The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Self-Adaptive Monte Carlo Localization for Mobile Robots Using Range Sensors

Lei Zhang, René Zapata and Pascal Lépinay
Laboratoire d'Informatique, de Robotique et de
Microélectronique de Montpellier (LIRMM)
Université Montpellier II
161 rue Ada, 34392 Montpellier Cedex 5, France
{lei.zhang,rene.zapata,pascal.lepinay}@lirmm.fr

Abstract— **In order to achieve the autonomy of mobile robots, effective localization is a necessary prerequisite. In this paper, we propose an improved Monte Carlo localization using self-adaptive samples, abbreviated as SAMCL. This algorithm employs a pre-caching technique to reduce the on-line computational burden. Further, we define the concept of similar energy region (SER), which is a set of poses (grid cells) having similar energy with the robot in the robot space. By distributing global samples in SER instead of distributing randomly in the map, SAMCL obtains a better performance in localization. Position tracking, global localization and the kidnapped robot problem are the three sub-problems of the localization problem. Most localization approaches focus on solving one of these sub-problems. However, SAMCL solves all these three sub-problems together thanks to self-adaptive samples that can automatically separate themselves into a global sample set and a local sample set according to needs. The validity and the efficiency of our algorithm are demonstrated by experiments carried out with different intentions. Extensive experiment results and comparisons are also given in this paper.**

I. INTRODUCTION

Localization is the problem of determining the pose of a robot given a map of the environment and sensors data [1], [2], [3]. The robot pose comprises its location and orientation relative to a global coordinate frame. Localization problem can be divided into three sub-problems: position tracking, global localization and the kidnapped robot problem [4], [5], [6]. Position tracking assumes that the robot knows its initial pose [7], [8]. During its motions, the robot can keep track of movement to maintain a precise estimate of its pose by accommodating the relatively small noise in a known environment. More challenging is the global localization problem [4], [9]. In this case, the robot does not know its initial pose, thus it has to determine its pose in the following process only with control data and sensors data. Once the robot determines its global position, the process continues as a position tracking problem. The kidnapped robot problem is that a well-localized robot is taken to some other place without being told [4], [6], [10]. In practice, the robot is rarely kidnapped. However, kidnapping tests the ability of a localization algorithm to recover from global localization failures. This problem is more difficult than global localization. Difficulties appear in two aspects: one is how a robot knows it is kidnapped, the other is how to recover from kidnapping. The latter can be processed as a global localization problem.

Among the existing position tracking algorithms, the Extended Kalman Filter (EKF) is one of the most popular approaches [6], [11], [12]. EKF assumes that the state transition and the measurements are Markov processes represented by nonlinear functions. The first step consists in linearizing these functions by Taylor expansion and the second step consists in a fusion of sensors and odometry data with Kalman Filter. However, plain EKF is inapplicable to the global localization problem, because of the restrictive nature of the unimodal belief representation. Monte Carlo localization (MCL) is the most common approach to deal with the global localization problem. MCL is based on a particle filter that represents the posterior belief by a set of weighted samples (also called particles) distributed according to this posterior [6], [13], [14]. MCL is already an efficient algorithm, as it only calculates the posteriors of particles. However, to obtain a reliable localization result, a certain number of particles will be needed. The larger the environment is, the more particles are needed. Actually each particle can be seen as a pseudo-robot, which perceives the environment using a probabilistic measurement model. At each iteration, the virtual measurement takes large computational costs if there are hundreds of particles. Furthermore, the fact that MCL cannot recover from robot kidnapping is its another disadvantage. When the position of the robot is well determined, samples only survive near a single pose. If this pose happens to be incorrect, MCL is unable to recover from this global localization failure. Thrun *et al.* [6] proposed the Augmented_MCL algorithm to solve the kidnapped robot problem by adding random samples. However, adding random samples can cause the extension of the particle set if the algorithm cannot recover quickly from kidnapping. This algorithm draws particles either according to a uniform distribution over the pose space or according to the measurement distribution. The former is inefficient and the latter can only fit the landmark detection model (feature-based localization). Moreover, by augmenting the sample set through uniformly distributed samples is mathematically questionable. Thus, Thrun *et al.* [6], [10], [15] proposed the Mixture MCL algorithm. This algorithm employs a mixture proposal distribution that combines regular MCL sampling with an inversed MCL's sampling process. They think that the key disadvantage of Mixture MCL is a requirement for a sensor model that permits fast sampling of poses. To overcome this difficulty, they use sufficient statistics and density trees to learn a sampling model from data.

In this paper, we propose an improved Monte Carlo localization with self-adaptive samples (SAMCL) to solve the localization problem. This algorithm employs a pre-caching technique to reduce the on-line computational burden of MCL. Thrun *et al.* [6] use this technique to reduce costs of computing for beam-based models in the ray casting operation. Our pre-caching technique decomposes the state space into two types of grids. The first one is a three-dimensional grid that includes the planar coordinates and the orientation of the robot. This grid, denoted as G_{3D}, is used to reduce the on-line computational burden of MCL. The other grid is a two dimensional energy grid. We define energy as the total information of measurements (the more information, the higher energy). The energy grid, denoted as G_E, is used to calculate the Similar Energy Region (SER) which is a subset of G_E. Its elements are these grid cells whose energy is similar to robot's energy. SER provides a priori information of robot's position, thus, sampling in SER is more efficient than sampling randomly in the whole map. Finally, SAMCL can solve position tracking, global localization and the kidnapped robot problem together thanks to self-adaptive samples. Self-adaptive samples in this paper are different from the KLD-Sampling algorithm proposed in [6], [16]. The KLD-Sampling algorithm employs the sample set that has an adaptive size to increase the efficiency of particle filters. Our self-adaptive sample set has a fixed size, thus it does not lead to the extension of the particle set. This sample set can automatically divide itself into a global sample set and a local sample set according to different situations, such as when the robot is kidnapped or fails to localize globally. Local samples are used to track the robot's pose, while global samples are distributed in SER and used to find the new position of the robot.

The rest of this paper is organized as follows. In section II, we briefly review Monte Carlo localization. In section III, we introduce the SAMCL algorithm. Experiment results are presented in section IV and finally some conclusions are given in section V.

II. MONTE CARLO LOCALIZATION

In the probabilistic framework, the localization problem is described as estimating a posterior belief of the robot's pose at present moment conditioned on the whole history of available data. For mobile robots, the available data are of two types: perceptual data and odometry data [6].

$$bel(x_t) = p\left(x_t \left| z_{0:t}, u_{1:t}\right.\right) \tag{1}$$

Where x_t is robot's pose at time t, which is composed by its two-dimensional planar coordinates and its orientation. The belief function $bel(x_t)$ represents the density of probability of the pose x_t. The term $z_{0:t}$ represents all the exteroceptive measurements from time $\tau = 0$ to $\tau = t$ and $u_{1:t}$ represents control data from time $\tau = 1$ to $\tau = t$.

Equation (1) is transformed by Bayes rule, the Markov assumption and the law of total probability, to obtain the final recursive equation:

$$bel(x_t) = \eta p\left(z_t \left| x_t\right.\right) \int p\left(x_t \left| x_{t-1}, u_t\right.\right) bel(x_{t-1}) dx_{t-1} \tag{2}$$

where η is a normalization constant that ensures $bel(x_t)$ to sum up to one. The probability $p\left(x_t \left| x_{t-1}, u_t\right.\right)$ is called the prediction model or the motion model, which denotes the transition of robot state. The probability $p\left(z_t \left| x_t\right.\right)$ is the correction model or the sensor model, which incorporates sensors information to update robot state.

Different localization approaches represent the posterior $bel(x_t)$ in different ways. MCL represents this posterior belief by a set of N weighted particles distributed according to this posterior [6], [10]:

$$bel(x_t) \propto \left\{ \left\langle x_t^{[n]}, \omega_t^{[n]} \right\rangle \right\}_{n=1,\cdots,N} \tag{3}$$

here $x_t^{[n]}$ is a particle that represents a hypothesized pose of robot at time t. The non-negative numerical parameter $\omega_t^{[n]}$ is the importance factor which gives a weight to each particle. The initial belief $bel(x_0)$ may be represented by particles drawn according to a uniform distribution over the state space if the initial pose of the robot is unknown, or by particles drawn from a Gaussian distribution centered on the correct pose if the initial pose is known approximately.

III. THE SAMCL ALGORITHM

The SAMCL algorithm can solve efficiently all the three sub-problems of localization together. The whole process is illustrated in Fig. 1. SAMCL is implemented in three steps: (1) Pre-caching the map, (2) Calculating SER, (3) Localization. The first step is executed off line, the other two steps are run on line.

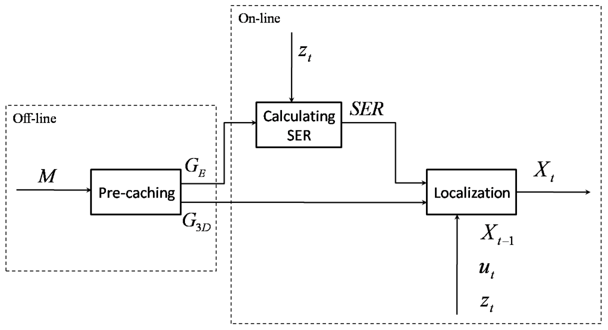


Fig. 1. The process of the SAMCL algorithm.

A. Pre-caching the Map

In the localization problem, the map is supposed to be known by the robot. Hence, a main idea is to decompose the given map into grid and to pre-compute measurements for each grid cell. Our pre-caching technique decomposes the state space into two types of grids.

Three-dimensional grid (G_{3D}). It includes planar coordinates and the orientation. Each grid cell is seen as a pseudo-robot that perceives the environment and stores these measurements. When SAMCL is implemented, instead of computing measurements of the map for each particle on line, the particle is matched with the nearest grid cell and then simulated perceptions stored in this cell are assigned to the particle. Measurements are pre-cached off line, hence the pre-caching technique reduces the on-line computational burden. Obviously, the precision of the map describing depends on the resolution of the grid.

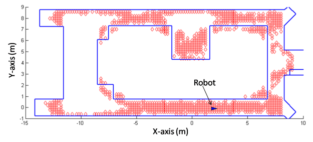
Two-dimensional energy grid (G_E). In this grid, each grid cell pre-computes and stores its energy. Energy is the total information of measurements. For distance sensors, we define i^{th} sensor's energy as $1 - d_i/d_{\max}$, d_i is the measurement of i^{th} sensor and $d_{\max}$ is the maximum distance that sensors are able to "see". Then we calculate the sum of energy of all the sensors. The advantage of using total energy of all the sensors is no need to consider the orientation of the robot, thus we can reduce one-dimensional calculation. Please note that we can calculate the sum of energy to reduce one-dimensional calculation based on an assumption that the robot has a circular or quasi-circular body and sensors are distributed uniformly around its circumference. If a robot has non-circular body or non-uniformly distributed sensors, it will obtain different energy at the same location but different orientation. Hence, we have to consider the orientation when we use these robots. These grid cells nearby obstacles will have larger energy than those in the free space. The inputs of this step are the map M. The outputs are a three-dimensional grid (G_{3D}) and a two-dimensional energy grid (G_E). The process of calculating energy for grid cells is shown in Alg. 1. Here $\tilde{a}_i(k)$ represents energy of the i^{th} sensor of the k^{th} cell, $\tilde{E}(k)$ is total energy of the I sensors of the k^{th} grid cell. In line 6, we normalize total energy $\tilde{E}(k)$. Like this, energy $\tilde{a}_i(k)$ and total energy $\tilde{E}(k)$ has the same value interval $[0, 1]$ as probability density. This energy grid is used to calculate SER that will be presented in Section III-B.

```
1: for all the grid cell $k \in \{1, \cdots, K\}$ do
2:     for all the distance sensors $i \in \{1, \cdots, I\}$, each measurement $\tilde{d}_i(k) < d_{max}$ do
3:         $\tilde{a}_i(k) = 1 - \tilde{d}_i(k)/d_{\max}$
4:         $\tilde{E}(k) = \sum_{i=1}^{I} \tilde{a}_i(k)$
5:     end for
6:     normalize $\tilde{E}(k) = \frac{1}{I}\tilde{E}(k)$
7: end for
```

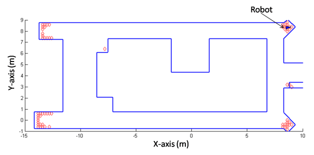
Algorithm 1: Calculating energy for each grid cell

B. Calculating SER

Similar energy region (SER) is defined as a subset of G_E. Grid cells in SER have similar energy with the robot. SER may be seen as the candidate region for sampling, in which particles have higher probability. Information provided by SER is used to match the position of the robot, such as the robot is in the corridor or in the corner, is nearby obstacles or in the free space. Fig 2 shows SER when the real robot is located in a corridor (a) and in a corner (b). To distribute global samples, SER provides a priori choice. Obviously, sampling in SER is more efficient than sampling stochastically in the entire map. Especially, if the robot is in a distinct region such as Fig. 2(b), the advantage of sampling in SER is more significant.



(a)



(b)

Fig. 2. SER when the robot is in the corridor (a) and in the corner (b).

The inputs of this step are the energy grid G_E obtained off-line in the pre-caching phase and the range measurements of the robot at time t. The output is SER. In Alg. 2, a_i represents energy of the i^{th} sensor of the real robot, E is total energy of the I sensors of the real robot, which is normalized in line 5. δ is a given threshold that determines the size of SER.

```
1: for all the distance sensors $i \in \{1, \cdots, I\}$, each measurement $d_i < d_{max}$ do
2:     $a_i = 1 - d_i/d_{\max}$
3:     $E = \sum_{i=1}^{I} a_i$
4: end for
5: normalize $E = \frac{1}{I}E$
6: for all the grid cell $k \in \{1, \cdots, K\}$ do
7:     defining SER in the grid cell $k$ if $\left|E - \tilde{E}(k)\right| < \delta$
8: end for
```

Algorithm 2: The calculating SER algorithm

C. Localization

The SAMCL algorithm uses self-adaptive samples to solve the position tracking, global localization and the kidnapped robot problems together. Self-adaptive samples can automatically divide themselves into a local sample set and a global sample set and transform between them according to different situations. SAMCL maintains local samples by MCL and distributes global samples in SER. When the robot is well localized, SAMCL only maintains local samples around the robot. Once the robot is kidnapped, part of samples migrate from local samples to global samples. After the robot re-localizes itself, global samples are converted as one part of local samples. Global samples are able to help the robot recover from kidnapping. But they may also induce a wrong reaction, for instance, in symmetrical environments, all the particles in symmetrical regions may have high probability and the pose of robot could be ambiguous. Hence, global samples should only appear when the robot is "really" kidnapped. We value whether the robot is kidnapped by measuring the probabilities of particles. If the maximum of probabilities of particles is less than a threshold, the robot will deduce that it has been kidnapped.

The inputs of the last step are the particle set X_{t-1}, motion control u_t, measurements z_t, the three-dimensional grid G_{3D} and SER. Its output is the particle set X_t. The SAMCL algorithm is summarized in Alg. 3, N_T denotes the total number of particles used in this algorithm, N_G is the number of global samples distributed in SER, and N_L denotes the number of local samples used for tracking the robot. We explain this algorithm in five parts.

Part1: sampling total particles. Line 2 generates a particle $x_t^{[n]}$ for time t based on the particle $x_{t-1}^{[n]}$ and the control u_t. Line 3 determines the importance weight of that particle. Particularly, measurements of the particle are searched in G_{3D}.

Part2: determining the size of the global sample set and the local sample set. It distributes the number of global samples and local samples according to the maximum of importance factors ω_t. If ω_t^{max} is less than the threshold ξ, we assume the robot is kidnapped, part of particles N_G are divided as global samples. If not, all the particles are local samples. α determines the ratio of global samples and local samples. The reason why we do not use all the particles as global samples is that the robot may mistakenly believe that it is kidnapped. This more often occurs in incomplete maps. Keeping part of local samples can reduce this mistake. ξ is a sensitive coefficient, which determines the sensitivity of SAMCL. The greater ξ may make robot more sensitive to kidnapping, but on the other hand the robot mistakes more frequently.

Part3: resampling local samples. It is the operation to resample local samples that is identical to regular MCL. At the beginning, importance factors ω_t are normalized. Local samples are drawn by incorporating the importance weights.

Part4: drawing global samples. A real trick of the SAMCL algorithm is in part 4, global samples are distributed in SER with a uniform distribution. The advantage of sampling in SER is more efficient. This part is only executed when the robot considers itself to be kidnapped.

Part5: combining two particles sets. At last, local sample set X_t^L and global sample set X_t^G are combined. The new sample set X_t will be used in the next iteration.

```
Sampling total particles
1: for $n = 1$ to $N_T$ do
2:     generate a particle $x_t^{[n]} \propto p\left(x_t \left| x_{t-1}^{[n]}, u_t\right.\right)$
3:     calculate importance factor $\omega_t^{[n]} = p\left(z_t \left| x_t^{[n]}, G_{3D}\right.\right)$
4: end for
Determining the size of the global sample set and the local sample set
1: if $\omega_t^{max} < \xi$ then
2:     $N_L = \alpha \cdot N_T$
3: else
4:     $N_L = N_T$
5: end if
6: $N_G = N_T - N_L$
Resampling local samples
1: normalize $\omega_t$
2: for $n = 1$ to $N_L$ do
3:     draw $x_t^{[n],L}$ with distribution $\omega_t^{[n]}$
4:     add $x_t^{[n],L}$ to $X_t^L$
5: end for
Drawing global samples
1: for $n = 1$ to $N_G$ do
2:     draw $x_t^{[n],G}$ with uniform distribution in SER
3:     add $x_t^{[n],G}$ to $X_t^G$
4: end for
Combining two particle sets
1: $X_t = X_t^L \cup X_t^G$
2: return $X_t$
```

Algorithm 3: The SAMCL algorithm

IV. Experiments

The SAMCL algorithm described in this paper has been tested with a Pioneer 3-DX mobile robot in a real office environment. The robot has an onboard laptop with 1.06GHz Intel Core 2 Solo U2100 CPU and 1024M of RAM, and SAMCL is implemented with MATLAB. The experiment environment is the first floor of our laboratory. Fig. 3 shows the ground plan and the expected trajectory. The robot should follow this trajectory and go around in the corridor. The real environment of this corridor is shown in pictures. There are several unmodeled obstacles in the corridor, such as cabinets and tables (see pictures A and B of Fig. 3). We use this incomplete map to test the robustness of our algorithm. SAMCL inherits the advantage of MCL, so it can treat these unmodeled obstacles as sensors noise. Because our map is quasi-symmetrical, to recover from kidnapping in this map is more difficult. The resolution of the three-dimensional grid G_{3D} is $0.2m \times 0.2m \times pi/32$ and the resolution of the energy grid G_E is $0.2m \times 0.2m$ in our experiments. The size of the experiment map is about $25m \times 10m$.

Two experiments were performed (see the video attached to this paper). The first one aimed at testing the ability of global localization and the robustness of the SAMCL algorithm by adding artificial errors to wheel encoder reading. The second one focused on testing the ability of recovering from kidnapping. In order to get reliable statistical results, each experiment was repeated 20 times.

A. Global Localization with Artificial Errors

In the first experiment the robot would localize itself with unfaithful odometry. In practice, these enormous errors of odometry are often caused by wheels sliding on the slippery ground or by the robot passing the concave-convex road. In order to simulate coarse odometry, we added about 27% artificial errors to each wheel. Because of testing the ability of localization, the sensitive coefficient ξ was given a low sensitive value.

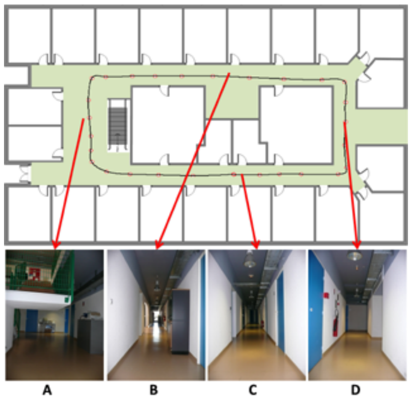


Fig. 3. The ground plan including the expected trajectory. Pictures show the real environment (with unmodeled obstacles).

The localization result is illustrated in Fig. 4, line A and line B represent the trajectories of the weighted average of particles and odometry, respectively. The weighted average of particles is obtained by $x_t = \sum_{n=1}^{N} x_t^{[n]} * \omega_t^{[n]}$. As we can see, odometry has totally lost because of gradually accumulated errors. On the contrary, SAMCL still gives a good localization result. Average errors of final poses of localization and odometry are shown in Table I.

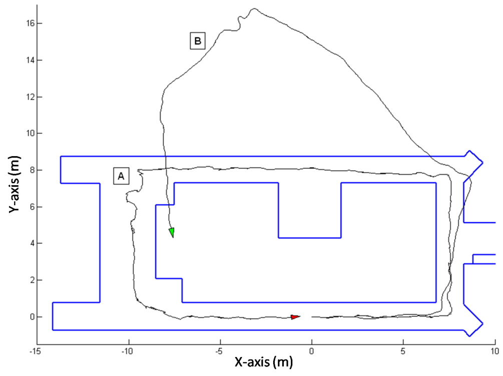


Fig. 4. The result of global localization with artificial errors. Line A and line B present the trajectories of the weighted average of particles and odometry, respectively.

TABLE I
AVERAGE ERRORS OF THE FINAL POSE OF GLOBAL LOCALIZATION WITH ARTIFICIAL ERRORS

	x	y	θ
Localization	$0.469m$	$0.031m$	$17.1°$
Odometry	$6.353m$	$7.301m$	$72.5°$

B. Kidnapping

The second experiment demonstrates the ability of the SAMCL algorithm to recover from kidnapping, which is the most difficult issue. We kidnapped the robot at the beginning of the trajectory after particles converging. Put differently, after the robot was well localized, we took it to about $7m$ far away in its moving direction. Moreover, we added about 27% artificial errors to each wheel. In order to make the robot find kidnapping more quickly, the sensitive coefficient ξ was given a medium sensitive value.

Fig. 5 illustrates the distribution of the self-adaptive sample set during the process of recovering from kidnapping. In the beginning, the robot is well localized as shown in Fig. 5(a). Then the robot is kidnapped from position A to position B (position B is about $7m$ far away from position A in the moving direction of the robot). Next, kidnapping brings on probabilities of particles reducing, when the maximum of probabilities is less than ξ, global samples are divided and distributed in SER, as shown in Fig. 5(b). The robot moves forward and perceives the environment. Because of the quasi-symmetry of environment, SAMCL gives out three probable poses of the robot after resampling, depicted in Fig. 5(c). The robot continues to move and perceive, SAMCL discards two probable poses and confirms the correct pose of robot, shown in Fig. 5(d).

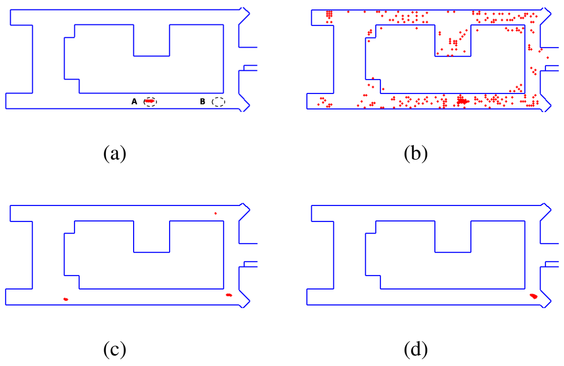
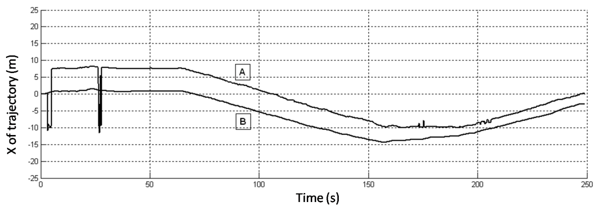


Fig. 5. The distribution of the self-adaptive sample set during the process of recovering from kidnapping.

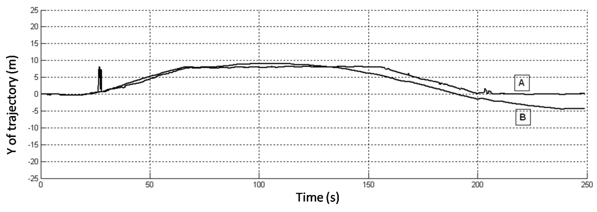
In this experiment, the final pose of the Pioneer robot is measured, that is $x = 0.79, y = 0.02$ in the Cartesian coordinate. For the convenience of analysis, trajectories given by the weighted average of particles (line A) and odometry (line B) are decomposed to X-axis and Y-axis. As shown in Fig. 6, the final pose of localization is $x = 0.43, y = 0.09$, but the final pose of odometry is $x = -2.96, y = -4.35$. Obviously, the localization result is better than odometry. From the figure, we can also find that the robot discovers kidnapping at $3^{rd}s$ and recovers at $6^{th}s$. In the later process, it mistakes once, but it re-localizes in less than $2s$ interval. Average errors of final pose of localization and odometry are shown in Table II.

TABLE II
AVERAGE ERRORS OF THE FINAL POSE OF KIDNAPPING

	x	y	θ
Localization	$0.605m$	$0.076m$	$13.2°$
Odometry	$5.6728m$	$5.017m$	$45.3°$



(a)



(b)

Fig. 6. The result of kidnapping. Trajectories are decomposed to X-axis (a) and Y-axis (b). Line A and line B depict the trajectories of the weighted average of particles and odometry, respectively.

C. *Comparisons*

Results of this part are obtained by simulation. Fig. 7 shows the success rate of recovering from kidnapping as a function of the number of particles. The success rate increases with the number of particles, both for sampling in SER and for sampling randomly. However, with the same size particle set, the success rate of sampling in SER is much higher than sampling randomly. For example, when using 300 particles, the success rate of sampling in SER may achieve to 33%, while this rate of sampling randomly is only 11%. To reach the same success rate, sampling randomly has to use 900 particles, while using 900 particles, the success rate of sampling in SER has achived to 91%.

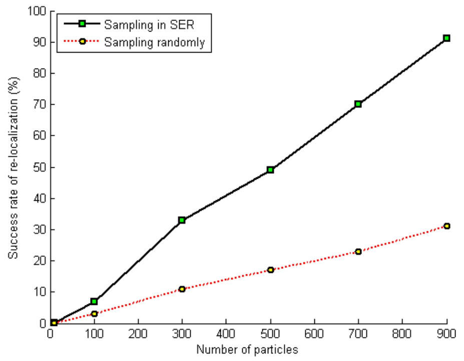


Fig. 7. The success rate of recovering from kidnapping as a function of the number of particles.

V. CONCLUSIONS

In this paper, we proposed an improved Monte Carlo localization with self-adaptive samples (SAMCL) to solve the localization problem. This algorithm inherits all the advantages of MCL, moreover it improves in several aspects. SAMCL employs an off-line pre-caching technique to solve the expensive on-line computational cost problem of regular MCL. We define Similar Energy Region (SER) in this paper. SER provides a priori information of the robot's pose. Hence sampling in SER is more efficient than sampling randomly in the entire environment. Because of using self-adaptive samples, SAMCL can deal with the kidnapped robot problem as well as position tracking and global localization.

We designed two experiments to verify the validity of the SAMCL algorithm. The first one achieved global localization even by adding artificial errors to wheel encoder reading. The second one confirmed the ability of recovering from kidnapping. The simulation results show the success rate of sampling in SER is much higher than sampling randomly with the same size of particle set.

The future work would address to the issue of applying the SAMCL algorithm in the multi-robot localization problem.

REFERENCES

[1] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, pp. 195–207, 1998.

[2] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," in *Proceedings of IJCAI-97*. Morgan Kaufmann, 1997.

[3] D. Fox, W. Burgard, and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[4] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Probabilistic algorithms and the interactive museum tour-guide robot minerva," *International Journal of Robotics Research*, vol. 19, pp. 972–999, 2000.

[5] S. I. Roumeliotis and G. A. Bekey, "Bayesian estimation and kalman filtering: a unified framework for mobile robot localization," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '00)*, vol. 3, 2000, pp. 2985–2992.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, September 2005.

[7] B. Schiele and J. L. Crowley, "A comparison of position estimation techniques using occupancy," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, 1994, pp. 1628–1634.

[8] G. Weiss, C. Wetzler, and E. von Puttkamer, "Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans," in *Proceedings of the International Conference on Intelligent Robots and Systems*, vol. 1, 1994, pp. 595–601.

[9] A. Milstein, J. N. Sánchez, and E. T. Williamson, "Robust global localization using clustered particle filtering," in *AAAI-02*, 2002, pp. 581–586.

[10] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[11] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[12] M. S. Grewal and A. P. Andrews, *Kalman filtering: theory and practice*. Prentice-Hall, Inc., 1993.

[13] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo localization: Efficient position estimation for mobile robots," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, July 1999, pp. 343–349.

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

[15] S. Thrun, D. Fox, and W. Burgard, "Monte Carlo localization with mixture proposal distribution," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2000, pp. 859–865.

[16] D. Fox, "Adapting the sample size in particle filters through kld-sampling," *International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1003, 2003.