that occurs when an unexpected movement happens to the target due to interaction with its surroundings. It causes incorrect pose estimation of the target. Several methods have been proposed

to solve this problem in previous known-map situation, for example, Monte Carlo Localization [10–12] that performs global localization [13] regularly. In [14], the authors assume that the robot is kidnapped to the explored area during SLAM and is relocalized successfully. However, these methods cannot be directly applied in previous unknown environments since SLAM divides the environment into explored area and unexplored area, and the target cannot judge whether it is kidnapped to the explored area before a correct judgement. The detailed reason and analysis are described in Section 2.

In this paper, we firstly describe why detecting and recognizing kidnapping are necessary during the SLAM process. To the authors' knowledge, this is the first study to give an analysis of different situations about kidnapping in SLAM. Second, we describe a property of the filter-based SLAM that corrects the entire map of the environment with current observations after the “update” process. Based on this property, a method 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). Third, a new classification for kidnapping is proposed to provide more information on 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 in EKF-SLAM and PF-SLAM with a mobile robot.

This paper is organized as follows. Section 2 describes an analysis of kidnapping in SLAM which indicates the motivation of our research. In Section 3, we introduce the overall design of the DKDR. In Section 4, we describe three metrics that are used in the new method along with their thresholds. In Section 5, we report on the addition of the proposed method to EKF-SLAM while the simulation was running in MATLAB. We also present the results of experiments using Gmapping [15] with DKDR in a realistic environment. Next, we discuss the Receiver Operating Characteristic (ROC) to demonstrate the validity and accuracy of the method. The discussion is given in Section 6, and the conclusion is presented in Section 7.

2. Analysis of Kidnapping in SLAM

Previous studies about kidnapping mainly concentrated on correcting the target's state (position and orientation) after the kidnapping with previous known-map situation, which is called kidnapping recovery. With a previous known map, existing kidnapping recovery methods such as Monte Carlo Localization (MCL) and scan-to-map matching [12, 16] can effectively solve the problem. However, the situation of kidnapping in SLAM with the previous unknown map is different. For providing an intuitive explanation, a mobile robot is utilized as the target. In the previous unknown environment, the robot needs to explore the unknown area with SLAM, as shown in Figure 1. Therein, white circles represent undetected features in the environment. An exteroceptive sensor with a limited range represented as a red dashed circle is carried by a mobile robot shown as a red triangle. During the SLAM, the robot collects useful information (features' positions) about its surroundings while it is moving 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. While the robot is performing SLAM, the constructed mapping areas can be classified as explored areas (detected features) or unexplored areas (undetected features). It is necessary to judge the kidnapping type to decide what kidnapping recovery strategy should be employed. For example, if the robot was kidnapped into an explored area as shown in Figure 1(c), existing kidnapping recovery methods can be applied directly to correct the robot's pose. However, if the robot was kidnapped into an unexplored area as shown in Figure 1(d), directly utilizing existing methods may result in a wrong estimated pose. Although there is no known map actually around it, the robot still tries to estimate its pose with the already collected information. This problem is easily

FIGURE 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.

encountered when the robot is kidnapped to a similar scene in an explored area after the kidnapping. Kidnapping detection and recognition should be the first step of the overall solution to kidnapping in SLAM to provide basic information on the kidnapping recovery process.

In previous unknown-map situations, kidnapping causes not only an incorrect estimation of the robot's pose but also a deformation of the mapping result. In SLAM algorithms [17, 18] without filters, the incorrect information is directly added to the explored map after kidnapping, which affects the accuracy 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, making the explored map information difficult to recover after kidnapping. To prevent this scenario, a more strict

FIGURE 2: Overall DKDR workflow.

check about kidnapping is required. With timely detection, the correct information about the explored area will not be deformed by kidnapping, and some of this information can then be utilized for recovery from the kidnapping with sufficient conditions.

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 [19], two main types of kidnapping (types “A” and “B”) are proposed in this paper. Type “A” kidnapping occurs when the robot does not correctly estimate its pose after its actual pose 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 pose 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 method may cause a fault if it is applied directly. Similar situations such as entering an elevator, being pushed away by other robots, or falling down the stairs should also be recognized as type “A” kidnapping. Type “B” kidnapping occurs when the robot does not actually change its pose; the predicted pose however is significantly changed. For example, the robot is stuck in an area or is moving in a slippery area. In this case, since the robot is just kidnapped in an explored area, the ordinary recovery methods could be applied directly. In contrast to previous research, we do not think that the wake-up problem should be included in kidnapping. Both type “A” and “B” kidnapping events are divided into two subtypes 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 design a suitable recovery algorithm, such as the range of scan-map matching.

Existing methods for kidnapping detection can be divided into two types, physical and mathematical methods. Physical methods use specific sensors (e.g., barometer [20], accelerometer [21], and switch [22]) to measure whether or not kidnapping has occurred; however, each sensor can detect only a specific type of kidnapping (e.g., the robot is lifted up or stuck in an area). On the other hand, mathematical methods utilize sensors such as wheel encoder and laser range finder (LRF) to observe more types of kidnapping. 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 [14], 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 [19] needs two independently operating sensor resources along with previous test data. Its design of the kidnapping classification does not consider the previous unknown-map situation.

3. Overall DKDR Design

3.1. DKDR Workflow. As shown in Figure 2, 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, 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.

Before introducing DKDR framework, we briefly describe three ordinary SLAM processes (predict, observe, and update). The robot’s state is described by the vector $X_r = [x_r, y_r, \phi_r]^T$, in which (x_r, y_r) represents the position and ϕ_r represents the orientation of a frame attached to the robot. The state of features is denoted by $X_m = [X_{m1}^T, X_{m2}^T, \dots]^T$, in which $X_{mi} = [x_{mi}, y_{mi}]^T$ represents the position of the feature i in the global coordinates. X_{mi} is given by

$$X_{mi} = \begin{bmatrix} \cos \phi_r & -\sin \phi_r \\ \sin \phi_r & \cos \phi_r \end{bmatrix} \begin{bmatrix} {}^L x_i \\ {}^L y_i \end{bmatrix} + \begin{bmatrix} x_r \\ y_r \end{bmatrix}, \quad (1)$$

where $({}^L x_i, {}^L y_i)$ represents the position of feature i referred to the local coordinates frame mounted on the robot. Therefore, the state vector is $X = [X_r^T, X_m^T]^T$ which contains both the robot state X_r and the feature states X_m .

In the “predict” process, the predicted state $X(k+1 | k) = [X_r^T(k+1 | k), X_m^T(k+1 | k)]^T$ at time step k is given by

$$X(k+1 | k) = F(X(k | k), u(k)) + w(k) \\ = \begin{bmatrix} f(X_r(k | k), u(k)) \\ X_m(k | k) \end{bmatrix} + \begin{bmatrix} w_r(k) \\ O \end{bmatrix}, \quad (2)$$

where $(X_r(k | k), X_m(k | k))$ is the state at the time step k , $u(k)$ indicates the control measurement at time step k , $w_r(k)$ is the process noise assumed to be white Gaussian with zero mean and covariance Q , and the function f depends on the motion model.

Prediction of the state covariance matrix $P(k+1 | k)$ is given by

$$P(k+1 | k) = \nabla F_X P(k | k) \nabla F_X^T + Q, \quad (3)$$

where ∇F_X is the Jacobian of F with respect to X evaluated at $X(k | k)$ and $P(k | k)$ denotes the state covariance matrix at time step k .

The observations $Z(k+1 | k)$ that are obtained from the state $X(k+1 | k)$ at the time step k in the “observe” process are given by

$$Z(k+1 | k) = H(X(k+1 | k)) + v(k), \quad (4)$$

where H defines the nonlinear coordinates transformation from the state to the observation $Z(k+1 | k)$. The noise $v(k)$ is assumed to be white Gaussian with zero mean, and $Z(k+1 | k+1)$ is measured from the actual environment and its covariance matrix is denoted by R . To compare observations in sequential time steps, kidnapping can be detected and distinguished in the prior-check process.

In the “update” process, the state X and the associated covariance matrix P are updated by a filter such as EKF using the observation Z , which is given by

$$X(k+1 | k+1) \\ = X(k+1 | k) \\ + K(k+1) [Z(k+1 | k+1) - H(X(k+1 | k))], \quad (5)$$

$$P(k+1 | k+1) \\ = P(k+1 | k) - K(k+1) S(k+1) K(k+1)^T,$$

where

$$K(k+1) = P(k+1 | k) \nabla H_X^T S(k+1)^{-1}, \quad (6) \\ S(k+1) = \nabla H_X P(k+1 | k) \nabla H_X^T + R,$$

and ∇H_X is the Jacobian of H with respect to X evaluated at $X(k+1 | k)$.

If kidnapping occurred, the difference between observations $Z(k+1 | k)$ and $Z(k+1 | k+1)$ would be enlarged. Thus, the value of the component in (5), $Z(k+1 | k+1) - H(X(k+1 | k))$, would be significantly increased. $K(k+1)$ includes the state covariance matrix $P(k+1)$ multiplying

$Z(k+1 | k+1) - H(X(k+1 | k))$, while $P(k+1)$ 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 X_m would be changed obviously after the “update” process. By comparing the differences between $X_m(k | k)$ and $X_m(k+1 | k+1)$, 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.

3.2. Prior-Check and Posterior-Check Processes. Two new processes, the prior-check and the posterior-check processes, are introduced in 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.

To evaluate whether the robot has moved to the designated position, the metric Q_p and its thresholds are needed. A predicted robot state $X_r(k+1 | k)$ is generated as an output in the “predict” process. The predicted observation $Z(k+1 | k)$ can thus be calculated using (4). When the robot actually turns to the predicted state, the actual observation $Z(k+1 | k+1)$ should be similar to the predicted observation $Z(k+1 | k)$. 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. The characterization of these kidnapping types is listed in Table 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. T_1 , T_2 , T_3 , and T_4 denote suitable thresholds. 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 due to slippage or another external force.

FIGURE 3: Workflows of prior-check and posterior-check processes.

TABLE 1: Different types of kidnapping.

Type	Condition 1	Condition 2
Nonkidnapping	$\overline{AP} \leq T_1$	$\overline{CL} > T_3$
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	$\overline{AP} > T_2$	$\overline{CL} \leq T_4$

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 in 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 pretests. If kidnapping cannot be detected with these two metrics, another metric Q_s is required for the posttest 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

$$CR = PrR \vee PoR,$$

$$PrR = \begin{cases} 1, & \text{if report kidnapping,} \\ 0, & \text{otherwise,} \end{cases} \quad (7)$$

$$PoR = \begin{cases} 1, & \text{if report kidnapping,} \\ 0, & \text{otherwise,} \end{cases}$$

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 PoR. 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 a new SLAM process. Since this study is focused on the detection and classification of kidnapping, recovery methods are not discussed in the paper. 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 alarms 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 [23, 24]. 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.

4. 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.

4.1. Metrics Q_p , Q_o , and Q_s . Q_p and Q_o compare the difference in observations directly. Without coordinate transformation, these metrics need less computation. For getting an efficient calculation, the metrics are calculated based on Euclidean distance, which is more efficient than other kinds of distances, such as Mahalanobis distance.

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

$$Q_p(k) = \sqrt{\frac{1}{N} \sum_{i=1}^N \|Z_i(k+1 | k+1) - Z_i(k+1 | k)\|^2}, \quad (8)$$

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

$$Q_o(k) = \sqrt{\frac{1}{N} \sum_{i=1}^N \|Z_i(k+1 | k+1) - Z_i(k | k)\|^2} \quad (9)$$

and Q_s at time step k is given by

$$Q_s(k) = \sqrt{\frac{1}{M} \sum_{i=1}^M \|X_{mi}(k+1 | k+1) - X_{mi}(k | k)\|^2}, \quad (10)$$

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

4.2. Thresholds. The accuracy of detection and classification is related to the metrics' thresholds, and a suitable method

FIGURE 4: Profile of half-normal distribution.

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 4). Because the standard deviation can reflect the percentage of events, as shown in Figure 4, the thresholds can be determined by the percentage of kidnapping events. 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

$$\sigma_p(k) = \sqrt{\frac{1}{k} \sum_{i=1}^k Q_p(i)^2}. \quad (11)$$

The threshold of Q_o is also T_{p1} . The threshold of the metric Q_s can be determined by its standard deviation σ_s shown in

$$\sigma_s(k) = \sqrt{\frac{1}{k} \sum_{i=1}^k Q_s(i)^2}. \quad (12)$$

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 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. Please

TABLE 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	$(T_s, +\infty)$	$(T_s, +\infty)$	$(T_s, +\infty)$	$(T_s, +\infty)$

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

TABLE 3: Simulation conditions.

Condition	Value
Robot speed	0.3 m/s
Control cycle	0.2 s
Observation cycle	0.2 s
Max. range of observation	3 m
SD of speed noise	0.09 m/s
SD of orientation noise	9°
SD of observation position noise	0.01 m
SD of observation angle noise	1°

*SD: standard deviation.

note that Table 1 only describes the conception of kidnapping type which is different from Table 2.

5. Simulations and Experiments

Simulations and experiments 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 the following conditions:

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

We aim to show the correctness of DKDR in the simulations, including the response of DKDR and the phenomenon after kidnapping without detection. In addition to the simulations, experiments were conducted to demonstrate the performance of DKDR under more realistic conditions (unknown data association). To show the universality of DKDR, different filter-based SLAM algorithms were separately applied in simulations and experiments.

Simulations were executed using MATLAB on a personal computer (CPU: 3.40 GHz Intel Core i5, memory: 8 GB DDR3). The source code was based on the EKF-SLAM algorithm in the SLAM package of Bailey and Durrant-Whyte [3]. We modified and implemented our method into this package. The simulation conditions are shown in Table 3. The shape of the robot is shown in Figure 5.

The map and different situations are depicted in Figure 6. As indicated in Figure 6(a), “RPF” denotes the real position

FIGURE 5: Dimensions of the robot used in simulations.

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 6(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 the distance between them becomes larger as time step passes. 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 6(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 6(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 7. To judge whether kidnapping had actually occurred, we calculated real data without uncertainty at each time step. The distance between the predicted and actual positions of the robot is shown in Figure 7(a). The difference between the angles of the predicted and actual orientations of the robot is shown in Figure 7(b). From Figure 7, we could easily determine how the differences in the robot’s position and orientation

FIGURE 6: Results of the simulation in different situations. (a) Simulation map. (b) Normal SLAM process. (c) Kidnapping result without detection. (d) Kidnapping result with detection.

changed due to kidnapping. The values of Q_p for kidnapping and nonkidnapping situations are shown in Figure 7(c). In the nonkidnapping situation, Q_p is lower than the first threshold T_{p1} . When type A.2 kidnapping occurs in the 501st 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 6(d). Q_s is lower than T_s before kidnapping. Q_o is not shown here since it is only calculated once after Q_p beyond its threshold, which has been denoted in Figure 3. Since Q_o is only calculated after

kidnapping is detected, shown in Figure 3, it is not shown in Figure 7.

We performed several simulations for kidnapping and nonkidnapping 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 4. About the report of DKDR, the

FIGURE 7: Metrics used to detect kidnapping. (a) Distance. (b) Angle. (c) Q_p . (d) Q_s .

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 nonkidnapping time steps out of the total number of actual nonkidnapping 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.

The simulated results for kidnapping type classification are shown in Table 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

TABLE 4: Operating characteristics of DKDR.

True positive rate	False positive rate
0.9990	0.0228

events of a certain type out of the total number of actual other types of kidnapping.

Several experiments were performed to verify the performance of DKDR in a real static environment. The proposed method was applied to Gmapping (FastSLAM 2.0) using Robot Operating System (ROS) [25] in an indoor environment. Gmapping is one of the most common SLAM algorithms based on PF in ROS. For demonstrating our method's

TABLE 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

FIGURE 8: Robot platform.

TABLE 6: Simulation conditions.

Condition	Value
Average robot speed	0.3 m/s
Control cycle	0.2 sec
Filter cycle	0.3 m or 5°
Max. range of sensor	4 m
Scan angle of sensor	180°
SD of speed noise	0.05 m/s
SD of orientation noise	3°
SD of observation position noise	0.02 m
SD of observation angle noise	0.36°

* SD: standard deviation.

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 a 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 [26], JPDA (Joint Probabilistic Data Association) [27], and LSDA (Landmark Sequence Data Association) [28]. 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 the experiments, a 2D LRF and a laptop (CPU: 2.20 GHz Intel Core i3, memory:

4 GB DDR2) were mounted on the robot platform, as shown in Figure 8. Some key parameters used in the experiments are shown in Table 6. 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.

Different experiments were designed to demonstrate the performance of DKDR in real static indoor environments. Since kidnapping can be efficiently detected and recovered from by using GPS sensors in outdoor environments, outdoor experiments were not considered in this paper. In the first experiment, we tested the response of DKDR in a nonkidnapping situation since any detection system has some probability of false alarm.

The map of the environment treated as a ground truth and an example of a nonkidnapping situation with Gmapping in this environment are shown in Figure 9. Figure 9(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 a red line and blue and green dots, respectively. This trajectory is for a nonkidnapping situation. The trajectory only shows the approximate path followed by the robot. The results obtained using Gmapping in the nonkidnapping situation are shown in Figure 9(b); Gmapping 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 nonkidnapping 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 10. Figure 10(a) shows the starting and ending positions of the kidnapping event. Additionally, the trajectories for the kidnapping and nonkidnapping 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 11 m. Subsequently, the robot moves to the end point (green point) along the red trajectory. During this process, the kidnapping also does not happen. Without DKDR, the information of the mapping result is directly added to the original map after the kidnapping event (Figure 10(b)). After kidnapping, the mapping result created by Gmapping without DKDR does not match the ground truth within a reasonable error. Figure 10(c) shows the result with DKDR. While

FIGURE 9: The map and the mapping result of a nonkidnapping situation. (a) Map and trajectory. (b) Mapping result.

FIGURE 10: The example of type A.2 kidnapping. (a) Map and trajectory. (b) Mapping result without DKDR. (c) Mapping result with DKDR.

kidnapping happens, DKDR detects this abnormal event and keeps the original information from being deformed by the incorrect information. By comparing the areas of the mapping results before kidnapping in Figures 10(a) and 10(b), it is indicated that the original information was slightly deformed by kidnapping.

The data from the experiment described above are shown in Figure 11. 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 12. The map shown in Figure 12(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 11 m. The robot then moves along the red trajectory until the end point. Figure 12(b) shows the mapping result without DKDR. Since

FIGURE 11: The response of metrics. (a) Q_p . (b) Q_s .

FIGURE 12: The example of type B.2 kidnapping. (a) Map and trajectory. (b) Mapping result without DKDR. (c) Mapping result with DKDR.

FIGURE 13: The response of metrics. (a) Q_p . (b) Q_s .

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 12(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. Figure 13 shows the responses of Q_p and Q_s . In the nonkidnapping 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 as 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 13(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 13 and are further discussed in the following section.

The experimental results for detecting kidnapping and distinguishing different types of kidnapping are shown in Tables 7 and 8, respectively. If the robot is moved more

TABLE 7: Operating characteristics of DKDR.

True positive rate	False positive rate
0.9810	0.0728

than 10 m, it is recognized as type A.2 kidnapping. Type B.2 kidnapping is defined as kidnapping where the robot stops at some specific point until its odometry reading exceeds 10 m. The ranges of type A.1 and B.1 are set at 2 m.

6. Discussion

The ability and performance of DKDR are demonstrated in Section 4. 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 experiments is clearly not the same. Several factors affect the performance. First, the noise of the entire system cannot be guaranteed to follow the Gaussian distribution in a real environment, which affects the accuracy of the SLAM algorithm and eventually the DKDR classification performance. Second, 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

TABLE 8: 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

cannot detect kidnapping. Third, if the robot is moved into a place that is similar to the environment around the kidnapping position, such as in a corridor without any other apparent features, since it is difficult for the sensor to distinguish the differences between the environments before and after kidnapping, DKDR also cannot detect kidnapping in this case.

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 increases. Although the robot is stuck until the odometry data gets over 10 m, the filter was executed many times. The data of the example is shown in Figure 13(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 did not change the wheel odometry data directly in the simulation, because this change cannot 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. 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.

7. Conclusion

In this paper, we have proposed a double-checking framework for kidnapping detection and recognition within the

filter-based SLAM. Our kidnapping detection framework comprises two processes embedded in SLAM algorithms. With three metrics and their thresholds, judging whether kidnapping has occurred and what type of kidnapping occurred is easy. Using the proposed framework, different types of kidnapping can be detected and recognized. The results of the simulations and experiments demonstrate the validity and feasibility of the proposed framework. The experimental execution time of the proposed method is only 27 μ s. The application of DKDR to different filter-based SLAM algorithms shows the simplicity and universality of our framework. The proposed method can solve the problem of 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 method introduced in this paper could fail. Thus, we need to improve our method to solve such a problem.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

- [1] J. Borenstein, H. R. 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.
- [2] S. Bruno and O. Khatib, *Springer handbook of robotics*, Springer Science Business Media, German, 2008.
- [3] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.
- [4] M. Csorba, *Simultaneous localisation and map building [Ph.D. thesis]*, University of Oxford, 1997.
- [5] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 2432–2437, IEEE, Barcelona, Spain, Spain, April 2005.
- [6] M. Montemerlo and T. Sebastian, "Fastslam 2.0," *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pp. 63–90, 2007.
- [7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "A factored solution to the simultaneous localization and mapping problem," *Aaai/Iaai*, pp. 593–598, 2002.
- [8] S. Ahn, M. Choi, J. Choi, and W. K. Chung, "Data association using visual object recognition for EKF-SLAM in home environment," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006*, pp. 2588–2594, chn, October 2006.

- [9] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, pp. 3562–3568, Beijing, China, October 2006.
- [10] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '99)*, pp. 1322–1328, IEEE, Detroit, Mich, USA, May 1999.
- [11] 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.
- [12] L. Zhang, R. Zapata, and P. Lépinay, "Self-adaptive monte carlo localization for mobile robots using range sensors," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '9)*, pp. 1541–1546, St. Louis, Mo, USA, October 2009.
- [13] A. Milstein, J. N. Sánchez, and E. T. Williamson, "Robust global localization using clustered particle filtering," *AAAI/IAAI*, pp. 581–586, 2002.
- [14] 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.
- [15] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.
- [16] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based SLAM," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.
- [17] J. Nieto, T. Bailey, and E. Nebot, "Recursive scan-matching SLAM," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 39–49, 2007.
- [18] E. Tsardoulas and L. Petrou, "Critical rays scan match SLAM," *Journal of Intelligent and Robotic Systems*, vol. 72, no. 3–4, pp. 441–462, 2013.
- [19] D. Campbell and M. Whitty, "Metric-based detection of robot kidnapping," in *Proceedings of the 2013 6th European Conference on Mobile Robots, ECMR 2013*, pp. 192–197, esp, September 2013.
- [20] 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 *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems: Celebrating 50 Years of Robotics, (IROS '11)*, pp. 847–852, San Francisco, Calif, USA, September 2011.
- [21] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard, "Temporally scalable visual SLAM using a reduced pose graph," in *Proceedings of the 2013 IEEE International Conference on Robotics and Automation, ICRA 2013*, pp. 54–61, deu, May 2013.
- [22] S. Lee, S. Lee, and S. Baek, "Vision-based kidnap recovery with SLAM for home cleaning robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 67, no. 1, pp. 7–24, 2012.
- [23] 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.
- [24] 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.
- [25] M. Quigley, K. Conley, B. Gerkey et al., "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.
- [26] D. Rodriguez-Losada and J. Minguez, "Improved data association for ICP-based scan matching in noisy and dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation, (ICRA '07)*, pp. 3161–3166, Roma, Italy, April 2007.
- [27] R. H. Wong, J. Xiao, and S. L. Joseph, "A robust data association for simultaneous localization and mapping in dynamic environments," in *Proceedings of the IEEE International Conference on Information and Automation (ICIA '10)*, pp. 470–475, Harbin, China, June 2010.
- [28] 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.