Line-based SMC SLAM Method in Environment with Polygonal Obstacles

David C.K. Yuen and Bruce A. MacDonald

Department of Electrical and Electronic Engineering,
University of Auckland, Private Bag 92019, Auckland, New Zealand.
d.yuen@auckland.ac.nz, b.macdonald@auckland.ac.nz

Abstract

A Sequential Monte Carlo Simultaneous Localisation and Map-building (SMC SLAM) algorithm based on a multiple particle filter architecture has been devised for a robot operating in a polygonal obstacle-filled environment. Obstacles are represented as lines, rather than the points used in previous work. Inherited the flexible assumptions of SMC methods, the proposed algorithms accepts non-Gaussian noise distribution. A scanning range finder is assumed to be the main robot sensor. Careful selection of extracted obstacle features reduces computation time while allowing the use of standard SMC updating method. The experimental results compare well with the established Extended Kalman Filtering method.

1 Introduction

Autonomous robots are expected to adapt to the ever changing environment with minimal user intervention. The assumed use of *a priori* map by many localisation algorithm is not acceptable. The Simultaneous Localisation And Map-building (SLAM) method builds a map concurrently when the robot is estimating its own position in the initially empty internal map. Its implementation should deliver a good answer to the challenging localisation requirement of autonomous robots. SLAM can be considered as a state and parameter estimation problem, with the robot positions being the states and the static obstacle positions being the parameters. It differs from ordinary state and parameter estimation problems as the number of parameters to be estimated is unknown initially, and changes as new obstacles are encountered. It is vital to ensure the computation complexity and the integrity of the estimation have not adversely affected upon the detection of new obstacles when modifications are being made to existing algorithms.

Extended Kalman Filtering (EKF) is the prevalent SLAM technique [Dissanayake *et al.*, 2001; Guivant and Nebot, 2001]. It extends the original Kalman filtering method to those nonlinear systems modelled well with a first order Taylor approximation. Other limitations, such as the assumption of a Gaussian noise model, have not been overcome. The Sequential Monte Carlo (SMC) method, also known as particle filtering, is a versatile simulation-based approach to compute the posterior distribution. The SMC method has been adopted in robot localisation research and shown remarkable improvement for many localisation problems, including position tracking, global localisation and even the difficult kidnapped robot problem [Doucet *et al.*, 2000; Fox *et al.*, 1999]. However, its application to SLAM is still in its early stage. Murphy [2000] applied the Rao-Blackwellised particle filter to solve the SLAM problem but only in a 10×10 grid world. Rather than simply augmenting the new obstacle position to a single set of particle filters, an alternative algorithm, which initialises multiple particle filters to estimate the state and parameters separately, has extended the SMC application to solve the continuous state-space SLAM problem [Yuen and MacDonald, 2002]. The novel multiple filter structure keeps the computation under control when extra obstacles are added. Based on simplified models for both the robot and the environment, the simulation gives promising results.

The SMC SLAM algorithm outlined in [Yuen and MacDonald, 2002] processes each set of data as a consecutive data sequence. It approximates the obstacles by points, which may not be the best obstacle representation for our intended indoor robot application. In many indoor environments, lines will be the dominant extracted feature if the robot is equipped with a ranging sensor, such as a laser range scanner. Representing these line obstacles by series of points is not only inefficient, but also affects the probability calculation as the relationship between the extracted features has not been expressed properly. Besides, a set
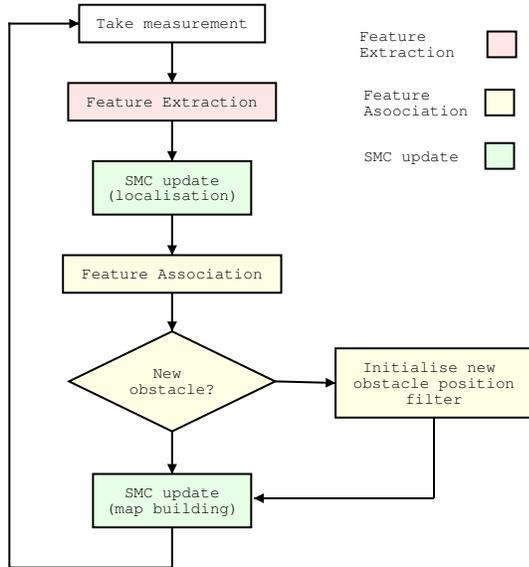
Figure 1: The flowchart of the SMC SLAM algorithm

of somehow arbitrary herustic rules has been adopted to handle the large number of detected point obstacles. In order to reduce the number of active particle filter, obstacle position filters are removed while their estimates are extracted to an internal map whenever the obstacle position estimates become reasonably stable. These herustic rules are difficult to maintain and are vulnerable to real world modelling errors and noises.

In this work, a feature extraction stage is introduced to provide a more compact data representation for the subsequent stages in our SMC SLAM algorithm. Line is the chosen feature for our targeted indoor robot applications. The algorithm can be decomposed into a number of modules: feature extraction, feature association/new obstacle detection and SMC update (Figure 1). Rather than adopting the more straight-forward 2-point line segment expression [Yuen and MacDonald, 2003], all the obstacles are represented in (ρ, θ) form. The obstacle detection module is simplified considerably upon the improved feature extraction stage.

Section 2 is a brief introduction to SMC method. In Section 3, we provides an overview of the new, line based, continuous state-space SMC SLAM algorithm before discussing the implementation details of the feature extraction (Section 3.2), new obstacle detection/feature association (Section 3.3) and SMC update (Section 3.4). The algorithm has been implemented on our indoor research B21r robot and the experimental results are compared with the established EKF method in Section 4 before concluding the paper in Section 5.

2   Introduction to SMC

The section provides a brief introduction to the SMC methods, also appeared in our early work in [Yuen and MacDonald, 2002]. More detailed overviews of

SMC and SMC based robot localisation are presented in Arulampalam *et al.* [2002] and Fox *et al.* [2003], respectively.

SLAM estimates the robot and obstacle positions. Fundamentally, it is the search for the hidden states and unknown parameters for the system. SMC methods are primarily developed to estimate the posterior distribution of a system with hidden states $\{x_k; k \in \mathbb{N}\}$, using the available observations $\{z_k; k \in \mathbb{N}\}$. SMC methods have been applied successfully to solve many different robot localisation problems [Doucet *et al.*, 2000; Fox *et al.*, 1999].

SMC is a simulation-based realisation of the Bayes Theorem (Equation 1) which aims to calculate the posterior distribution from the available observation data.

$$p(x_{k+1}|z_{k+1}) =$$
$$p(x_{k+1})\frac{p(z_{k+1}|x_{k+1})}{\int p(z_{k+1}|x_{k+1})p(x_{k+1})dx_{k+1}} \quad (1)$$

A set of P estimates is kept in each SMC filter. SMC method is also known as particle filtering because of this. In addition to the estimated states (or parameters) x, an importance factor f is assigned to each particle. A set of particles Ψ in a filter is defined as $\Psi = \{x^i, f^i\}_{i=1:P.}$. The particle states are randomised at initialisation with the importance factor set to $\frac{1}{P}$.

Belief [1] is a common notation for the posterior in much robotics and AI literature.

$$Bel(x_{k+1}) = p(x_{k+1}|z_{k+1}, u_k) \quad (2)$$

In terms of the *Belief*, measurement probability $p(z_{k+1}|x_{k+1})$ and the state transition probability $p(x_{k+1}|x_k, u_k)$, a form of recursive Bayesian filter can be derived:

$$Bel(x_{k+1}) =$$
$$\eta \, p(z_{k+1}|x_{k+1}) \int p(x_{k+1}|x_k, u_k)Bel(x_k) \, dx_k \quad (3)$$

where η represents a normalisation constant.

Algorithm 1 is the simplest SMC implementation. It effectively evaluates the *Belief* expression (Equation 3) from right to left. At time $k + 1$, a particle i is first sampled from Ψ. The chance of being selected is proportional to the importance factor f_k. Then, the hidden state estimation is updated by sampling the state transition probability $p(x_{k+1}|x_k, u_k)$. Finally, the measurement probability $p(z_{k+1}|x_{k+1})$ is evaluated. The new importance factor f_{k+1} is obtained by scaling the previous factor f_k by $p(z_{k+1}|x_{k+1})$. In addition to the state transition and measurement probability distribution, the presence of a system $x_{k+1} = F_{sys}(x_k, u_k)$ and

---

[1] Let u be the input signal. The influence of u cannot be observed till one step later in discrete system. Therefore, the input signal that affects step $k + 1$ should be u_k rather than u_{k+1}.

a perceptual model $z_{k+1} = F_{perceptual}(x_{k+1})$ are assumed. With the system model, the predicted state in the $k+1$ step can be calculated from the previous state x_k and the input u_k. The predicted measurement can in turn be calculated from the predicted state using the perceptual model. If the predicted measurement is close to that of the actual observation, the state prediction is more likely to be close to the actual state. A higher importance factor will be allocated. The procedure is repeated sequentially for each particle in each round of filter update. However, it is vulnerable to degeneracy, which happens when the importance factors for all but one of the particles become negligible.

Algorithm 1 Simplest SMC Implementation

---

for each particle in Ψ **do**
   1. Importance sampling
   2. State estimation update
   3. Importance factor update
end for

Algorithm 2 avoids degeneracy by using Sampling Importance Resampling (SIR) [Gordon *et al.*, 1993], which introduces the normalisation and resampling stage at the end of each update. The resampling procedure is similar to the importance sampling procedure mentioned, but the importance factor of all resampled particles would be reset to $\frac{1}{P}$. A particle with a higher importance factor f would have a proportionally higher chance of getting resampled. The SIR algorithm forms the basis of our multiple particle filter architecture.

Algorithm 2 SIR Particle Filter

---

for each particle in Ψ **do**
   State estimation update
   Importance factor update
end for
Calculate: $f_{sum} = \sum_{i=1:P} f_k^i$
for $i = 1 : P$ **do**
   Normalise: $f_k^i = f_k^i / f_{sum}$
end for
Resample$[\{x_k^i, f_k^i\}_{i=1:P}]$

---

3 Line based SMC SLAM method

3.1 Overview

In SLAM, the robot position is estimated as the states, while the static obstacle positions are estimated as the time-invariant parameters. The new obstacles are detected and added to the parameter estimation list as the robot advances. SLAM can rely only on the onboard sensors. To perceive the surroundings, many robots are equipped with a ring or rings of range finding sensors[2]. Since these sensor arrays give highly correlated data, it is more efficient to extract prominent features from them before further processing.

---

[2]Another common configuration is to mount a sensor on a rotating platform. Functionally, it is the same as a ring of sensors.

Algorithm 3 outlines the line based SLAM algorithm. Instead of using a single particle filter to estimate both the robot and obstacle positions, the multiple particle filters idea proposed in our previous work [Yuen and MacDonald, 2002] is also adopted in this study. One particle filter is devoted to estimate the robot position Ψ_0 and one also for each of the obstacle positions $\Psi_{1:L}$.

Algorithm 3 SMC SLAM

Initialise state filter Ψ_0 for robot position
$L = 0$ {L is the total number of new obstacles}
repeat
   $\lambda_{robotCentric}$=*FeatureExtract(*ζ_{k+1}*)*
   SMC update for robot position filter Ψ_0
   z_{k+1}=*ToGlobalFrame(*$\lambda_{robotCentric}$,$\bar{s}_{k+1}$*)*
   if *FoundNewObstacle()* **then**
     $L = L + 1$
     Initialise new parameter filter Ψ_L
   end if
   $M_{\varphi z}$=*FeatureAssociation(*$\bar{z}_{k+1}$, $\bar{\varphi}_k^{1:L}$*)*
   for $i = 1 : L$ **do**
     if $M_{\varphi z}(i) \neq \emptyset$ **then**
       SMC update for obstacle position filter Ψ_i
     end if
   end for
until the last time step

The current implementation assumes laser range scanner readings as the raw input ζ. Line segments $\lambda_{robotCentric}$ are extracted from the measurement data in the feature extraction stage (*FeatureExtract()* in Section 3.2) before SMC updating for the robot position (Section 3.4 – localisation). Although lines are the chosen features in this study, the discussion could be generalised to other types of features.

Local line segments are converted to the normal form with reference to the global origin in *ToGlobalFrame()* with Equation 9. The observed locations of the lines are compared with those already known to detect new obstacles (Section 3.3). An extra particle filter will be created to estimate the position of each new obstacle. The feature extraction stage significantly reduces the data dimensions and often results in good performance improvement to the implementation. However, the number and the order of features extracted from each set of measurement can change. Feature association is a necessary stage before obstacle positions can be updated. A corresponding feature list $M_{\varphi z}$ between the current measurement $\bar{z}_{k+1}$ and the existing obstacle position estimates $\bar{\varphi}_k^{1:L}$ is generated. The obstacle position filters which has found corresponding features from the latest observation are then updated one at each time in a randomised order (Section 3.4 – map building).

3.2 Feature Extraction

Scanning laser range data is assumed to be available during the development of the feature extraction module. LMS-200 is a typical laser range scanner. Some
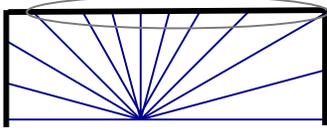
Figure 2: Over-representation of data: if each raw data were treated as a separated measurement, the circled line would be sampled much more frequently than the rest in the posterior distribution calculation during SMC update.
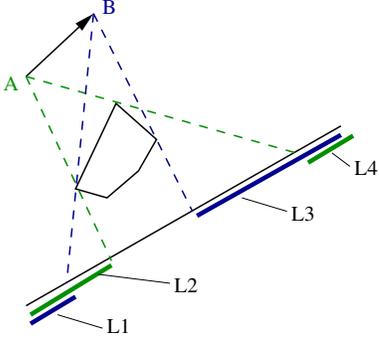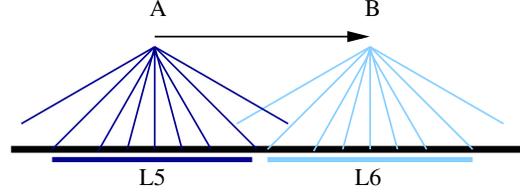

Figure 4: Limited sensor range effect: when the robot moves from A to B, the observed line segment seems to be different. In fact, it is the result of limited sensor range and both lines can be described by the same line equation.

Figure 3: Occlusion effect: when the robot moves from A to B, the view of L2 & L4 seems to be replaced by L1 & L3, but all four line segments actually belong to the same line.

of its operating parameters are taken as a design reference. In a typical operating mode, the unit can return the obstacle distance at 1° increments for the entire 180° in front of the unit with a maximum range set to 8.20 m. Since the maximum range is smaller than the dimension of the testing space, the map needs to be built incrementally even if the testing hall is empty.

Each laser range scan generates fairly large amounts of data. It is therefore desired to extract features from these highly correlated raw data before further processing. The reduction in data dimension improves the efficiency of the estimation process. Besides, it should be more appropriate to model neighbouring data points as a single object if they are describing the same geometric entity. It also mitigates the data under/over-representation caused by taking measurements at a fixed angular interval as illustrated in Figure 2. A lot more samples will be associated to a particular geometric entity if it lies more orthogonal and closer to the scanning laser beam. The contribution of those obstacles which are slightly further or inclined to the scanning laser beam should not be discounted in the measurement probability calculation (Section 3.4). Since typical indoor environments are filled with polygonal obstacles, it is natural to choose line segments as the basic obstacle representation. A line segment can be described fully with 4 parameters, e.g. the x, y coordinates for the start and end points. However, due to occlusion (Figure 3) and the limited sensor range

(Figure 4), the robot can only sense part of the line segment at any given time. The apparent position of the start and end points of the line segments keep on changing as the robot moves. The estimation for obstacle positions would be upset because the line segment parameters are supposed to be time-invariant. Since these line segments (L1–L4 from Figure 3 and L5,L6 from Figure 4) come from the same line, it is more convenient to replace these line segments with the corresponding line equation.

Starting from one end of the scan, line segments are fit to the raw data sequence ζ returned by the laser range scanner. The slope and the intercept of the line segments are found using least-squares.

$$y = mx + c$$

if$(-1 <\text{slope}< 1)$, minimise $\sum_{i=1}^{n} [y_i - (c + mx_i)]^2$

$$c = \frac{\sum y \sum x^2 - \sum x \sum xy}{n \sum x^2 - (\sum x)^2} \tag{4}$$

$$m = \frac{n \sum xy - \sum x \sum y}{n \sum x^2 - (\sum x)^2} \tag{5}$$

else, minimise $\sum_{i=1}^{n} [x_i - \frac{1}{m}(y_i - c)]^2$

$$c = \frac{n \sum y^2 - (\sum y)^2}{\sum x \sum y^2 - \sum y \sum xy} \tag{6}$$

$$m = \frac{n \sum y^2 - (\sum y)^2}{n \sum xy - \sum y \sum x} \tag{7}$$

The average Sum of Squared Error (SSE) is updated when the next point is added to the existing line segment. If the average SSE is larger than threshold $h_{newLine}$, a separated line segment will be introduced. This method tends to associate the first few points of the next line segment to the previous one, until the new line is created. Therefore, the distance from the terminal points to the neighbouring line segments are calculated and would be associated with the closer line segment. Line segments shorter than $h_{minLength}$ are discarded. The resulting list of line segments is denoted as $\lambda_{robotCentric}$.

So far, no assumption is made upon the global robot position. Line segments are recorded as if the robot is the origin in the coordinate system. The line parameters can readily be converted once the relative current robot position to the global reference origin [3] is available in the SMC robot position update step (Section 3.4). To avoid processing problems at a later stage when handling vertical line with the point slope form, we follow the common practice of adopting the normal form and use ρ and θ as the line parameters:

$$x \cos \theta + y \sin \theta = \rho \qquad (8)$$

where ρ is the perpendicular distance of the line from the origin and θ is the orientation of this perpendicular line.

The goal of the *ToGloablFrame()* procedure is to obtain the normal form (ρ, θ) of a line segment $\lambda^i_{robotCentric}$ with respect to the global reference origin. The terminal points of line segment $\lambda^i_{robotCentric}$ can be converted to the global coordinate once the robot position x_r, y_r, ϕ_r is known. The perpendicular point (x_{prep}, y_{prep}) from the origin to this line segment is then calculated as follow:

$$m_1 = \frac{y_2^i - y_1^i}{x_2^i - x_1^i} \qquad (9)$$

$$m_2 = \frac{-1}{m_1} \qquad (10)$$

$$x_{prep} = \frac{-y_1^i + m_1 * x_1^i}{m_1 - m_2} \qquad (11)$$

$$y_{prep} = m_2 \times x_{prep} \qquad (12)$$

ρ and θ are the distance and the orientation angle to this perpendicular point respectively.

3.3 Feature Association and New Obstacle Detection

Since line features rather than raw sensor data are accepted as the processed measurement data, the dimension of the processed measurement vector is no longer constant. To update the estimates in an obstacle position particle filter, it is necessary to identify the corresponding feature currently observed. On the other hand, a new obstacle can be assumed to be found if there are no corresponding previous estimates.

Feature association is started by preparing feature lists for both the current measurement $\bar{z}_{k+1} = \{\rho, \theta\}$ and previous estimates . The previous estimates are stored in the obstacle position particle filters. Since each particle contains slightly different estimates, the centroid of the particles is taken as the representative feature. The procedure is repeated on each obstacle position particle filters to generate the previous estimate feature list $\bar{\varphi}_k^{1:L}$.

The Weighed Absolute Difference (*WAD*) between ρ and θ is calculated as the dissimilarity metric d in this

study.

$$\Delta\theta = min(|\theta_1 - \theta_2|, 2\pi - |\theta_1 - \theta_2|) \qquad (13)$$

$$WAD = c_{w1}|\rho_1 - \rho_2| + c_{w2}|\Delta\theta| \qquad (14)$$

The Hausdorff distance $\mathbb{H}(A, B)$ is the maximum distance of set A to the nearest point in set B [Rot, 1991].

$$\mathbb{H}(A, B) = max_{a \in A}\{min_{b \in B}\{d(a, b)\}\} \qquad (15)$$

The detection of new obstacle can be confirmed if the Hausdorff distance $\mathbb{H}(\bar{z}_{k+1}, \bar{\varphi}_k^{1:L})$ between the features extracted from current measurement $\bar{z}_{k+1}$ and previous particle filter estimates $\bar{\varphi}_k^{1:L}$ is larger than $h_{similarFeature}$. An obstacle position filter is initialised for each newly detected obstacle as illustrated in Section 3.4.

A minimum *WAD* list, which identifies the minimum distance between a previous particle filter estimate and each feature extracted from the current measurement $\bar{z}_{k+1}$, is established. Two features are considered as similar if the difference is smaller than $h_{similarFeature}$. There are two possible outcomes for each entry.

- A similar feature is found: record the indices between the matched pair to the corresponding feature list $M_{\varphi z}$. Suppose feature i in the previous particle filter estimates list $\bar{\varphi}_k^{1:L}$ is similar to the feature j in the current measurement list $\bar{z}_{k+1}$, the entry in the corresponding feature list $M_{\varphi z}(i)$ is set to j.

- A Similar feature cannot be found: the obstacle observed in previous round cannot be found. This can be a result of occlusion by other obstacles, or the robot moving away from the detection range, or erroneous initial observation. Since there is no conclusive evidence, a null entry is recorded, $M_{\varphi z}(i) = \emptyset$, so that the update for the corresponding obstacle position particle filter (Section 3.4) can be disabled.

3.4 SMC update

For a SLAM application, states and parameters are estimated simultaneously. New obstacles can also be detected at any time. The individual filters $\Psi_{0:L}$ for both states and parameter estimation are implemented as unmodified SIR filters. Like most particle filters proposed so far, the SIR filter is developed for systems with a fixed number of states. Instead of inserting the extra parameter estimates directly into the same estimate vector, the estimate vector is split into smaller components and develops a separate estimator for each component. This is similar to the idea of using a multiple model bootstrap filter for tracking rapidly manoeuvring targets [McGinnity and Irwin, 2000]. However, each filter in [McGinnity and Irwin, 2000] estimates the same state while our implementation estimates different components with different filters. The full state x is first split into two components, $x = (s, \varphi)$, where

---

[3]Usually, it is taken as the starting robot position in SLAM, but can be taken as any other arbitrary point.

s is an estimate of the time–varying portion, in this case the robot position (states), and φ is an estimate of the time–invariant portion (parameters), in this case the map of obstacles. The time–invariant set is further divided so that an individual filter only estimates the position for a single obstacle. With the detection of L obstacles, the state filter for robot position estimation is denoted as Ψ_0. The obstacle position filters, i.e. the parameter filters, are denoted as $\Psi_{j=1:L}$.

The subdivision of the full vector from a single filter into multiple filters is appropriate for SLAM application because of the degree of independence between the components. The proposed subdivision of filters offers a number of advantages. First, the particles in the new filter can simply be initialised with an equal importance factor. It is difficult to determine the proper adjustment if extra states are inserted to an existing filter. Second, extra particles are required in more complex system. The interaction between the states implies that the number of extra particles required can be large. The total number of small filters is linearly proportional to the number of detected obstacles after the proposed filter subdivision. Since the small filters are of similar complexity, a fixed number of particles is assigned to each. The extra computation is then linear in the number of detected obstacles.

Although the multiple particle filter approach had been used in [Yuen and MacDonald, 2002], the SMC SLAM updating procedure is different. In [Yuen and MacDonald, 2002], the robot position and obstacle particle filters are updated after the new obstacle detection. The robot position filter Ψ_0 is updated (localisation) before updating of the obstacle position filters $\Psi_{1:L}$ (map building) for this implementation. Due to efficiency concerns, it is not feasible to repeat the feature association process too often. Since the current estimated robot position is required for feature association, it is better to update the robot position filter first.

Localisation

At the start, a particle filter Ψ_0 is initialised to track the robot position. Since SLAM makes no assumption to any external reference frame, rather than spreading them uniformly across the whole work space, it is reasonable to assign the particles randomly around an arbitrary starting point.

For the update of robot position filter Ψ_0, the system model $s_{k+1} = F_{sys}(s_k, u_k)$ predicts the latest robot position from the previous position s_k and the control command issued u_k. The perceptual model evaluates the centroid obstacle positions $\bar{\varphi}_k^{1:L}$ are evaluated from each of the obstacle position filters as the expected measurement. The *ToGlobalFrame()* function is then called to convert the line segments $\lambda_{robotCentric}$ to the line normal form $z = (\rho, \theta)$ by assuming the correct robot position as s_{k+1}. The measurement probability

of the robot position assumes a Laplace distribution [4] with the average WAD between z and $\bar{\varphi}_k^{1:L}$ being the variable and c_{m1} being the parameter. Due to occlusion, not all of the obstacles can be observed at once. The sum of difference calculation only accounts for the observed line obstacles.

Map building

No particle filter is assigned for obstacle position estimation initially. A new SIR obstacle filter is initialised for each newly detected obstacle from the current measurement. The filter particles are assigned randomly around the detected position.

The obstacle position filters $\Psi_{1:L}$ are updated in an randomised order. If the corresponding feature entry $M_{\varphi z}(i)$ indicates that no similar features are currently observed, the updating for filter Ψ_i would be disabled. Since these are parameter filters, the system model simply assumes the actual parameter values to be time-invariant. The previous centroid obstacle position $\bar{\varphi}_i$ is taken as the expected measurement for the perceptual model. The corresponding measurement is identified from $\bar{z}_{k+1}(M_{\varphi z}(i))$. The measurement probability is assumed to follow the Laplace distribution with parameter c_{m2}. Then, the WAD between $\bar{\varphi}_i$ and $\bar{z}_{k+1}(M_{\varphi z}(i))$ is calculated as the variable for the measurement probability distribution.

4 Results and Discussion

The practical choices for our SMC implementation are listed in this paragraph. 2000 particles are assigned for the state filter and 200 are assigned for each parameter filter. The robot position filter estimates the x,y coordinate and the orientation angle ϕ of the robot, while the obstacle position filters estimate the line parameters in normal form, $(\rho_i, \theta_i)_{i=1:L}$. For the feature extraction and association processes, the thresholds $h_{newLine}$, $h_{minLength}$ and $h_{similarFeature}$ are taken as 0.0001 m^2, 0.5 m and 1.0 respectively. WAD was introduced as the dissimilarity metric for the line parameters. The weighings c_{w1} and c_{w2} are taken as 1.0 and 1.5 respectively. The posterior distribution of both the state and parameter estimates follow the exponential distribution. The Laplace distribution parameters c_{m1} and c_{m2} for the measurement probability of the state and parameter filters are taken as 0.2 m^{-1}.

The proposed SMC SLAM method is compared with the established EKF SLAM algorithm. To facilitate the comparison, the EKF implementation follows as close to that of SMC as possible. In EKF SLAM, the robot position x, y, ϕ and obstacle positions $\rho_i, \theta_i; i = 1 : L$ are estimated in the same vector. The EKF implementation adopts the linearised version of the system and perceptual models used in SMC. The new obstacle detection procedure and the associated thresholds are also the same as the SMC implementation.

---

[4]$p(x)dx = \frac{1}{2\mu} exp(-|x/\mu|)dx$

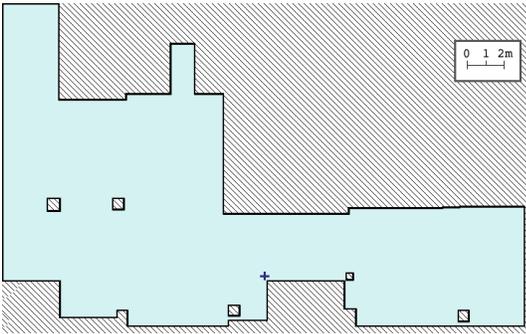Figure 5: A snapshot of the testing environment and the robot

Figure 6: A sketch of the floor plan of the environment. It is for illustration only. No map is supplied at the start of the SLAM operation.

Since EKF accepts only Gaussian noise, the posterior distribution calculation is adjusted accordingly.

A B21r robot equipped with an LMS-200 laser scanner as the main navigation sensor was employed to collect the navigation data for the localisation algorithms. The tests were carried out in the reception area, about 15×15 m, outside our university laboratories (Figure 5). The shape of the room is outlined in Figure 6. A test sequence consists of 406 sets of data points were examined in this section. Figure 7 shows a typical scan by the LMS-200 unit. The scanner covers the 180° in front of the robot. The detection range is 8.2 m for the selected operation mode, which is smaller than the dimension of the testing environment. It is clear that lines are a good obstacle representation for this type of indoor environment. As in most SLAM experiment, the robot assumes no previous knowledge of the environment, i.e. no map is provided initially.

Figure 8 shows the robot trajectory recorded by the odometry. The robot looped around the workspace twice before travelled back to near the starting point. Due to the accumulation of navigation error, the odometry becomes increasingly unreliable. The diagram also shows the EKF and SMC estimates for the robot position. When the EKF estimated trajectory is superimposed onto the actual room map, part of the trajectory
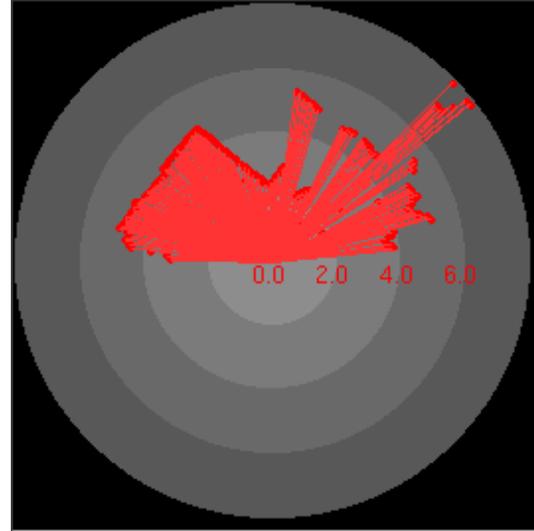

Figure 7: Typical LMS scan

collides with the existing obstacles. It is clear that the estimation provided by EKF SLAM is not as accurate as the SMC method in this case. Since the measurement of ground truth robot position is difficult to carry out in real time, the robot was stopped 7 times during the course to allow for manual position measurement. The localisation error is tabulated in Table 1.

Sample	Odometry Only(m)	SMC (m)	EKF (m)
1	0.00	0.01	0.01
34	0.36	0.38	0.42
133	1.05	0.98	1.07
177	1.18	0.19	1.26
206	1.52	0.35	0.53
229	1.37	0.41	0.36
313	2.51	0.86	0.93
406	2.87	0.19	2.19

Table 1: Localisation errors for different position estimation methods. Since SMC is a stochastic method, the algorithm was repeated 3 times. Only the means are shown here.

The proposed SMC method is more effective than EKF in correcting the odometry error. In SLAM operation, the number of obstacles detected L increases monotonically in time. In EKF, the estimates for extra obstacles are included to the same vector. The complexity is $O(L^3)$ due to the matrix inversion step. For the proposed SMC implementation, an extra filter is generated whenever a new obstacle is detected. Since the number of particles in the new filter is fixed, complexity is still $O(L)$. Although updating a large number of particles is a disadvantage computationally for smaller maps, the difference levels out in more complex environment. The number of obstacles detected is about 20 for the test environment. Using an Athlon XP
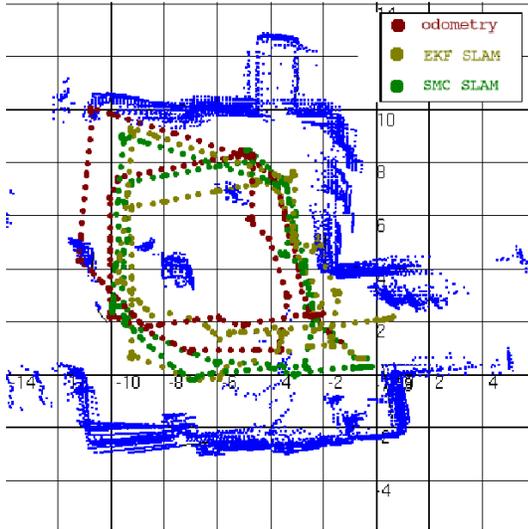
Figure 8: The robot trajectories given by the odometry, EKF SLAM and SMC SLAM. The blue points are the individual laser scanner range returns assumed to be taken at the SMC SLAM estimated position. Their spread is a good indication for the combined performance of the localisation and map-building operation.
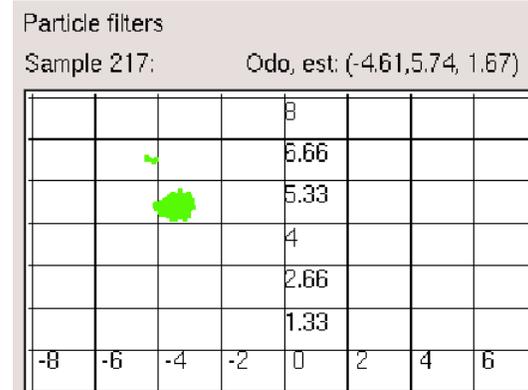

Figure 9: Particle distribution in the robot position estimation filter. Particles with a higher importance factor are represented in a darker colour. The distribution is clearly bimodal in this case.
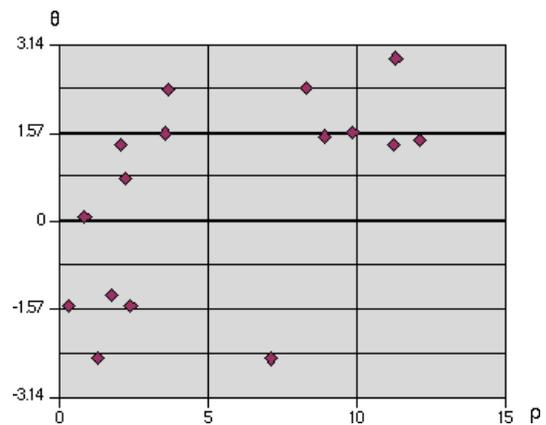
1600 PC, the average updating time for all filters is 1.2 s with the current SMC implementation. The computation speed reasonable for near real time robotic application. On the other hand, the running time might not be acceptable for real world environment if the data-sequence rather than feature based method was used.

The SMC implementation allows for a non-Gaussian error distribution. As shown in Figure 9, the particle distribution for the robot position estimation can be multimodal. The feature extracted from the laser scanner is often quite simple. Localisation sometimes needs measurements from more than a single step especially when similar views are present in the work space. The posterior distribution would appear as multimodal when the algorithm is still resolving the ambiguity. Assuming a Gaussian error distribution, as in the EKF approach, would force the estimation algorithm to choose the most likely position before sufficient evidence is collected and thus affects the robustness.

The proposed map building method stores the line parameters rather than line segment. The perceived internal map in (ρ, θ) space is shown in Figure 10. While it is sufficient for the robot localisation, extra parameters would have to be recorded if a floor plan-style map is desired.

Figure 10: The estimated obstacle line parameters shown in ρ, θ space.

5   Conclusions

The SMC methods do not rely on the Gaussian noise assumption and have found successful applications in many different robot localisation problems. SMC ap-

plication to the SLAM problem is however in an early stage of development.

This work describes a practical SMC implementation, which performs the SLAM operation in an environment filled with polygonal obstacles. A scanning range finder is assumed to be the main robot sensor. This algorithm adopts a multiple particle filter approach. New obstacles are detected after line features are extracted from the sensor data. Various stages of the algorithm are carefully designed so that standard SMC updating technique can be applied.

The experimental results shows that the localisation error is substantially reduced in the SMC implementation when compared with the established EKF method. The computation load is also acceptable for near real-time robot applications.

References

[Arulampalam *et al.*, 2002] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2), 2002.

[Dissanayake *et al.*, 2001] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[Doucet *et al.*, 2000] A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2000.

[Fox *et al.*, 1999] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artifical Intelligence*. AAAI, 1999.

[Fox *et al.*, 2003] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *Pervasive Computing*, pages 10–19, Jul–Sep 2003.

[Gordon *et al.*, 1993] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings-F Radar & Signal Processing*, 140(2):107–113, 1993.

[Guivant and Nebot, 2001] J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, 2001.

[McGinnity and Irwin, 2000] S. McGinnity and G.W. Irwin. *Sequential Monte Carlo Methods in Practice*, chapter 23. Springer, 2000.

[Murphy, 2000] K.P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing System*, volume 12, pages 1015–1021. MIT Press, 2000.

[Rot, 1991] G. Rot. Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38:123–127, 1991.

[Yuen and MacDonald, 2002] David C.K. Yuen and Bruce A. MacDonald. A comparison between EKF and sequential monte carlo techniques for simultaneous localisation and map-building. In *Australasian Conference on Robotics and Automation*, Auckland, New Zealand, 2002.

[Yuen and MacDonald, 2003] David C.K. Yuen and Bruce A. MacDonald. An evaluation of sequential monte carlo technique for simultaneous localization and map-building. In *IEEE International Conference on Robotics and Automation 2003*, Taipei, 2003.