UTILIZATION AND OPTIMIZATION FOR PARTICLE FILTERING MULTI-ROBOT SLAM

JUNPEI ZHONG

M.Phil

The Hong Kong Polytechnic University

2010

THE HONG KONG POLYTECHNIC UNIVERSITY

Department of Electrical Engineering

Utilization and Optimization for Particle Filtering Multi-Robot SLAM

by

Junpei Zhong

A thesis submitted in partial fulfillment of the requirements

for the degree of Master of Philosophy

Sep 2009

Declaration of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

Signed ____________________

Junpei Zhong

Affectionately Dedicated to

My Grandfather

Abstract

The utilization of multi-robot systems has a major advantage when comparing to single robot systems, for example, with multiple robots working together, it has the potential to accomplish a task faster than a single robot. However, when a team of robots is sharing the same worksite, the Simultaneously Localization and Mapping (SLAM) problem becomes much more difficult to resolve because a huge amount of information is needed to be processed as well as analyzed. But on the other hand, multi-robot SLAM can be more efficient if robots can exchange and share information regarding their sensed data properly.

In the SLAM problem, especially for Autonomous Underwater Vehicle (AUV) and Unmanned Aerial Vehicle (UAV), it is necessary to include non-linear and non-Gaussian parameters, for which the traditional Kalman Filter (KF) cannot yield ideal solution. In applications involving non-linear and non-Gaussian parameters, Particle Filters (PF), which are based on the concept of Monte Carlo simulation, are more suitable estimation techniques. However, in problems involving multiple dimensions, such as the multi-robot SLAM problem, when a huge number of particles are being used, two problems, namely particle impoverishment and sample size dependency, will occur during the particle updating stage and these problems will become more severe. The problems will reduce the accuracy of the estimation results and resampling algorithms, such as Sequential Importance Sampling, Stratified Resampling and Systematic Resampling are used to alleviate these two problems.

In this thesis, a novel PF algorithm for tackling the particle impoverishment and sample size dependency problems is being studied and its application in a multi-robot system is examined. In this algorithm, Ant Colony Optimization (ACO) is incorporated into the generic particle filter in order to drive the proposal distribution to approximate the optimal solution. Mathematical proof and results obtained from a single variable estimation problem as well as from the robot localization problem show that, after the ACO optimization, better proposal distribution and more accurate estimation results can be obtained.

In order to evaluate the performance of the ACO improved PF (PF_{ACO}) when applied to non-linear and non-Gaussian problems, such as the localization and SLAM problem, studies were conducted and utilization of PF_{ACO} algorithm for multi-robot systems was introduced. In a multi-robot environment, when two robots encounter, the same information on the same estimation problem represented by the two sets of particles will be re-evaluated based on information conveyed by particles from different sets. The particles are then merged into a single set and in such cases, parallel computing can be applied in order to reduce the processing time. By software simulation, our results are better than those from traditional approaches both in estimation error and execution time.

Publications Arising from the Thesis

- J.P. Zhong, Y. F. Fung. A Biological Inspired Improvement Strategy for Particle Filters. Proceedings of IEEE International Conference on Industrial Technology. 2009: 1-6.

- J.P. Zhong, Y. F. Fung, M. J. Dai. Ant Colony Optimization Assisted Particle Filters. International Journal of Control, Automation, and Systems. June, 2010.

- J.P. Zhong, Y.F.Fung. An Improved Particle Filter and its Application in Multi-robot SLAM. International Journal of Robotic Research: Special Issue on Stochastic Robotics. (Submitted in Jan 2010)

- J.P. Zhong, Y.F.Fung. A Proof and Case Study about PFACO Algorithm. Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews. (Submitted in May 2010)

Acknowledgement

First, I would like to thank my supervisor, Dr. Y.F. Fung, for his keen and thorough supervision – his detailed and discerning comments and criticisms on every stage of the work. From his guidance and advices, I believe it has been a good learning experience in my life during the M.Phil. degree.

Second, I would also like to show my gratitude to the Department of Electrical Engineering, the Hong Kong Polytechnic University provided financial support and good facilities during the research studies. Thanks to all lecturers and staffs in the department for giving me help, suggestions and inspirations.

Third, I would like to acknowledge the suggestions from Dr. X.Z. Zhang and Mr. M.J. Dai, and the Matlab simulation program originally developed from Dr. A. Vale.

Last but not least, I would like to thank all my family members in mainland China and Hong Kong for all their love throughout this process, and all my friends for their friendships and encouragement.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

s_t : the state vector describing the location and orientation of the vehicle at time t;

u_t : the control vector, applied at time t-1 to derive the vehicle to a state s_t at time t;

m: a vector describing the value of map grids which is assumed to be time invariant;

$\tilde{m}$: a vector describing the estimated value of map grids;

X_R : a vector describing the true robot position in axis X;

Y_R : a vector describing the true robot position in axis Y;

$\tilde{X}_R$: a vector describing the estimated robot position in axis X;

$\tilde{Y}_R$: a vector describing the estimated robot position in axis Y;

ω : a variable describing the robot angle;

$\tilde{\omega}$: a variable describing the estimated robot angle;

z_t : an observation taken by the robot of the environment at time t;

v_t^{tran} : transition model noise at time t;

$v_t^{measure}$: measurement model nose at time t;

f(): transition model;

h(): measurement model;

$s^t = \{s_{0,} s_1, ..., s_t\}$:the history of robot poses;

$u^t = \{u_1, u_2, ..., u_t\}$: the history of control inputs;

$z^t = \{z_1, z_2, ..., z_t\}$: the history of observations;

μ_t : Mean vector of multivariable Gaussian distribution;

Σ_t : Covariance matrix of multivariable Gaussian distribution;

$\tilde{s}_t$: approximation of state vector;

$\tilde{s}_t^+$: optimized approximation of state vector

$\tilde{z}_t$: approximation of measurement vector

$\tilde{z}_t^+$: optimized approximate measurement vector

$\hat{s}_t$: a posterior estimate of the state at step t

A : the Jacobian matrix of partial derivative of f with respect to x,

W : the Jacobian matrix of partial derivative of f with respect to w,

H : the Jacobian matrix of partial derivative of h with respect to x,

V : the Jacobian matrix of partial derivative of h with respect to v;

w(i) :weighting of particle i

M : number of particles before resampling

N : number of particles after resampling

$\tilde{M}$: current number of particles in residual resampling;

$\bar{M}$: remaining number of particles in residual resampling;

θ : a vector describing the location of landmark whose true location is assumed to be time invariant

S : a search space defined over a finite set of discrete decision variable,

Ω : a set of constraints among the variable,

$f : S \to R_0^+$ an objective function to be minimized in optimization problem;

η_{ij} : heuristic information corresponding to i and j;

τ_{ij} : pheromone corresponding to i and j;

d_{ij} : a distance corresponding to i and j;

H_s^k :search entropy of kth ant in sth search state;

n_r : the number of robots;

n_m : the number of map grids;

d: the number of dimension in SLAM;

$D(p \| q)$: K-L Divergence of q from p ;

α_m and β_m: constant indicating the relative accuracy of measurement in mapping correction;

Chapter 1 Robot, Multi-robot and SLAM

1.1 Robots

A robot is a virtual or mechanical artificial system. In practice, with dedicated electro-mechanical design, by its appearance or movements, it has intent or agency of its own to accomplish tasks. There is no consensus on what machines should be qualified as robots, but there is a general agreement among experts and the public that robots tend to do some or all of the followings: moving around, operating a mechanical limb, sensing and manipulating their environments, and exhibiting intelligent behavior, especially behavior which mimics humans or other animals, either mentally or physically.

At present, based on their applications, there exist two types of robots, namely general-purpose autonomous robots and dedicated robots. The general-purpose autonomous robots typically possess the ability to navigate autonomously within the environment, handling some basic tasks and communicate with human or other robots. General-purpose robots may perform a variety of functions simultaneously or they may take on different roles at different times of the day. Dedicated robots concentrate on a specific service or industrial tasks, such as car production, packaging and automated guided vehicles.

While most robots are being used in industry or at home, performing labor intensive or life saving jobs, new types of robots are under development in cutting-edge laboratories around the world and these include swarm robots [1, 2], nanorobots [3], reconfigurable robots[4], fully autonomous rescue robot [5, 6], etc.

1.2 Multi-robot System

The research of multi-robot system has grown very fast in recent years. It is probably due to the fact that researchers realize the multi-robot system has advantages over single robot system. First, multi-robot system which is composed of low-cost robots is more fault-tolerant than a single robot that has expensive equipment [7]. Also, it has the ability to finish complex tasks more efficiently than a single robot. For example multiple robots can estimate their position faster and more accurately due to their ability of exchanging information related to their positions faster whenever they sense each other[8]. Moreover, by scheduling the tasks cooperatively, less computational burden is assigned to each robot in a multi-robot system than given to a single robot by exchanging, broadcasting and integrating information among robots [9, 10].

1.3 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) is one of the active topics in robotic navigation research. Mobile robot system that equipped with SLAM solution as well as able to navigate autonomously is a very promising feature because they can be applied to explore environments where it is impossible for human beings to visit, such as underwater [11, 12], underground [13] or even in other planets, such as Mars [14] that are not suitable to carry-out long-term human activities due to safety concerns and cost effectiveness issues.

One of the basic prerequisites in robotic autonomous navigation is to endow robots with spatial memory as well as orientation and directional processing abilities that parallel those of human beings, therefore, they possess the ability to build a map and from this self-generated map to localize themselves. Moreover, the SLAM methods provide a mean through which a map of the environment can be built while at the same time providing an estimation of the location of the robot [15].

The key challenge of SLAM is to deal with various kinds of uncertainties, including noise, ambiguity, biases, and modeling errors. In [16], Brooks summarized this problem and suggested probabilistic approaches to solve it:

> *Mobile robots sense their environment and receive error laden readings. They try to move a certain distance and direction, only to do so approximately. Rather than try to engineer these problems away it may be possible, and may be necessary, to develop map making and navigation algorithm which explicitly represent these uncertainties, but still provide robust performance.*

In fact, numerous studies discovered that, by controlling statistical bounds on the estimated and observed environmental information, it is possible to achieve the approximation by balancing the uncertainties in SLAM due to system noise and over-confidence by relying on measured data which exceeds the objective accuracy of approximation.

Although many other localization and mapping methods exist, e.g. GPS, if the application is not an air-based system or when GPS signals cannot be received accurately, then establishing a solution for the SLAM problem becomes necessary [17, 18]. Dead reckoning is another process to estimate one's current position based on a previously determined position. However, even if we have motion measurement, without the observation information to correct the prior estimated pose and mapping and formulate the posterior values, it is no doubt that the estimated parameters will gradually deviate from the real one [19, 20]. Assuming a situation where we are operating in a GPS-denied environment and to solve the self-localization problem, it is very difficult to obtain a comparatively accurate result with only the motion measurements or even with a combination of Inertial Navigation System (INS) such as accelerometers, gyroscopes, wheel encoders, doppler, or image based pseudo-inertial measurements, since very small motion errors will propagate themselves forward in time and enlarging the error. This problem, along with the lack of state observability, will lead to a diverge in the estimation results; in other words the accuracy of the state estimation will quickly worsen [21, 22]. In certain situations many of these problems can be alleviated, or at least delayed, by enforcing certain constraints on portions of the navigation filter [22], which serves as statistical estimator based on data obtained various sensors.

Moreover, if we only attempt to build a map or localize a target using relative measurements, such as range and bearing information, it may not be possible to build a map without knowing the robot's location. In situations where good *priori* information is known, in regards to the robot's location, a solution to the map building problem can be obtained. However, as soon as the robot begins to move around, the uncertainty in its location will aggregate without bound for reasons that have been stated previously, so the map building result will degrade. Solving the SLAM problem is the logical resolution to these issues.

Another advantage for solving the SLAM problem is that it allows us to apply instruments which are light-weight and low-cost, as opposed to attempt to solve the mapping and localization problems separately which may in fact require more costly and complex instrumentations (e.g. GPS). When implemented correctly, a solution to the SLAM problem should involve a minimal amount of instrumentation. Additionally, solving the SLAM problem can be done in an autonomous fashion using online algorithms; therefore it can be anticipated that little or no user input will be required for utilizing a SLAM instrument suite.

Due to advantages produced by the SLAM technique when comparing to other localization and mapping methods, there exist applications which successfully adopt SLAM method to produce substantial improved results [23]. A number of successful experiments for indoor mobile robots have been performed under various indoor situations[22] and these robots implement some kinds of experimental SLAM algorithms, such as DP-SLAM that uses novel data structures to achieve real-time SLAM algorithm without landmarks [23, 24], 6D-SLAM which is capable of closing the loop in six-dimension with 3D scanners [25], and vSLAM (Visual SLAM) [26]. The literature [27] presented a comprehensive survey on various algorithms when applied to indoor mobile robot experiments. In addition, special outdoor applications can be found in undersea autonomous vehicles [27-29] , aerial unmanned vehicle [30] and robotic exploration of mines [31]; future application in space exploration [32] is also possible. Currently, in many of these applications, the SLAM solution is only used for navigation purpose, however some of those systems do make use of information obtained from the created map for other purpose [31].

In addition to robotics applications, a complete SLAM solution may help people to enhance their capabilities in localization and mapping, which is potentially a very powerful tool. In particular, it is hypothesized that the correct implementations of SLAM solutions combined with the proper instrument suite will enable one to design systems that could be used by humans, which operate autonomously. Instead of using the heavy and expensive instrument, such systems only equip light-weight and low-cost instrument that can be developed in future. One example of this can be used in military force to localize targets at a distance, to track future troop movements and real-time mapping. This scenario filled by SLAM based systems would be for environments in which GPS signals either are not accurate enough, or are not dependable, such as the urban canyon or indoor environments [33]. Similar SLAM systems for non-military applications may also exist in areas of human space planet exploration, as well as in terrestrial applications such as for the use of search and recovery teams in situations when GPS signals are not accessible or are not too accurate.

1.4 SLAM Solutions

A common scenario adopted in SLAM application involves a robot traveling in an area giving neither the absolute position of the robot nor an accurate map of the environment. In order to estimate a consistent map, the robot has to process information related to its surrounding features and information (e.g. heading and position information). To reduce the error, in this process, these relative observations of features of the surrounding are used to jointly estimate the environment by inferring the position and heading, associated with the previously stored information (called *prior*). To accomplish this joint estimation, there are many SLAM algorithms that exist in the literature, two main branches of which are the Kalman Filters and Particle Filters.

1.4.1 Kalman Filters

The first SLAM solution is based on the Kalman Filter concept. As an recursive filter, the Kalman Filter is an optimal Bayesian estimator that operates strictly based on the assumptions that the system can be estimated by a Gaussian posterior probability distribution together with a linear motion and measurement model [34, 35]. Researchers later derived a technique that the nonlinear motion and measurement model can approximately assume to be linear, which resulted in the Extended Kalman Filter that is an analytical approximation of the Bayes' filter. This recursive solution provided by the extended Kalman Filter is sufficient if the posterior probability distribution for SLAM states can be adequately characterized by a uni-modal Gaussian distribution [36].

1.4.2 Particle Filters

Other approaches are based on the concept of Monte Carlo Methods, which has the ability to present posterior probability with a large number of discrete, weighted samples[37]. In the realization of SLAM, each sample (particle) is a hypothesis of the posterior (robot pose and the corresponding set of landmarks) that is propagated according to a motion model and then evaluates the fitness of this hypothesis relative to the target distribution [38], which mostly related to measurement information. From this particle-based approach, some algorithms are successful including the DP-SLAM [23, 24] and FastSLAM [39], as well as methods that attempt to describe the geometry of objects in the map with various methodologies [40, 41]. Unfortunately, many of these methodologies have serious limitations. For example, the DP-SLAM algorithm is formulated for use with a specific type of measurement device. In the cases that attempt to describe the complex geometries of the environment, it is difficult to compute a solution in real-time especially for higher-dimensional or geometrically complex situations, such as with numerous landmarks of various shapes.

1.4.3 Particle Filtering SLAM

In the following, we will briefly outline Particle Filters (PF) adopted in SLAM application, while details of the particle filter will be introduced in Chapter 2. PF typically draws a new set of particles after weights have been assigned. In PF, uncertainty of the state is stored in dispersion of these uniformly weighted samples; a broader spread implies more uncertain estimation. Consequently, multi-modal distributions from state constraints or nonlinear propagation can be easily approximated. However, too many particles having small weight will decrease the efficiency of particle representation, which means that more particles will be needed to construct the distribution. To avoid the problem, the hypotheses are compared with a feature observation, and those particles which are consistent with the observation are given larger weights and duplicated, that is, the resampling process. After resampling, particles have smaller weights are likely to be eliminated. Surviving particles are then propagated according to motion control information at the next time step, and these processes are being repeated.

Advantages of the PF based SLAM include the ability to represent an arbitrarily complicated posterior distribution of the robot pose, as well as many independent estimates of the map. Additionally, PF converge to any distributions with infinite particles [37] so that theoretically it will achieve the optimal posterior solution. As computational power increases, estimators based on PF will only improve their characterization of the posterior. Unlike the Kalman Filters, the computational complexity of the PF scales linearly with the dimension of the state, allowing favorable application in online SLAM applications. Furthermore, the most important features are that the PF based SLAM algorithms have demonstrated solutions to two unsolvable problems in SLAM: global localization and the kidnapped robot problem [42]. Both problems utilize the multiple hypothesis nature of PF to determine true position under the initial global uncertainty.

Although there are numerous advantages in PF SLAM, it also includes certain disadvantages that are difficult to overcome especially the particle impoverishment and particle size dependency problem. When particles are evenly distributed but do not match the likelihood distribution in a plausible manner, the efficient sample size is decreased, and therefore, the particle impoverishment problem occurs [37]. As a result, intuitively, this PF application needs more sample to generate the posterior, resulting in particle size dependency. The two problems are somehow related to the failure that the proposal distribution (in most cases characterized by the transition model) is not able to meet the target distribution (from observation), usually derived from an accurate sensor measurement[37]. This scenario is becoming increasingly relevant to the current trends in inertial systems produce smaller, chip-based accelerometers and gyros [43]. In these cases, an extremely accurate feature observation device would be ideal, since, hypothetically, there are an infinite number of particles preserved in the algorithm. Practical implementations are restricted to a finite number of particles, the problem becomes more severe especially in higher dimension problems, such as in multi-robot SLAM, which causes the estimated dimension to be very high compared to the particle size [44].

1.5 Major Contributions

To avoid the problems in PF, the basic idea is to duplicate particles with larger weights and eliminate the low-weighted ones; this is achieved by the re-sampling process. Previous researchers show numerous resampling methods [45-47]. However, they become useless when applied to the multi-robot SLAM application because of the fact that it is a high dimension estimation problem. Furthermore, these methods are not promising enough because the copied samples are no longer statistically independent after resampling so the previous convergence result will be lost. It is called losing sampling diversity [48]. This thesis proposes a novel method to optimize the particle distribution which meets the following requirements:

- The Particle Impoverishment and Particle Size Dependency problem can be avoided in multi-robot SLAM application.

- This kind of PF is suitable to distributed implementation in a multi-robot processing system, e.g. it may be computed in parallel.

1.6 Thesis Outline

In Chapter 2, the concept of PF will be discussed. They are considered to be form of sequential methods of Monte Carlo Simulation, using a set of random samples to draw the posterior density function of a Bayesian Estimation. Furthermore, some improved PF methods, as well as their significances and drawbacks, are discussed.

In Chapter 3, we introduce a biologically inspired method namely the Ant Colony Optimization method. This method mathematically simulates the nature behavior of ants; they can always travel along the shortest path between their food and colony, because they leave a matter called pheromone along their paths. In recent years, this phenomenon is noticed by scientists and, accordingly, created such an optimization method – the ACO.

In Chapter 4, we will introduce an improved Particle Filter method, which possibly avoid the PF limitations and also is suitable for implementation in a multi-robot system. Considering the requirements, we apply ACO into PF to re-arrange the particles so that the proposal distribution is similar to the target distribution.

In Chapter 5, we present mathematical and experimental proofs of the ACO improved PF. In the mathematical proof, the K-L Divergence between the proposal distribution and the target distribution will show that ACO actually has the ability to optimize the particle distribution. In addition, various PFs are employed to accomplish the same estimation task in order to compare their performances.

In Chapter 6, utilization of the ACO improved PF on a multi-robot system is discussed. The basic idea of the system is as follows: before the encounter between robots, PFs are running independently in each robot. As soon as the robots encounter, the localization PF still runs separately, while the mapping task can be achieved with two approaches based on our requirements. In the efficient approach the mapping estimation is assigned to robots; in the accurate approach, mapping estimation is running separately in each robot, and then, combination and correction of all estimated maps is completed to build an accurate map.

In Chapter 7, techniques presented in previous chapters are implemented in a Matlab simulation platform. To further compare the performance of ACO improved PF, results of other estimation methods are also presented. Finally, Chapter 8 states the main conclusions and further direction for PF based SLAM research.

Chapter 2 SLAM and its Particle Filtering Implementation

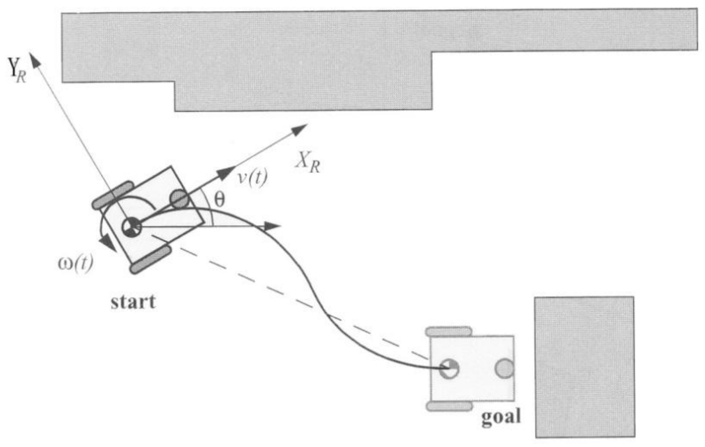
2.1 Introduction

Solving the Simultaneous Localization and Mapping (SLAM) problem is the key to implement autonomous robot systems because the SLAM solution builds the surrounding map and pin-points the location of the robot in the map which is necessary information for mobile robot navigation. In addition, the possibility to explore the surrounding and accomplish tasks within an estimated environment can also be achieved if a map is available. Applications with full SLAM solution is necessary in situations when global measurements, such as GPS, are not available, for instance in applications where a robot operates indoors or in an urban canyon. However, in SLAM, small errors due to signal noises from various sources including sensors and actuators will aggregate. In such situations, uncertainties related to locations of objects in the environment as well as poses of robots become tightly linked. This is due to the fact that in these situations, the robot must rely on relative measurements related to its current pose. To solve this probabilistic problem, SLAM researchers mainly focus on the probabilistic derivation of the general Bayes' Filters [49], and their solution based on Extended Kalman Filters (EKF), Particle Filters (PF) and their descendants.

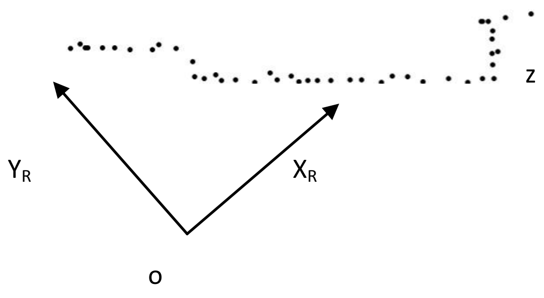
This chapter begins with a review of the Bayes' Filter and its application to the SLAM problem. It is followed with a brief introduction to the Kalman Filters, and PF, for which an in-depth discussion is given. In addition, some improved PF algorithms are also introduced.

2.2 SLAM Fundamentals

As shown in Figure 2.1, we consider a mobile robot moving through an environment taking relative observations using a noisy sensor.



(a)



(b)

Figure 2.1 Demonstrations of Mobile Robot navigation and its sensor measurement

(a) The mobile robot is navigating in the environment, the robot state includes $\{X_R, Y_R, \omega\}$ *; (b) Measurement from the robot, the grid-occupancy information from the sensor is regarded as relative observation* $\{z\}$ *so that the state* $\{X_R, Y_R, \omega\}$ *can be estimated.*

As mentioned in Chapter 1, the robotic SLAM problem can be treated as a probabilistic problem, estimating the robot's location as well as parameters of the map are based on the maximum probability derived from the given measurements and inputs. This estimation problem can be related to a Hidden Markov Model (HMM), as shown in Figure 2.2, which is to estimate the single variable state given the measurement.

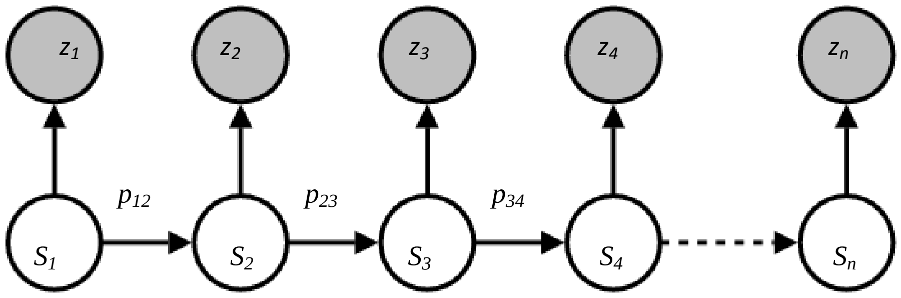
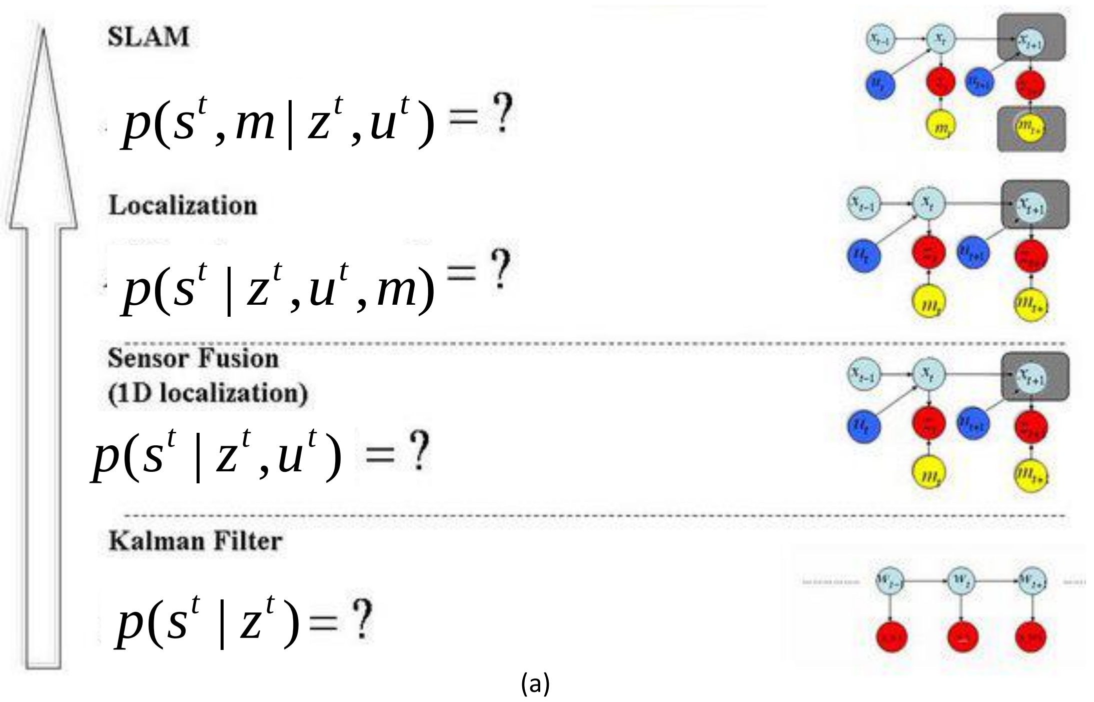


Figure 2.2 Hidden Markov Model

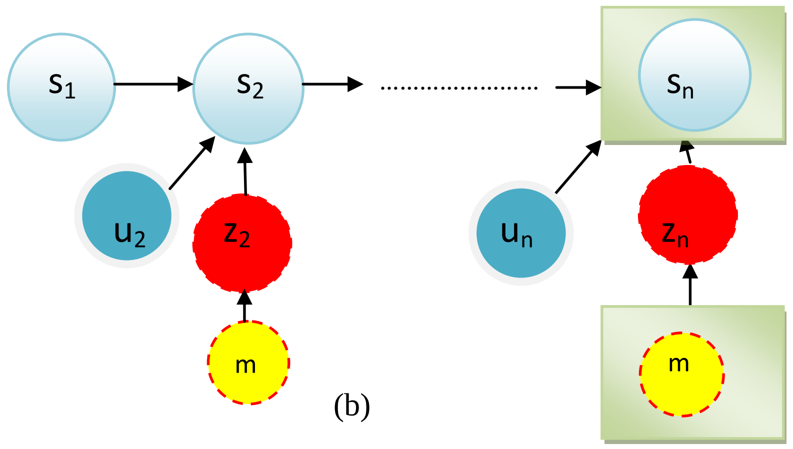
In the second row, each shape represents a random variable that can adopt any of a number of values. The random variable s_t is the hidden state at time t (with the model from the above diagram, $s_t \in \{s_1, s_2, s_3, ..., s_n\}$. The random variable z_t is the observation at time t, $z_t \in \{z_1, z_2, z_3, ..., z_n\}$. The arrows in the diagram denote conditional dependencies representing the probability $\{p_{12}, p_{23}, p_{34}, \ldots\}$ between states.

From Figure 2.2, it is clear that the conditional probability distribution of the hidden variable s_t at time *t*, given the value of the hidden variable s_{t-1}, depends *only* on the value of the hidden variable s_{t-1}; the values at time $t-2$ and before have no influence. This is called the Markov property. Similarly, the value of the observed variable z_t only depends on the value of the hidden variable s_t (both at time *t*).

We can augment this Hidden Markov Model (HMM) idea to a 1D-Estimation, Localization and SLAM as shown in Figure 2.3. In the upper part of this figure, more unknown variables are introduced, which can be regarded as a special form of HMM model. In order to solve this problem based on the HMM model, usually we employ the Bayes' Filter.


SLAM
$p(s^t, m \mid z^t, u^t) = ?$
Localization
$p(s^t \mid z^t, u^t, m) = ?$
Sensor Fusion
(1D localization)
$p(s^t \mid z^t, u^t) = ?$
Kalman Filter
$p(s^t \mid z^t) = ?$

(a)



(b)

Figure 2.3 A Dynamic Bayesian Network in SLAM and different problems of mobile robot application

(a) From the basic HMM model, to SLAM problem. More states and measurements are added into the network. (b) SLAM can be regarded as the special case of HMM to estimate two unknown variables: the robot state and the map.

In probabilistic form, the SLAM posterior is represented as:

$$p(s^t, m \mid z^t, u^t) \qquad (2.1)$$

where s^t denotes the history of the state vector from time 0 to t: $\{s_0, s_1, s_2, \ldots, s_t\}$, z^t denotes the history of measurement from time 1 to t, u^t denotes the history of control vector, and *m* denotes a vector describing the value of map grids which is assumed to be time invariant. This probability distribution describes the joint posterior density of map (either in grid-occupancy, landmark or topology representation) and robot state (from time 1 to *t*) given the recorded observations and control inputs up to and including time *t*. In general, a recursive solution to the SLAM problem is applied. Starting with an estimate for the distribution $p(s_{t-1}, m \mid z^{t-1}, u^{t-1})$ at time *t-1*, the joint posterior, following a control u_t and observation z_t, is computed using Bayes' theorem [42] . This computation requires that a state transition model and an observation model are defined describing the effect of the control input and observation respectively.

The transition (motion) model for the robot can be described in terms of a probability distribution on state transitions in the form given in Equation 2.2.

$$p(s_t \mid s_{t-1}, u_t) \qquad (2.2)$$

That is, the state transition is assumed to be a Markov process which assumes that the next state s_t depends on the immediately preceding state s_{t-1} and the applied control u_t and is independent of the observations (z_t) and the map (*m*).

The observation model describes the probability of making an observation z_t when the robot location and map information are known and is generally described in the form represented by Equation 2.3.

$$p(z_t \mid s_t, m) \qquad (2.3)$$

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

The SLAM algorithm is now implemented in a two-step recursive (sequential) form with prediction (time-update) and correction (measurement-update).

Time-update (Prediction)

$$p(s_t, m \mid z^{t-1}, u^t) = \int p(s_t \mid s_{t-1}, u_t)\, p(s_{t-1}, m \mid z^{t-1}, u^{t-1})\, ds_{t-1} \qquad (2.4)$$

Measurement Update (Correction)

$$p(s_t, m \mid z^t, u^t) = \frac{p(z_t \mid s_t, m)\, p(s_t, m \mid z^{t-1}, u^t)}{p(z_t \mid z^{t-1}, u^t)} \qquad (2.5)$$

Equations 2.4 and 2.5 provide a recursive procedure for calculating the joint posterior $p(s^t, m \mid z^t, u^t)$ for the robot state s_t and the map m at time t based on all observation z^t and all control input u^t up to and including time t. The recursion also is a function of a motion model (2.2) and an observation model (2.3).

The above two steps formulates the Bayes' Filter by Bayes theorem:

$$p(s_t, m \mid z^t, u^t) = \eta\, p(z_t \mid s_t, m) \int p(s_t \mid s_{t-1}, u_t)\, p(s_{t-1}, m \mid z^{t-1}, u^{t-1})\, ds_{t-1} \qquad (2.6)$$

where η is a proportionality constant. This is only an optimal solution in theory and it can never be obtained analytically because the integration in Equation 2.4 is usually intractable.

It is worth noting that the map building problem may be formulated as computing the conditional density $p(m \mid s^t, z^t, u^t)$. This assumes that the location of the robot s_t is known (or at least deterministic) at all times, subject to the information of initial location. A map *m* is then constructed by fusing observations from different locations. Conversely, the localization problem may be formulated as computing the probability distribution $p(s_t \mid z^t, u^t, m)$. This assumes that the map information is known with certainty, and the objective is to compute an estimate of the robot location relative to the map. Therefore, the computation of SLAM joint probability becomes a mutual calculation process, that is, we first estimate the robot state, construct the surrounding estimation map based on previous state, and then compute the new state again after the movement and prediction.

Note that in the filter discussed in the following chapters, the map (variable *m*) is represented by the grid-occupancy method[50], which divides the map into several grids (Figure 2.4). Each grid can be represented as obstacle or space. Although it has larger dimension number than the landmark based method [51], it is much more accurate and practical than the landmark presentation because it is not always possible to define a landmark in a real application. The grids-representation is an approximation to the reality but it is sufficient for purposes of algorithm development being discussed. The error due to the approximation can, for the most part, be absorbed into the measurement error as well as into uncertainty in grid map locations so it has the ability to achieve the map accuracy requirements.

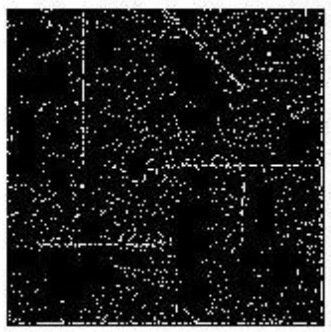


Figure 2.4 Example of grid-occupancy map

The measurement model usually includes a model of the physical measurements, as well as the noise characteristics of these measurements. To be consistent with what is used throughout the analysis given in the thesis, range and bearing measurements pairs will be considered as simultaneous measurements of the environment, that is, both sets of data from two measurements will affect the SLAM result in the same time-step. This is a reasonable assumption for a number of instruments that are used in SLAM implementations, such as the SICK laser.

Solutions to the probabilistic SLAM problem involve finding an appropriate representation for both the motion model (Equation 2.2) and observation model (Equation 2.3) that allow efficient and consistent computation of the prior and posterior distribution in Equation 2.4 and 2.5. Until now, the most common representation is in the form of a state-space model with additive Gaussian noise, leading to the application of the Extended Kalman Filter [52-56]. An alternative method is to describe the robot motion model given in Equation 2.2 with a set of samples to demonstrate either a Gaussian or a non-Gaussian probability distribution. This leads to the use of PF. These methods are being discussed in the following sections.

2.3 Extended Kalman Filter

Though difficult or impossible to compute in closed-form, Equation 2.6 can be approximated by restricting the SLAM posterior to a Gaussian probability density function (*pdf*). When a motion and observation models can be regarded as a linear function, together with noise coming from a Gaussian distribution, it is possible to apply Kalman Filter to this recursive process to obtain the optimal Bayesian posterior [52-56] . The SLAM problem can be interpreted as a state space system, and therefore it is suitable to be solved by the Kalman Filter framework [33], although in many real-life applications there are often non-linearity involved.

To overcome this drawback, the linearization of non-linear motion and measurement models forms the basis of the Extended Kalman Filter (EKF), an analytic approximation of the optimal filter for non-linear situations [52]. This formulation is essentially the same as the traditional Kalman Filter, but the linearization performed allows for its compliable to the real world's system dynamics and the measurement models.

The EKF formulation is based on three assumptions:

- Firstly it assumes the posterior is a Gaussian distribution, which is described as $p(s)$.
- Secondly, noises in the motion model and the environmental measurements are assumed to be noise with Gaussian distribution.
- Thirdly, it also assumes that the substantial errors are not incurred by making use of linear approximations of the dynamics of the state propagation in time, as well as in the physics of states relationships for environmental measurements.

In EKF, the mean posterior contains robot pose information (2D or 3D position and heading) and the mean position estimate for each mapped environment. To further adjust the estimation, the state covariance and pair-wise correlations between states are stored in the filter covariance matrix, where the variances which lie on the diagonal as the a priori knowledge with regard to the uncertainty in the individual states.

As a nonlinear version of Kalman Filter (KF), EKF represent the SLAM posterior distribution as a high-dimensional multivariable Gaussian parameterized by a mean μ_t and covariance Σ_t for each state. Compared to KF, the state transition and observation models need not be linear function but may instead be differentiable functions given in Equations 2.7 and 2.8:

$$s_t = f(s_{t-1}, u_t) + v_t^{tran} \quad (2.7)$$

$$z_t = h(s_t) + v_t^{measure} \quad (2.8)$$

Where v_t^{tran} and $v_t^{measure}$ are the noises which are assumed to be zero mean white noises. The function *f* can be used to compute the predicted state from the previous estimate and similarly the function *h* can be used to compute the predicted measurement from the predicted state, both of which are nonlinear functions. However, *f* and *h* cannot be applied to the covariance directly as KF does. Instead, a matrix of partial derivatives (called the Jacobian) is computed. At each time step the Jacobian is evaluated with current predicted states. These matrices can be used in the Kalman filter equations. This process essentially linearizes the non-linear function around the current estimate.

The main procedure of EKF can be illustrated by the following flow-chart, Figure 2.5.

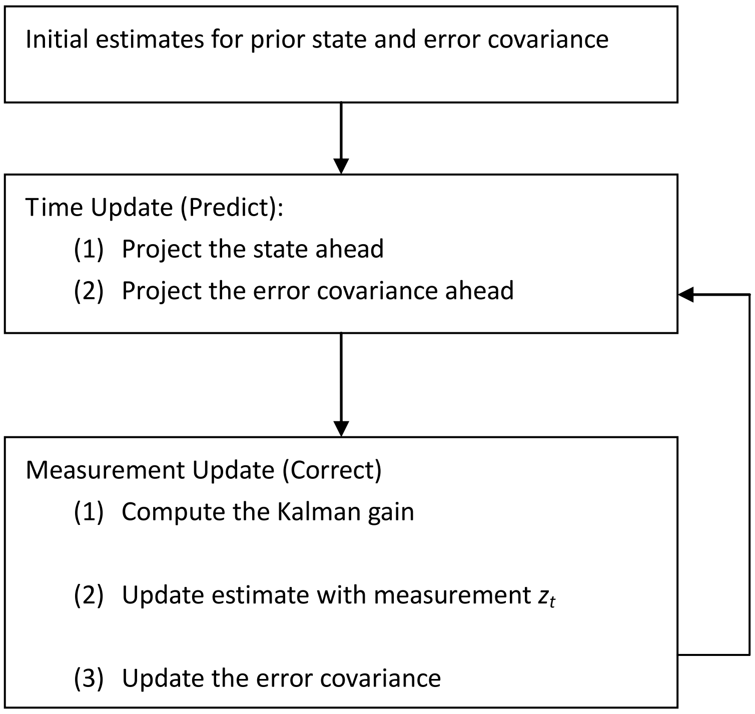

Initial estimates for prior state and error covariance
Time Update (Predict):
(1) Project the state ahead
(2) Project the error covariance ahead
Measurement Update (Correct)
(1) Compute the Kalman gain
(2) Update estimate with measurement z_t
(3) Update the error covariance

Figure 2.5 Flow-chart of EKF

The detailed derivation of EKF is omitted in this section, which may be found in various literatures[54, 55]. Referring to the time update step in Figure 2.5, the state and covariance are estimated from the previous time step *t-1* to the current time step *t*. The measurement updates equations correct the state and covariance estimates with the measurement z_t.

An important feature of EKF is that the Jacobian H_t in the equations for the Kalman gain K_t serves to correctly propagate or "magnify" only the relevant component of the measurement information. For example, if there is not a one-to-one mapping between z_t and the state via *h*, the Jacobian H_t affects the Kalman gain so that it only magnifies the portion of the residual $z_t - h(\hat{s}_t^-)$ that does not affect the state.

One disadvantage of the basic EKF is the growth in complexity of the filter with the dimension of state-space in its map. This is primarily due to the fact that this method, in its original formulation, relies on a covariance matrix of size $O(N^2)$, where *N* is the number of estimation states. Another drawback of the basic EKF SLAM algorithm is the single-hypotheses data association [57]. It is a decision-making process in which an incoming measurement is either matched with an existing landmark in the filter map or deemed as a new feature. This decision process is not a trivial task in SLAM applications, where the pose and map uncertainty and measurement noise can all contribute to data association failure, and then can induce estimate divergence [51]. In the basic EKF framework, the filter must pick one association (e.g. a feature in landmark representation, or a grid in the grid-occupancy representation) for reference, typically with a maximum likelihood heuristic, and the effects of an incorrect decision probably will lead to the divergence.

However, there has been significant effort on the improvement of EKF in recent years. One method which has been shown to resolve the complexity issues surrounding the use of EKF formulations and is considered as the state-of-the-art by many researchers, is known as *Atlas*[58]. This method assumes constant time, or at least bounded time, operation and is therefore not dependent on the size of the map. Moreover, it should be noted that the use of this *Atlas* method is not limited to EKF solutions of the SLAM problem, because it is a general framework to reduce complexity, rather than a filtering method itself. There are other EKF related methods which also claim either constant time or improved time operation and have experimentally shown to produce good results, e.g. Sparse Extended Information Filters (SEIF) [59, 60] and compressed EKFs [61, 62].

2.4 Particle Filters

EKF method solves the non-Gaussian estimation problem by linearizing the non-linear function around the current estimate, which is the first order Taylor expansion of the nonlinear functions. Successful attempts at solving the SLAM posterior without restraining its form to Gaussian distribution employ a more recent estimation tool called Particle Filters (PF). They are a large class of Monte Carlo estimation methods used in partially observable Markov chains[42]. First proposed in the 1950s [63], the PF has recently enjoyed attention while the enhancements in applied statistics and computer processing speeds have prompted its application to a broader range of estimation problems [46, 52, 64]. Improvements to the basic PF techniques by Gordon [46], Liu and Chen[65] in the mid-to-late 1990s have produced recursive Bayesian estimators with established theoretical convergence that are no longer bound to the Gaussian assumption that are imposed on the Kalman Filter and its derivatives. Furthermore, the PF are credited with having solved the global localization and the kidnapped robot problem [43], both of which were previously unsolved and considered to be crucial for a robust mobile robot application[66].

In this section, we ignore the specific application of SLAM and concentrate on the general framework of PF (or the single variable estimation problem). The Bayes' Filter recursion in Section 2.6 becomes the following:

$$p(s_t | z^t, u^t) = \eta p(z_t | s_t) \int p(s_t | s_{t-1}, u_t) p(s_{t-1} | z^{t-1}, u^{t-1}) ds_{t-1} \qquad (2.9)$$

In order to simplify the following discussion, comparing to the SLAM Bayes' Filter in Equation 2.6, the above equation does not include the parameter of the map (*m*).

Generally, the PF algorithm is illustrated in the following pseudo-codes.

```
Algorithm 2.1: The generic PF
$[\{s_t(i), w_t(i)\}_{i=1}^N] = PF[\{x_{t-1}(i), w_{t-1}(i)\}_{i=1}^N, z_t]$
Initialization: Generate particle samples $\{s_0(i), w_0(i)\}_{i=1}^N$
Prediction:
For i =1:M
        • Predict $s_t(i) \sim q(s_t(i) | s_{t-1}(i), z_t)$
        • Assign the particle a weight
End For
Measurement update
Calculate total weight: $t = sum[\{w_t(i)\}_{i=1}^M]$
For i=1:M
        • Normalize: $w_t(i) = t^{-1} w_t(i)$
End For
Resampling
```

The above estimation is shown in the following detailed recursive algorithm:

1. Propagation Step:

Estimate the new pose $s_t(i)$ from $p(s_t \mid s_{t-1}, u_{t-1})$, which propagating each s_{t-1} using independent samples with u_{t-1}, which is defined by a motion model.

$$s_t(i) = f(s_{t-1}(i), u_t \qquad (2.10)$$

where u_t represents the input motion information, however in this case the input noise can be represented by any probability distribution that can be sampled from. This process approximates the following predictive density:

$$p(s_t \mid s_{t-1}, u_{t-1})\, p(s_{t-1} \mid z^{t-1}, u^{t-1}) \qquad (2.11)$$

2. Measurement Step:

Particles are drawn from a proposal distribution $q(s_t)$ according to a Sequential Monte Carlo technique known as importance sampling [46, 52]. Weights are then assigned to the particles based on:

$$p(s_t \mid z_t, u_t) = \sum_{i=1}^{M} q(s(i))\, w(\qquad (2.12)$$

where w is a set of importance weights given by the ratio of the target (posterior) distribution to the proposal distribution:

$$\tilde{w}_t(i) = \frac{p(s_t \mid z^t, u^t}{q(s_t(i))} \qquad (2.13)$$

and then normalized according to:

$$w(i) = \frac{\tilde{w}(i))}{\sum_{j=1}^{M} \tilde{w}(j))} \qquad (2.14)$$

where M is the total number of particles used to represent the distribution. The original PF uses the motion model $p(s_t \mid s_{t-1}, u_t)$ as the proposal distribution, so the assigned weighting factor becomes:

$$\tilde{w}_t(i) = p(z_t \mid s_t \qquad (2.15)$$

which in SLAM applications is the robot observation or perceptual likelihood [32, 43] . A detailed derivation of the PF can be found in [67]. Applying this principle to the recursive Bayesian framework is resulted in the sequential importance sampling.

3. Re-sampling Step:

Let us discuss the situation when particle impoverishment occurs. If we take samples from the posterior and proposal distribution, we can observe that weights of samples which are far away from the true value will decrease during the iterations according to Equation 2.13, while just a small number of particles can maintain their weights.

In order to illustrate the weight function Equation 2.13 is being expanded into the following:

$$\begin{aligned} w_t(i) &= \frac{p(z^t \mid s^t)\, p(s^t)}{q(s^t(i) \mid z^t)} \\ &= \frac{p(z^t, s^t)}{q(s^t(i) \mid z^t)} \\ &= \frac{p(s^t \mid z^t)\, p(z^t)}{q(s^t(i) \mid z^t)} \\ &\propto \frac{p(s^t \mid z^t)}{q(s^t(i) \mid z^t)} \end{aligned} \tag{2.16}$$

The ratio in the last line of the above equation is called the importance ratio. We can see that samples $q(s^t \mid z^t)$ which are far away from the numerator $p(s^t \mid z^t)$ will decrease its corresponding weight values in the iterations based on Equation 2.16, while just a small number of particles can maintain their weights. This iteration runs over time, so the variance of the whole set of particles increases. A detailed proof can be found in [68]. Therefore, the unconditional variance (that is, when the observations are regarded as random) of the importance ratios increases over time. [69, 70]

In that case, we want the proposal density to be very close to the posterior density. When this happens, we obtain the following results for the mean and variance [66].

$$E_{q(\cdot|z^t)}\left(\frac{p(s^t|z^t)}{q(s^t|z^t)}\right) = \qquad (2.17)$$

and

$$\mathrm{var}_{q(\cdot|z^t)}\left(\frac{p(s^t|z^t)}{q(s^t|z^t)}\right) = E_{q(\cdot|z^t)}\left(\left(\frac{p(s^t|z^t)}{q(s^t|z^t)} - 1\right)^2\right) \qquad (2.18)$$

An increase in the variance has an adverse effect on the accuracy of the Monte Carlo simulations [71]. In other words, the variance should be close to zero in order to obtain reasonable estimations. In practice, the degeneracy caused by the increase in variance can be observed by monitoring the importance weights. Typically, what can be observed is that, after a few iterations, one of the normalized importance weights approaches to 1, while the remaining weights tend to zero. It is called particle impoverishment or particle degeneracy.

To measure the degeneracy of the PF from the particle weights, the effective sample set size M_{eff} is introduced in [72] [47], and it is defined as:

$$M_{eff} = \frac{M_s}{1 + \mathrm{var}(w_t^*(i))} \qquad (2.19)$$

where $w_t^*(i) = p(s_t(i)|z^t) / q(s_t(i)|s_{t-1}(i), z_t)$ is referred to the “true weight”. Although this cannot be computed determinably, an estimate of $\widehat{M}_{eff}$ of M_{eff} can be obtained by Equation 2.20:

$$\widehat{M}_{eff} = \frac{1}{\sum_{i=1}^{M_s}(w_t(i))^2} \qquad (2.20)$$

where $w_t(i)$ is the normalized weight. Notice that $M_{eff} \le M_s$ and small M_{eff} infers that severe degeneracy happens. According to [73-75], the optimal proposal distribution function that minimizes the variance of the true weights $w_t^*(i)$ conditioned on $s_{t-1}(i)$ and z_t.

$$
\begin{aligned}
q(s_t \mid s_{t-1}(i), z_t)_{opt} &= p(s_t \mid s_{t-1}(i), z_t) \\
&= \frac{p(z_t \mid s_t, s_{t-1}(i)\, p(s_t \mid s_{t-1}(i))}{p(z_t \mid s_{t-1}(i))}
\end{aligned}
$$

(2.21)

Related to the definition of weight, we yield the optimal weighting as Equation 2.22:

$$
\begin{aligned}
w_t(i) \propto w_{t-1}(i) &\frac{p(z_t \mid s_t(i))\, p(s_t(i) \mid s_{t-1}(i))}{q(s_t(i) \mid s_{t-1}(i), z_t)} \\
&= w_{t-1}(i)\, p(z_t \mid s_{t-1}(i)) \\
&= w_{t-1}(i) \int_i p(z_t \mid s_t(i))\, p(s_t(i) \mid s_{t-1}(i)) ds_t
\end{aligned}
$$

(2.22)

From the above equation, we can see that the choice of proposal distribution is optimal since for a given $s_{t-1}(i)$, $w_k(i)$ takes the same value (Equation 2.21), whatever sample is drawn from $q(s_t \mid s_{t-1}(i), z_t)_{opt}$. Hence, conditional on $s_{t-1}(i)$, $\mathrm{var}(w_t(i)) = 0$. This is the variance of the different $w_t(i)$ resulting from different sampled $s_t(i)$.

However, in real applications, it may not be so straightforward to achieve the optimal proposal distribution ($p(s_t \mid s_{t-1}, z_t)$), because it requires the ability to sample from $q(s_t \mid s_t(i), z_t)$ and to evaluate the integral over the new state. Using a non-optimal proposal distribution in PF, a large number of samples are thus effectively removed from the sample set because their importance weights become numerically insignificant. That is the reason why resampling scheme is an important research topic related to PF applications.

Nevertheless, the importance sampling approximation depends on how close the proposal distribution is to the target distribution because it operates based on their weights. As illustrated in Figure 2.6, if the higher likelihood area is too narrow or if there is little overlap between the prior and the likelihood, one needs to move the samples to regions of high likelihood. Various improved resampling approaches have been proposed to solve this problem.

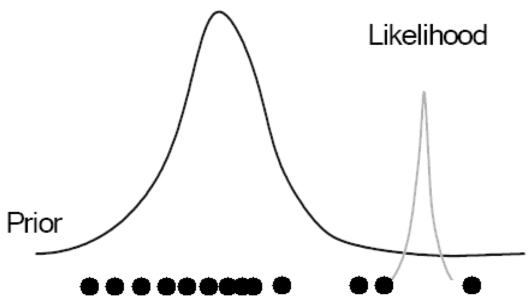


Figure 2.6 A scenario of particle impoverishment

When new measurements (i.e. the likelihood) appear in the tail of the prior, the particles predicted from the prior density will distribute far from the likelihood.

To reallocate computational resources and obtain a more accurate distribution, resampling the particles is necessary. Initially proposed by Gordon et al. [46], this resampling technique, known as Sampling Importance Resampling (SIR) or Bootstrap Filtering, produced the first effective PF [46, 76]. This recursion, is illustrated in Figure 2.7, makes the particles with higher weights reproduce more in the next time step while those with the smaller weights eliminated. This approach optimally will reach the Bayesian posterior in the limit of infinite particles [63, 77].

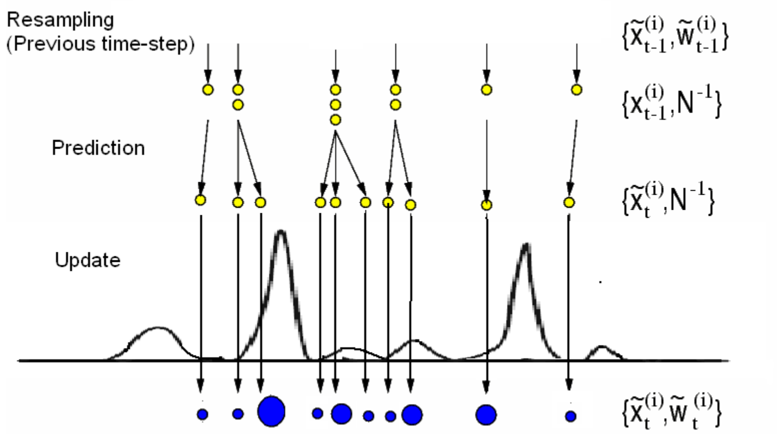

Resampling
(Previous time-step)
$\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}$
$\{x_{t-1}^{(i)}, N^{-1}\}$
Prediction
$\{\tilde{x}_{t}^{(i)}, N^{-1}\}$
Update
$\{\tilde{x}_{t}^{(i)}, \tilde{w}_{t}^{(i)}\}$

Figure 2.7The basic PF uses discrete points and SMC methods to approximate an evolving posterior distribution

A resampling scheme associates to each particle $x^t(i)$ a number of "children", saying $N_i \in \mathrm{N}$, such that $\sum_{i=1}^{N} N_i = N$ (usually M=N). Several selection schemes have been proposed in the literatures. These schemes satisfy $\mathrm{E}(N_i) = N\tilde{w}_t^{(i)}$ but their performance varies in terms of the variance of the particles $\mathrm{var}(N_i)$. However, later literatures [47, 78] indicate that the restriction ($\mathrm{E}(N_i) = N\tilde{w}_t^{(i)}$) is unnecessary to obtain convergence results. So it is possible to design biased but computationally inexpensive selection schemes.

We will later present various resampling schemes, namely: Sampling Importance Resampling (SIR), Residual Resampling, Minimum Variance Sampling. However, we found that the specific choice of resampling scheme does not significantly affect the performance of the PF in our application due to finite particles, so we used SIR in all the experiments presented in following chapters.

Sampling Importance Resampling (SIR) and multinomial sampling

Many resampling techniques are derived from the work of Efron [67], Rubin [79] and Smith and Gelfand[80] . Resampling involves mapping the set $\{s^t(i), w^t(i)\}$ into an equally weighted random set $\{x^t(i) : i = 1,...,M\}$ with probabilities $\{\tilde{w}(i) : i = 1,...,M\}$ as proposed in the paper by Gordon [46, 81] as well as the mathematical proof. After constructing the cumulative distribution of the discrete set, a uniformly drawn sampling index *i* is projected onto the distribution range, as shown in Figure 2.8 and then onto the distribution domain. The intersection with the domain constitutes the new sample index *j*. That is, the vector $x^t(j)$ is termed to be the new sample set. Obviously, the vectors with the larger sampling weights will end up with more copies after the resampling process.

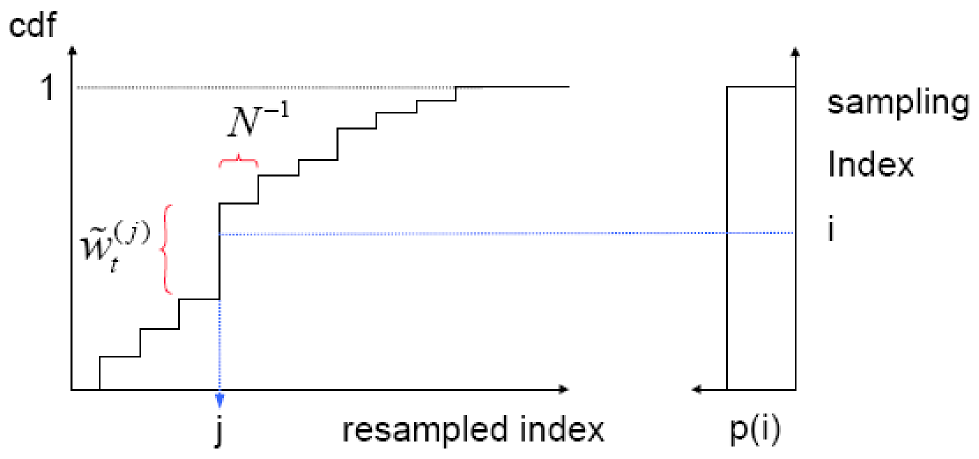


Figure 2.8Resampling process illustration, where a random sample $\{x^t(i), w_t(i)\}$ *is mapped into an equally weighted random measure* $\{x^t(j), N^{-1}\}$. *The index i is drawn from a uniform distribution.*

Sampling M times from the cumulative discrete distribution $\sum_{i=1}^{M} \tilde{w}_t^{(i)} \delta^t(i)(dx^t)$ is equivalent to drawing $(N_i; i=1,...,M)$ from a multinomial distribution referring to the parameters M and $\tilde{w}_t(i)$. This procedure can be implemented in $O(N)$ operations. As we are sampling from a multinomial distribution, the variance is $\text{var}(N_i) = N\tilde{w}_t(i)(1-\tilde{w}_t(i))$. As pointed out in [46, 82], it is possible to design better selection schemes with lower variance.

Residual Resampling

The residual resampling procedure involves three steps [65, 83]. Firstly, set $\tilde{M}_i = \lfloor M\tilde{w}_t(i) \rfloor$. Secondly, perform an SIR procedure to select the remaining $\bar{M}_t = M - \sum_{i=1}^{M} \tilde{M}_i$ samples with the updated new weights $w_t'(i) = \bar{M}_t^{-1}(\tilde{w}_t(i)M - \tilde{M}_i)$. Finally, add the results to the current $\tilde{M}_i$. For this scheme, the variance ($\text{var}(M_i) = \bar{M}_t(i) w_t'(i)(1-w_t'(i))$) is smaller than the one given by the SIR scheme. Moreover, this procedure is computationally cheaper.

Minimum Variance Sampling

This strategy includes the stratified sampling [47] and the Tree Based Branching Algorithm presented in [84]. One samples a set of *M* points uniformly in the interval [0,1], each of the points has a distance of M^{-1} apart. The number of children M_i is taken to be the number of points that lies between $\sum_{j=1}^{i-1} \tilde{w}_t(j)$ and $\sum_{j=1}^{i} \tilde{w}_t(j)$. This strategy introduces a variance on M_i even smaller than the residual resampling scheme, namely $\mathrm{var}(M_i) = \bar{M}_t w_t'(i)(1-\bar{M}_t w_t'(i))$. Its computational complexity is $\mathrm{O}(N)$.

However, generally speaking, the above methods are not promising enough because the copied samples are no longer statistically independent after resampling so the previous convergence result will be lost. It is called losing sampling diversity [48]. Furthermore, although these improved resampling methods are proved to have lower complexity and smaller variance than the traditional SIR in theory, the experimental results show the particle variance are not significantly different [85, 86] due to the limited number of samples, so two problems in PF are sometimes still exit in that scenario. We still choose the traditional SIR as the resampling method in the experiments presented in the rest of the thesis.

2.5 Improved Proposal Distribution Function

Nowadays, there still exist a large number of improved methods for PF algorithm. However, the success of the PF algorithm depends on the validity of following underlying assumptions [87]:

Monte Carlo (MC) assumption: The Dirac point-mass approximation provides an adequate representation of the posterior distribution.

Importance Sampling (IS) assumption: It is possible to obtain samples from the posterior by sampling from a suitable proposal distribution and applying importance sampling corrections.

If any of these conditions are not met, the PF algorithm will perform poorly. The discreteness of the approximation poses a resolution problem. In the resampling stage, any particular sample with a high importance weight will be duplicated many times. Consequently, the whole set of particles may finally degenerate into a single particle. This degeneracy will limit the ability of the algorithm to search for lower minima in other regions of the error surface. In other words, the number of samples used to describe the posterior density function will become too small and inadequate. A straightforward strategy to overcome this problem is to increase the number of particles. A more refined strategy is to implement a Markov chain Monte Carlo (MCMC) step after the selection step as discussed in the following section.

2.5.1 MCMC Move Step

After the selection scheme at time *t*, we derive *N* particles ($N \le M$) distributed marginally approximately according to $p(s^t \mid z^t)$. Since the selection step favors the creation of multiple copies of the "fittest" particles, it enables us to track time varying filtering distributions. However, many particles might not been copied during selection step ($N_i = 0$), whereas others might end up having a large number of children, the extreme case being $N_i = N$ for a particular value *i*. In this case, there is a severe depletion of samples. We, therefore, require a procedure to introduce sample variety after the selection step without affecting the validity of the approximation.

A strategy for solving this problem involves the introduction of MCMC steps of invariant distribution $p(s^t \mid z^t)$ on each particle [70, 82, 88-90]. The basic idea is that if particles are initially distributed according to the posterior $p(\tilde{s}^t \mid z^t)$, then applying a Markov chain transition kernel $\kappa(s^t \mid \tilde{s}^t)$, with invariant distribution $p(s^t \mid z^t)$ such that $\int \kappa(s^t \mid \tilde{s}^t) p(\tilde{s}^t \mid z^t) = p(s^t \mid z^t)$, still results in a set of particles distributed according to the posterior of interest. However, the new particles might have been moved to more interesting areas of the state space. In fact, by applying a Markov transition kernel, the total variation of the current distribution with respect to the invariant distribution can only decrease. Note that we can incorporate any of the standard MCMC methods, such as the Gibbs sampler [91] and Metropolis Hastings algorithms [92], into the filtering framework, but we no longer require the kernel to be ergodic. The MCMC move step can also be interpreted as sampling from the finite mixture distribution $N^{-1}\sum_{i=1}^{N} \kappa(s^t \mid \tilde{s}^t(i))$ .Convergence results for this type of algorithm are presented in [89].

2.5.2 Alternative PF Proposal Distribution

The selection of a proposal distribution function is one of the most critical design issues in importance sampling algorithms and forms the major issue addressed in the following chapters of this thesis. The preference for proposal distribution functions which minimize the variance of the importance weights is advocated by Doucet [82]. The following result has been proved in [70] that the proposal distribution $q(s_t \mid s^{t-1}, z^t) = p(s_t \mid s^{t-1}, z^t)$ minimizes the variance of the importance weights conditional on s^{t-1} and z^t.

This selection of proposal distribution has also been proposed by other researchers including Kong[69], Liu[93] and Zaritskii [94]. Nevertheless, the distribution

$q(s_t \mid s^{t-1}, z^t) = p(s_t \mid s_{t-1})$ (the transition prior, also is the motion model in SLAM application) is the most popular choice of proposal distribution function [34, 46, 47, 95, 96]. Although it results in higher Monte Carlo variation than the optimal proposal function $p(x_t \mid x^{t-1}, y^t)$ because it does not incorporate the most recent observation, it is usually easier to implement [46, 97, 98]. The transition prior is defined in terms of the probabilistic model governing the states' evolution and the process noise statistics. For example, if assumption of the Gaussian process noise model is held, the transition prior is simply defined in Equation (2.23):

$$p(x_t \mid x_{t-1}) = N(f(x_{t-1}, 0), Q_{t-1}) \tag{2.23}$$

As illustrated before, if we fail to use the latest available information to propose new values for the states, only a few particles will have significant importance weights when their likelihood are evaluated. It is therefore of paramount importance to move towards the regions of high likelihood. This problem also arises when the likelihood function is too narrow comparing to the prior function. In Chapter 4 and 5, we shall describe our method mathematically and experimentally, based on the Ant Colony Optimization, to re-arrange the particle set to achieve the optimal importance function.

Prior Editing [82] is an ad-hoc acceptance test for proposing particles in regions of high likelihood. After the prediction step, the residual error $e_t = z_t - h_t(\hat{s}_t(i))$ is calculated. If $|e_t| > K_t \sqrt{r}$, where r is the scale of the measurement error model and K_l is a constant chosen to indicate the region of non-negligible likelihood, then this sample $\hat{x}_t(i)$ is rejected. This iteration stops until a specified number of particles meet the criteria. This approach is too heuristic and therefore increasing the computation cost unless the rejection rate is small. In addition, it will also introduce a bias on the distribution of the particles.

There is another method called **Rejection Methods** [65]. It is based on the principle that if the likelihood is bounded, say $p(z_t \mid s_t) < R_t$, it is possible to sample from the optimal importance distribution $p(s_t \mid s_{t-1}, z_t)$ using an accept/reject procedure. First, we obtain a sample from the prior $\hat{s} \sim p(s_t \mid s_{t-1})$ and a uniform variable $u \sim U_{[0,1]}$. Subsequently, the sample from the prior is accepted if $u \leq p(z_t \mid \hat{s}_t) / R_t$. Otherwise, we will reject this sample and repeat the process until all *M* samples are accepted. Unfortunately, the rejection sampler requires a random number of iterations at each time step. This leads to expensive and unpredictable computation in high-dimensional space [98-100].

The auxiliary Particle Filter [100] allows us to obtain approximate samples from the optimal importance distribution by introducing an auxiliary variable *k*. Specifically, the aim of the algorithm is to draw samples from the joint distribution

$$q(s_t, k \mid s^{t-1}, z^t) \propto p(z_t \mid \mu_t(k))\, p(s_t \mid s_{t-1}(k))\, p(s^{t-1}(k) \mid z^{t-1}) \tag{2.24}$$

where $\mu_t(k)$, *k=1,…,M* is an additional variable equals to the mean, mode, or other value associated with the transition prior. One way to accomplish this objective is to evaluate the marginal auxiliary variable weights $g(k \mid s^{t-1}, z^t) \propto p(z_t \mid \mu_t(k)) p(x^{t-1}(k) \mid z^{t-1})$ and use them to select *N* particles from the transition prior. Typically, one boosts the samples set so that *M>N*. The PF then proceeds to evaluate the correlation weights

$$w_t = \frac{p(z_t \mid s_t(j))}{p(z_t \mid \mu_t(k_j))} \tag{2.25}$$

where *j = 1,…,N* and k_j denotes the *k*-th “parent” of particle *j*. Finally, the correction weights are used to perform a second resampling step to obtain *M* particles approximately distributed according to the posterior distribution.

Compared to the generic PF, the auxiliary PF can generate better estimates of the posterior whenever the likelihood is situated in one of the prior tails. On the other hand, if the likelihood and prior are mostly overlapped, the generic PF may produce more accurate estimates. The latter behavior is a consequence of the extra variance introduced by the additional selection step.

Another way to interpret the auxiliary PF is to treat the distribution $q(s_t, k \mid s^{t-1}, z^t)$ as an importance proposal. Therefore, the following importance weights are obtained as follows:

$$\begin{aligned} w_t &\propto \frac{p(s^t(k) \mid y^t)}{p(z_t \mid \mu_t(k))\, p(s_t \mid s_{t-1}(k))\, p(s^{t-1}(k) \mid z^{t-1})} \\ &\propto \frac{p(z_t \mid s_t(k))\, p(s_t \mid s_{t-1}(k))\, p(s^{t-1}(k) \mid z^{t-1})}{p(z_t \mid \mu_t(k))\, p(s_t \mid s_{t-1}(k))\, p(s^{t-1}(k) \mid z^{t-1})} \\ &= \frac{p(z_t \mid s_t(k))}{p(z_t \mid \mu_t(k))} \end{aligned} \tag{2.26}$$

However, the three methods presented above for an improved proposal distribution have numerous inefficiencies as discussed in the literature. Researchers also try to incorporate Kalman Filters to design better proposal distribution.

Extended Kalman Particle Filter

Extended Kalman Filter is also adopted to construct the proposal distribution function, by incorporating the most current observation with the optimal Gaussian approximation of the state [88, 101]. It relies on the first order Taylor series expansions of the likelihood and transition prior, as well as a Gaussian assumption on all random variables in question. In this framework, the EKF approximates the optimal MMSE (Minimal Mean Square Error) estimator of the system state by calculating the conditional mean of the state, given all observations. This is achieved in a recursive framework, by propagating the Gaussian approximation of the posterior distribution through time, combining it at each time step with the new observation. In other words, the EKF runs the following recursive approximation to the true posterior filtering density,

$$p(s_t | z^t) \approx p_N(s_t | z^t) = N(\bar{x}_t, \hat{P}_t) \qquad (2.27)$$

Within the PF framework, a separate EKF is used to generate and propagate a Gaussian proposal distribution for each particle, i.e.,

$$q(s_t(i) | s^{t-1}(i), z^t) = N(\bar{s}_t(i), \hat{P}_t(i)), i = 1, \ldots, N$$

(2.28)

That is, at time *t-1* one uses the EKF equations, with the new data, to calculate the mean and covariance of the importance distribution function for each particle. Next, we go to sample the *i-th* particle as the usual PF steps. This method requires that we propagate the covariance $\hat{P}(i)$ and specify the EKF process and measurement noise covariance. Since EKF is an MMSE estimator, this local linearization leads to an improved annealed sampling algorithm, whereby the variance of each proposal distribution changes with time. Ideally, the algorithm starts searching over a large region of the error surface and eventually, it concentrates on the regions with smaller errors.

Although EKF possibly creates a better proposal distribution by making a Gaussian assumption on the form of the posteriors as well as introducing inaccuracies due to linearization, in fact, the current observation at time *t* in the proposal distribution that generated by EKF will not be Gaussian. This can be easily shown by a Bayes' rule expansion of the proposal distribution [75].

Unscented Particle Filter

Similar to Extended Kalman PF, the Unscented PF uses Unscented Kalman Filter (UKF) as a distribution generation mechanism within the PF framework, because UKF is regarded as having a bigger support overlap with the true posterior distribution than the overlap achieved by the EKF estimates. This is in part related to the fact that the UKF calculates the posterior covariance accurately to the third order, while the EKF relies on a first order biased approximation. In short, the UKF uses a deterministic sampling technique known as the unscented transform to recursive minimum mean-square-error (RMMSE) estimation [102] to pick a minimal set of sample points (called sigma points) around the mean. These sigma points are then propagated through the non-linear functions, from which the mean and covariance of the estimate are then recovered up to the second order of the Taylor expansion. Consequently, the result is a filter which can capture the true mean and covariance more accurately. In addition, this technique removes the requirement to explicitly calculate Jacobians, which for complex functions can be a difficult task in itself.

2.6 PF SLAM

PF has a number of useful properties when compared to other methodologies for solving the SLAM problem, e.g. EKF. First, the PF can approximate arbitrarily complex probability distributions, where the EKF is only applicable to Gaussian descriptions at all levels of uncertainty. It is useful to apply PF into the non-linear and non-Gaussian environment. Additionally, PF is not significantly affected by non-linearities in the motion and measurement models of the robot.

For application of PF to the SLAM problem it is possible to begin at the SLAM posterior which is more complicated than the robot localization posterior $p(s_t, m \mid z^t, u^t)$. Similar to the Kalman Filter formulation, this probabilistic relationship is valid if the Markov assumption is valid. The Markov assumption states that if the current state is known then the previous and future data are conditionally independent. Similar to the EKF process, the SLAM posterior can be converted into the Bayes' Filter by the Bayes' Rule as stated in Equation (2.6).

The Bayes' Filter can be solved in an approximate manner using PF discrete samples. The SLAM posterior may be thought of as a belief state, where a single $belief(i)$ is used to present the hypothesis vector including robot's pose, the map of the environment, and an associated weight that defines the probability of the given belief being correct. In PF, this is presented by *M* samples of a continuous probability distribution along with the associated weight for each sample.

$$belief(i) = p(s_t(i), m(i) \mid z^t(i), u^t(i)) = \{s_t(i), m(i), p(i)\}_{i=1,\ldots,m} \tag{2.29}$$

At initialization of this belief state, $belief(i)$ is defined by whatever probability distribution is known to define the uncertainty in s_t and *m* [64].

$$p(s_0, m \mid z^0, u^0) = p(s_0, m) \tag{2.30}$$

First, if a prior information describing the uncertainty in the states is available it is used to define the initial distributions above. These initial distribution are then sampled from uniform distribution to create the particle representation of the state space $s_0(i)$, $w_0(i)$, in which the initial particle weighting $w_0(i)$ is set to $1/M$ [64].

The measurement step alters the belief state by assigning the particle weights using the likelihood of the particle occurrence given the measurement, z_t.

$$p(z_t \mid s_t(i), m(i)) \tag{2.31}$$

The remaining derivation of PF SLAM is similar to the general PF which was described previously in Algorithm 2.1, except that the PF SLAM includes two sets of variables (the robot pose and the map) to be estimated.

2.7 PF SLAM Challenges

Despite their abilities to track arbitrarily complex, multi-modal distributions, PF algorithms still have some challenges to be overcome. First, PF carries a pronounced computational requirement: the number of particles which is needed to track a variable increases the scales exponentially with the dimension of the variable state. With a SLAM posterior that includes hundreds of dimensions, which represent the robot's position and heading, grids in the map, it could require millions of particles to be tracked effectively [103, 104]. Thus, a considerable amount of computational power is needed if we want to finish the real-time SLAM solution.

Another drawback of the PF SLAM, as it was originally formulated, is that its computation scale is proportional to the size of state spaces. This is a result of the exponential time behavior of this implementation of PF, which is not acceptable for problems such as SLAM which has to solve numerous states in real-time. Moreover, problem will a higher dimension implies that it needs more samples to achieve the result.

2.7.1 Improved SLAM Approaches

Rao-Blackwellized Particle Filter and FastSLAM

The Rao-Blackwellized Particle Filter (RBPF) [105] has offered some solutions to this computational burden problem. This method has not only showed great enhancement to the computational efficiency, but also improvement in estimation accuracy. The concept of Rao-Blackwellized PF for application to the SLAM problem has been greatly developed by Montemerlo and Thrun [32, 77, 101]. Based on the Rao-Blackwellized concept, a recent innovation introduced by Montemerlo [77] solve the computation problem by conditioning the SLAM posterior on the entire path instead of the current pose. The basic premise is this: if the entire path of the robot is known, not just the current pose, a single landmark observation will not affect the location or uncertainty of any other map landmark. Consequently, landmark measurements are conditionally independent. All landmark correlations are ignored and the SLAM posterior can be represented as the product of the path posterior and N independent landmark estimators represented by Equation 2.32.

$$p(s_t,\theta \mid z_t,u_t) = p(s_t \mid z_t,u_t)\prod_{n=1}^{N} p(\theta_n \mid s_t) \qquad (2.32)$$

Montemerlo also illustrates that all update equations for the filter will depend only on the most recent pose under the Markov property of the SLAM posterior. Also, this has the benefit of only having to maintain N 2*2 covariance matrices for each particle as opposed to a full *(2N+3) * (2N+3)* covariance matrix. This factorization, illustrated in Figure 2.9, forms a PF based on the sampling architecture of Rao-Blackwellization, where a small subset of variables are sampled (the robot pose information) and other marginal $\prod_{n=1}^{N} p(\theta_n \mid s_t, z_t, u_t)$ are calculated in closed form (landmark estimation parameters) [93]. The application of this principle to the position-tracking PF was introduced by [94]. Building on the structure of Equation 2.33, Montemerlo developed an algorithm named FastSLAM that represents the posterior with *N+1* filters, one of each term represents a different hypothesis of the SLAM posterior. As in Equation 2.33 each particle represents robot pose and a set of independent data, statistics of landmarks, considered as marginal.

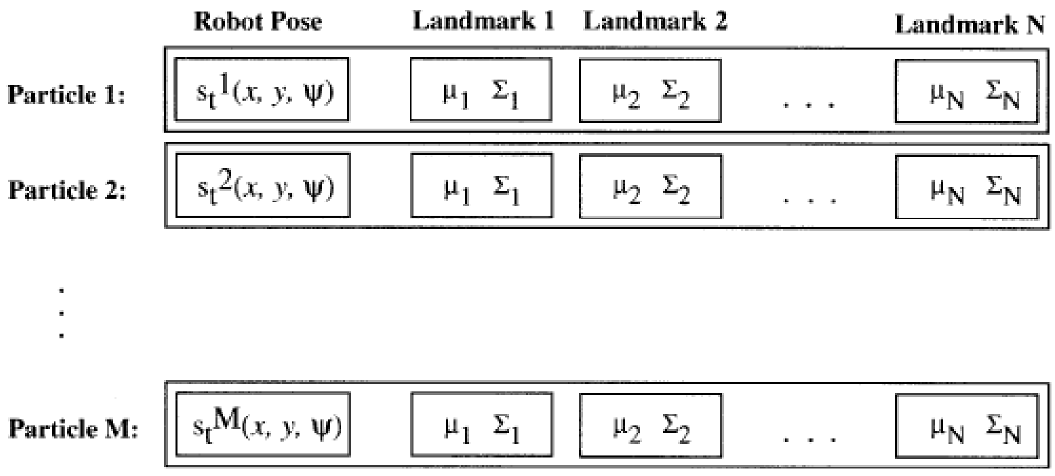


Figure 2.9 The factored SLAM posterior

Each particle carries a post estimate as well as map features. Map features are independent, so that they are considered to be marginal .Therefore, the states in particles are reduced.

$$S_t^{[m]} = \left\langle s_t^{[m]}, \mu_{1,t}^{[m]}, \Sigma_{1,t}^{[m]}, \ldots, \mu_{N,t}^{[m]}, \Sigma_{N,t}^{[m]} \right\rangle \qquad (2.33)$$

The bracketed notation represents the index of the particle. The agent pose information for each hypothesis $s_t^{[m]}$ is updated with the SIR method explained previously.

The rest of the SLAM posterior is estimated by independent Gaussian estimators representing the mean $\mu_{n,t}^{[m]}$ and covariance $\Sigma_{n,t}^{[m]}$ of each observed landmark. Given a two or three dimensional Cartesian space, these landmarks will be low-dimensional and fixed in size. Each particle carries its own set of landmark estimators. Taken in total, the particles form an array of *M* hypotheses that represent a discrete approximation to the optimal Bayesian SLAM posterior [32].

Researchers further broaden the above methods from landmark based map representation to grid-based map. Actually they perform a similar idea, which estimates a posterior $p(s^t \mid z^t, u^t)$ about potential trajectories s^t of the robot given its observations z^t and its odometry measurements u^{t-1}. This distribution is then used to compute a posterior over map grids and trajectories:

$$\begin{aligned} & p(s^t, m \mid z^t, u^{t-1}) && (2.34.1) \\ &= p(m \mid s^t, z^t, u^{t-1}) p(s^t \mid z^t, u^{t-1}) && (2.34.2) \\ &= p(m \mid s^t, z^t) p(s^t \mid z^t, u^{t-1}) \end{aligned}$$

(2.34)

where Equation 2.34.2 is obtained from 2.34.1 by assuming that *m* is independent of the odometry measurements u^{t-1} given all poses s^t of the robot and the corresponding observations z^{t-1}.

To estimate the first term of Equation 2.34.2, namely the posterior $p(s^t \mid z^t, u^{t-1})$, the Rao-Blackwellized mapping uses a PF in which an individual map is associated to every sample [95]. Each map is built given the observation z^t and the trajectory s^t represented by the corresponding particle. The naïve implementation of this idea of the Rao-Blackwellized PF requires *O(MK)* operations, where *M* is the number of particles in the PF and *K* is the number of map grids.

In some extents, the RBPF reduces the number of variables that must be sampled by identifying variables that do not need to be sampled to be computed. However, the construction of optimal proposal distribution is much more straightforward, and can be considered as alternative methods of RBPF.

2.8 Conclusions

In this chapter, we derive a possible solution of SLAM by the Bayes' Filter. Due to an intractable term, we use two categories of problem: Kalman Filters and PF to estimate the term which is not integrable. Comparing to Kalman Filters, PF require less computational effort, as well as having the ability to estimate non-Gaussian and non-linear distribution. However, PF, especially those applied in high dimensional problems, e.g. multi-robot SLAM, suffered from the particle impoverishment problem and particle size dependency. To eliminate these problems, a number of improved strategies, such as improved proposal distribution approaching the optimal solution, and Rao-Blackwellized Particle Filter constructing conditionally independence terms to reduce the computational dimension, are discussed and analyzed. In fact, the resampling step can be regarded as a combinatorial optimization problem. In next chapter, we will introduce one of its solutions, the Ant Colony Optimization, and attempt to use it in PF in the following chapters.

Chapter 3 Ant Colony Optimization

3.1 Introduction

Ant Colony Optimization (ACO) is inspired from the foraging behavior of some ant species and is often regarded as a metaheuristic method [106]. ACO is a general method for solving combinatorial optimization problems, whose framework will be discussed in Section 3.2. The ACO is inspired by the pheromone trail laying and following behavior of real ants which make use of pheromones as a communication medium. Similar to its biological counter-part, in ACO algorithm implementation, it simulates a colony of simple entities, called artificial ants, mediated by artificial pheromone trails to track the optimal solution. During this process, the pheromone trails is treated as a distributed numerical data set from which ants use to construct a solution and ants adopt during the algorithm's execution to reflect their search experience.

The development of ACO is inspired from natural behavior of ants. In the early nineties an algorithm called Ant System (AS) was proposed as a novel heuristic approach for the solution of combinational optimization problems [107-109] . It was first used to solve the traveling salesman problem (TSP) [110, 111] . Despite the encouraging initial results, AS could not compete with the most advanced algorithms, such as Cross-entropy method [112], Bees Algorithm [113, 114], Genetic and Evolutionary Computation [115, 116] for solving combinatorial optimization problems. However, it played an important role in stimulating different improved ACO algorithms such as MAX-MIN Ant System, Ant Colony System, etc., which obtain much better computational performance, and with applications in a variety of problems. In fact, improved ACO was applied to different optimization problems besides the TSP, including quadratic assignment, vehicle routing, sequential ordering, scheduling, routing in the complex networks, and so on [117-122] .

3.2 Framework of the ACO Algorithm

Ant Colony Optimization (ACO) has been formalized into a metaheuristic for combinatorial optimization problems by Dorigo and co-workers [109]. A *metaheurstic* is a set of algorithmic concepts that can be used to define heuristic methods applicable to a set of problems. In other words, metaheuristic is a general-purpose algorithmic framework that is able to be applied into different kinds of optimization problems with relatively few modifications, such as Simulated Annealing [123, 124] , Tabu Search [125-127], Iterated Local Search [128] , Evolutionary Computation [129-131], and Ant Colony Optimization[106, 109, 109, 132, 133].

Combinatorial optimization is a branch of optimization problem. Its domain is optimization problems where the set of feasible solutions is discrete or can be reduced to a discrete one, and the goal is to find the best possible solution. One of the most representative examples of combinatorial optimization problem is the Traveling Salesman Problem (TSP), while it also includes Vehicle Routing Problem, Minimum Spanning Tree Problem, etc. A model $P = (S, \Omega, f)$ of a combinatorial optimization problem consists of the following conditions:

- a search space **S** defined over a finite set of discrete decision variable $X_i, i = 1, \ldots, n$,
- a set **Ω** of constraints among the variable,
- an objective function $f : S \to R_0^+$ to be minimized.[1]

The generic variable X_i takes values that satisfies all constraints in $D_i = \{v_i^1, ..., v_i^{|D_i|}\}$.

A feasible solution $s^* \in S$, which is a complete assignment of values to variables that

[1] A maximization problem can be easily changed to a minimization problem, by modifying a maximizing function *g* into *f*, which *f*=-*g*.

satisfies all constraints in **Ω,** is called a global optimum if and only if: $f(s^*) \le f(s)$ for $\forall s \in S$.

The pheromone model of ACO can be derived after the general model of a combinatorial optimization problem. A pheromone value is associated with each possible solution component. Formally speaking, the pheromone value τ_{ij} is associated with a solution component c_{ij}. The set of all possible solution components is denoted by C.

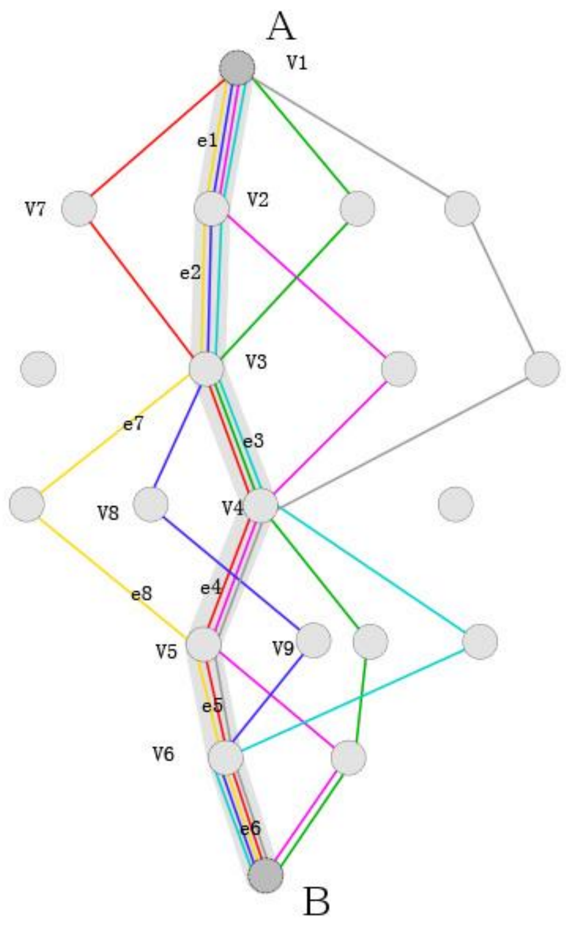


Figure 3.1 The Construction of ACO Solution

Assuming that ant colony moves from A to B, several edges through different vertices can be obtained. So a candidate solution can be represented by a set of vertices, such as $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ *and* $\{v_1, v_2, v_3, v_8, v_9, v_6\}$*, or a set of edges, such as* $\{e_1, e_2, e_3, e_4, e_5, e_6\}$ *and* $\{e_1, e_2, e_3, e_7, e_8, e_5, e_6\}$*, based on different applications*

In ACO, an artificial ant builds a solution based on the following rules.

1) The ants traversing the fully connected Construction Graph, denoted as $G_C(V,E)$ as shown in Figure 3.1, where *V* is a set of vertices and *E* is a set of edges. This kind of graph offers two ways to present the set of solution components ***C***: components may be represented either by vertices or by edges. Artificial ants move from vertex to vertex along edges in the graph, simultaneously building a partial candidate solution incrementally.

2) Ants deposit a certain amount of pheromone on the components; that is, either on the vertices or on the edges that they traverse. The amount $\Delta\tau$ of pheromone deposited is a user-defined parameter, or depending on the quality of the solution found. Subsequent ants use the pheromone information as a reference for their traverse path selection toward promising regions in the search space.

A more detailed description of the ACO is given below and the algorithm description is given in Algorithm 3.1.

1) ***Construct Ant Solutions***: A set of *m* artificial ants constructs solutions from elements of a finite set of available solution components $C = \{c_{ij}\}, i = 1,...,n, j = 1,...,|D_i|$. A solution construction begins with an empty partial solution $s^P = \varnothing$. At each iteration, the partial solution s^P is built by adding a feasible solution component from the set $N(s^P) \subseteq C$, which is defined as the set of components that can be added to the current partial solution s^P following all of the constraints in Ω.

The choice of a solution component from $N(s^P)$ is guided by a stochastic mechanism, which is biased by the pheromone associated with elements of $N(s^P)$. The rule for the stochastic choice of solution components varies across different ACO algorithms.

2) ***Apply Local Search***: Once solutions have been constructed, and before updating the pheromone, it is common to improve the solutions obtained by the ants through a local search. This phase, which is highly problem-oriented, is optional although it is usually included in state-of-the-art ACO algorithms.

3) ***Update Pheromone***: The aim of the pheromone update is to increase the pheromone values associated with good or promising solutions, and to decrease those that are associated with bad ones. The complete ACO algorithm is shown below:

```
Algorithm 3.1 The Ant Colony Optimization Metaheuristic

Set Parameters, initialize pheromone trails
While termination condition not met do
    • Construct Ant Solutions
    • Apply Local Search (optional)
    • Update Pheromones
End while
```

3.3 Alternative ACO algorithms

Algorithm 3.1 depicts the ACO framework, from which several alternative search methods are developed. The MAX-MIN Ant system and Ant Colony System are two most successful ones among others. In the following, we will introduce the algorithm of the original Ant System, as well as these two improved alternatives.

1) Ant System (AS): It is the first ACO algorithm proposed in the literatures [108, 109]. Its main characteristic is that, at each iteration, the pheromone values are updated by all the *m* ants that have built a solution in the iteration. The pheromone τ_{ij} associated with a candidate solution is updated according to Equation 3.1.

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta^{k} \tag{3.1}$$

where ρ is the evaporation rate, *m* is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on the path (i, j) by ant *k*:

$$\Delta\tau_{ij} = \begin{cases} Q/L_k\,, & \text{if ant k used path (i} \\ 0\,, & \text{otherwise} \end{cases} \tag{3.2}$$

where Q is a constant, and L_k is the length of the tour constructed by ant *k*.

In the construction of a solution, ants select the solution through a stochastic mechanism. When ant *k* is in original location *i* and has so far constructed the partial solution s^P, the probability of going to vertex *j* is given by the following equation.:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{ij} \in N(s^P)} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}}, & \text{if } \mathrm{c}_{ij} \in N(s^P) \\ 0, \text{ otherwise} \end{cases} \tag{3.3}$$

where $N(s^P)$ is the set of feasible components; that is, all the possible solutions. The parameters α and β control the relative importance of pheromone τ_{ij} versus the heuristic information η_{ij}, which is given by:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3.4}$$

where d_{ij} is the cost of the ant's tour when constructing the candidate solution. For example, it equals to the length between city *i* and *j* in TSP.

2) MAX-MIN Ant System (MMAS): This algorithm [134] is an improved strategy over the original Ant System. Its characterizing elements are that only the best ant updates the pheromone trails and that the value of the pheromone is bounded. The pheromone update is implemented as follows:

$$\tau_{ij} \leftarrow \left[(1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{best}\right]_{\tau_{min}}^{\tau_{max}} \qquad (3.5)$$

where $\tau_{\max}$ and $\tau_{\min}$ are respectively the upper and lower bounds imposed on the pheromone; the operator $[x]_b^a$ is defined as:

$$[x]_b^a = \begin{cases} a, & \text{if } x > \\ b, & \text{if } x \\ x, & \text{otherw} \end{cases} \qquad (3.6)$$

and $\Delta\tau_{ij}^{best}$ is

$$\Delta\tau_{ij}^{best} = \begin{cases} 1/L_{best}, & \text{if (i,j) belongs to th} \\ & 0, \text{ otherwise} \end{cases} \qquad (3.7)$$

L_{best} is the cost of constructing the best solution. This can (subject to the problem) either be the best found in the current iteration *iteration-best* (L_{ib})or be the best solution found since the start of the algorithm *best-so-far* (L_{bs}) or combination of both.

Concerning the lower and upper bounds on the pheromone values, $\tau_{\max}$ and $\tau_{\min}$, they are typically obtained empirically and adjusted for a specific problem [135]. Nevertheless, some guidelines have been provided for defining $\tau_{\max}$ and $\tau_{\min}$ on the basis of analytical consideration.[134]

3) Ant Colony System (ACS): The most important contribution of ACS [136, 136-138] is the introduction of a local pheromone update in addition to the

pheromone update performed at the end of the construction process (called offline pheromone update).

The local pheromone update in ACS is performed by all ants after each construction step. Each ant applies the pheromone only to the last path traversed

$$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi\cdot\tau_0 \qquad (3.8)$$

where $\varphi \in (0,1]$ is the pheromone evaporate coefficient, and τ_0 is the initial value of the pheromone.

The main goal of the local update is to diversify the search performed by subsequent ants during an iteration by decreasing the pheromone concentration on the traversed path; ants encourage subsequent ants to choose other edges and, hence, to produce different solutions. This makes it less likely that several ants produce an identical solution during one iteration.

The offline pheromone update, similar to MMAS, is applied at the end of each iteration by only one ant, which can be either iteration-best or best-so-far. However, the update equation is different as given in Equation 3.9.

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho)\tau_{ij} + \rho\cdot\Delta\tau_{ij}, & \text{if } (i,j) \text{ belongs to} \\ \tau_{ij}, & \textit{otherwise} \end{cases} \qquad (3.9)$$

As in MMAS, $\Delta\tau_{ij} = 1/L_{best}$, where L_{best} can be either L_{ib} or L_{bs}.

Another important difference between ACS and AS is in the decision rule used by ants during the construction process. In ACS, the so-called pseudorandom proportional rule is used; the probability for an ant to move candidate solution component *i* to another component *j* depends on a random variable *q* uniformly distributed over [0,1], and the parameter *q0*; if $q \le q_0$, then $j = \arg\max_{c_{ij}\in N(s^P)}\{\tau_{il}\eta_{il}^{\beta}\}$, otherwise Equation 3.3 is used. The following table summarizes different definitions applied in ACO algorithms discussed above.

	Pheromone	Probability
AS	$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$	$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{ij} \in N(s^{P})} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}}, \text{ if } c_{ij} \in N(s^{P}) \\ 0, \text{ otherwise} \end{cases}$
MMAS	$\tau_{ij} \leftarrow [(1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{best}]_{\tau_{\min}}^{\tau_{\max}}$	$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{ij} \in N(s^{P})} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}}, \text{ if } c_{ij} \in N(s^{P}) \\ 0, \text{ otherwise} \end{cases}$
ACS	$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi\cdot\tau_{0}$	$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{ij} \in N(s^{P})} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}}, \text{ if } c_{ij} \in N(s^{P}) \\ 0, \text{ otherwise} \end{cases}$ except $j = \arg\max_{c_{ij} \in N(s^{P})} \{\tau_{il}\eta_{il}^{\beta}\}$ when $q \le q_{0}$

Table 3.1 Definitions of Alternative ACO Algorithms

3.4 Convergence Proof

In Chapter 2, we introduced a couple of novel PF methods, the extended Kalman PF and Unscented PF which successfully construct sub-optimal proposal distributions. Similar to these ideas, when working in multi-robot SLAM problem, we plan to apply a metaheuristic method to construct the distributions. As one of the metaheuristic candidates, the convergence of ACO will be conducted in the following sections.

3.4.1 Previous Proof Works

In this section, we will propose two theorems for algorithms proposed in the previous section. First, we show that $ACO_{\text{glb},\tau_{\min}}$, which employs global best pheromone update rule and a lower limit of the feasible pheromone trails, is guaranteed to find an optimal solution with a probability that can be made arbitrarily close to one if given enough time. Second, we show that after the optimal solution was found with a fixed number of iteration t_0, the pheromone trails on the connections of the optimal solution are larger than those on any other connections. This result is then extended to show that an optimal solution can be constructed with a probability larger than $1-\bar{\varepsilon}(\tau_{\min},\tau_{\max})$, where $\tau_{\max}$ is the maximum value the pheromones may take. The detailed proof of Theorem 3.1 and its corollary can be found in [139].

Theorem 3.1: Let $P^*(t)$ be the probability that the algorithm $ACO_{\text{glb},\tau_{\min}}$ locates an optimal solution at least once within the first *t* iterations. Then, for an arbitrary choice of a small $\varepsilon > 0$ and for a sufficiently large *t*, it holds that

$$P^*(t) \geq 1-\varepsilon \tag{3.10}$$

and asymptotically

$$\lim_{t\to\infty} P^*(t) = 1 \tag{3.11}$$

Theorem 3.2: Let t^* be the iteration when the first optimal solution has been found. Then a value t_0 exists such that the following holds:

$\tau_{ij}(t) > \tau_{kl}(t)$, $\forall(i,j)\in s^*$, $\forall(k,l)\in L \wedge (k,l)\notin s^*$, and $\forall t > t^* + t_0 = t^* + \lceil (1-\rho)/\rho \rceil$

Corollary 3.1: Let t^* be the iteration when the first optimal solution has been found and $P(s^*,t,k)$ be the probability that an arbitrary ant *k* constructs s^* in the *t*th iteration, with $t > t^*$.

Then it holds that

$$\lim_{t\to\infty} P(s^*,t,k) \geq 1-\hat{\varepsilon}(\tau_{\min},\tau_{\max}) \tag{3.12}$$

3.4.2 Convergence Proof based on Entropy

In this section, a more general convergence theorem is proved based on entropy definition[140]. In information theory, entropy is a measure of the uncertainty associated with a random variable. The term usually refers to the Shannon entropy, which quantifies, in the sense of an expected value, the information contained in a message, usually in units such as bits. Equivalently, the Shannon entropy is a measurement of the average information content of a distribution based on a reference. Actually the term entropy in information theory is a measure of randomness in a probability distribution, but it also can demonstrate the distribution randomness in PF or the tendency of ants traverse in ACO.

Theorem 3.3:

Referring to Theorem (3.1), ACO converges to the optimal solution after enough iteration.

Proof:

Assuming that it exists M ants in the model, which is represented as k (k = 1,2,3…M). For a given problem, the algorithm does search in the number of N states in the state space. We represent the search state with the number s (s = 1, 2, 3…N)

The state model is depicted in Figure 3.2. We use $L_s^k = \{l_{s,1}^k, l_{s,2}^k, ..., l_{s,q(i,j)}^k\}$ to represent the state space that the ants search , in which $q(s,k)$ is the cumulative number function in the sth search state of kth ant. Therefore, the probability distribution function over the state space is $P_s^k = \{p_{s,1}^k, p_{s,2}^k, ..., p_{s,q(s,k)}^k\}$.

The search entropy of kth ant in sth search state is defined as,

$$H_s^k = -\sum_{r=1}^{q(s,k)} p_{s,r}^k \log_c p_{s,r}^k \tag{3.13}$$

where c means the base of the logarithm used.

According to the above definition, the search entropy is given below:

$$H^k = \sum_{s=1}^{N} H_s^k \tag{3.14}$$

which demonstrates the uncertainties of *k*th ant searching the solution in *s*th state. Therefore, the overall entropy of the ant colony is represented by Equation 3.15

$$H = \frac{1}{M} \sum_{k=1}^{M} H^k \tag{3.15}$$

which demonstrates the average uncertainties of the whole ant colony.

It is trivial to derive that the above three entropies reach their maximum value when all $p_{s,q(s,k)}^k$ have equivalent values.

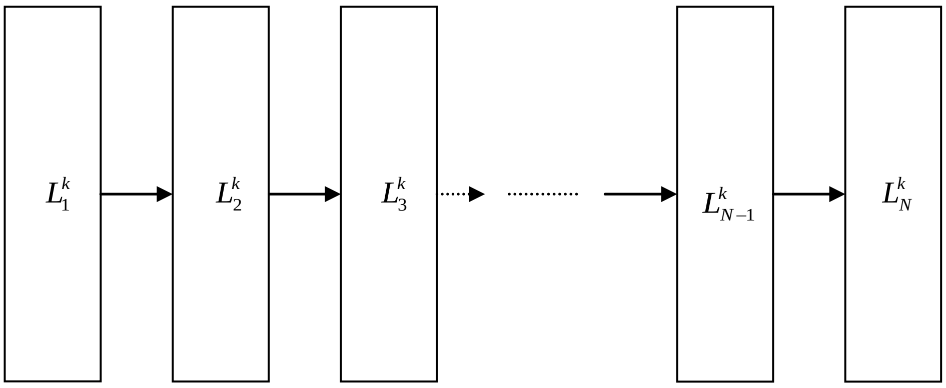


Figure 3.2 A search state model demonstration

The optimization process can be changed into the problem of searching for a solution under the above model, in which $L_s^k = \{l_{s,1}^k, l_{s,2}^k ..., l_{s,q(i)}^k\}$. The state *s* is independent of the ant number *k* while the search state only depends on the search probability P_s^k.

Theorem 3.4: If a problem can be generalize as the above model (Figure 3.2) then each status diagram is corresponding to a probability sequence.

When the ants select the target based on the probability sequence stably, the algorithm convergences if and only if

$$\lim_{t\to+\infty} H(t) = 0$$

Necessity: Assume that there is only one ant in the colony. It selects the direction based on the probability sequence $\bar{P}_s = (\bar{p}_{s,1}, \bar{p}_{s,2}, ..., \bar{p}_{s,q(i)})$. Since the algorithm stably converges, $\lim_{t\to+\infty} \bar{P}_s(t) = (0,0,...,0,1,0,..,0)$. Note that the position of 1 appearing is only related to the number of *s*. From the definition of entropy, we can derive $\lim_{t\to+\infty} H_s(t) = 0$. Let the search probability be $\bar{P}_s = \{\bar{p}_{s,t} \mid t = 1,2,...\}$. For the ant colony includes *M* ants, the search probability is

$$\bar{P}_s^k = \{\bar{P}_s(t) \mid t = 1, 2, ...\} = \bar{P}_s\{t \mid t = (k-1)M + k, t = \ldots\} \tag{3.16}$$

Since the search is convergence,

$$\lim_{t\to+\infty} \bar{P}_s^k(t) = \lim_{t\to+\infty} \bar{P}_s(t) = (0, 0, ...,) \tag{3.17}$$

Therefore, the entropy of the whole ant colony is still $\lim_{t\to+\infty} H(t) = 0$

Sufficiency: If the ant colony has only one ant, let us assume that the entropy in search state *i* is $\bar{H}_i = \{\bar{H}_i(1), \bar{H}_i(2), ..., \bar{H}_i(t), ...\}$.

Let the search entropy in time *t* is $\bar{H}(t)$, so

$$\lim_{t\to+\infty} \bar{H}(t) = 0 \Rightarrow \lim_{t\to+\infty} \bar{H}_i(t) = (\tag{3.18}$$

Since the probability sequence selection of ants is stable, we get

$$\lim_{t\to+\infty} \bar{P}_i(t) = (0,0,...,0,1,0,...,0) \tag{3.19}$$

Therefore, the algorithm converges.

For the ant colony includes M ants, the entropy of colony is

$$H_s = \{H_s^1(1), H_s^2(1), ..., H_s^M(1), H_s^1(2), H_s^2(2), ..., H_s^M(2), ..., H_s^1(t), H_s^2(t), ..., H_s^M(t), ...\} \tag{3.20}$$

Since $\lim_{t\to+\infty} \bar{H}(t) = 0 \Rightarrow \lim_{t\to+\infty} \bar{H}_s(t) = 0$, we know searching entropy of M ants for one solution equals to searching entropy of one ant for M solutions. Therefore $H_s \Leftrightarrow \bar{H}_s$.

From the above proofs, we can deduce that the probability sequence P_s corresponding to H_s is convergence. Obviously, the probability sequence of a certain ant in state *s* is identical to the subset of probability sequence corresponding to $\bar{H}_s$, so for any $1 \le k \le M$, we have $\lim_{t\to+\infty} P_s^k(t) = (0, 0, ..., 0, 1, 0, ..., 0)$. Also note that the position of 1 only depends on *s*. Therefore, the algorithm is convergent.

3.5 Conclusions

In this chapter, we have introduced the fundamental framework of Ant Colony Optimization, as well as three types of ACO algorithm within this framework. Based on the previous works about the convergence proof under a specific type of ACO algorithm, we propose a more general convergence proof based on entropy. In the next chapter, we will discuss how the ACO can be applied into PF in order to optimize the proposal distribution.

Chapter 4 Ant Colony Optimization Improved Particle Filter

4.1 Introduction

As discussed in previous chapters, PF based multi-robot SLAM system has two characteristics.

1) In general, the number of dimension in the multi-robot SLAM can be estimated by Equation 4.1.

$$d = 3 \times n_r + n_m , \tag{4.1}$$

Where n_r is the number of robots, n_m is the number of map grids. Since the number of map grids is significantly larger than the number of robots, it dominates the dimension of the state space. Therefore, the estimation of map consumes a huge proportion of the total computational effort. For example, assuming the situation that there are two robots navigating in an environment divided into 100×100 grids cooperatively; the cost of our experiment is equivalent to the estimation problem with 10006 dimensions as estimated by Equation 4.1. Note that the map with 100×100 grids may only represent an area of $100cm \times 100cm$ or $1m \times 1m$ in the real world if we define that one grid is $1cm \times 1cm$ in the map. Alternatively we can modify the grids definition into $5cm \times 5cm$, so as to relieve the computational requirements, but at the cost of reducing the resolution of the map representation.

Moreover, high dimension problem requires more particle samples to keep the error level [37] to a minimal. A straightforward method to improve the estimation accuracy is by using more particles, which will aggravate the computational burden. Nevertheless, a more elegant method is to optimize the particle distribution (or optimize the proposal distribution) so as to improve the particle efficiency, which was proposed by [75, 141]. A detailed analysis of this problem will be presented in following sections.

2) Another characteristic, or potential advantage in multi-robot system is its feasibility to distribute the computing process to different robots more efficiently comparing to a single robot system[142]. Since low-cost computing processor is already available in each robot, the system has the potential to accomplish real-time high-dimension estimation cooperatively.

To fully utilize the computational potential of multiple processors, there exist several distributed PF algorithms [86], which can share the computational burden by all the processors with some distributed implementations. Relevant methods will be reviewed and a distributed PF designed for multi-robot system will be discussed in Chapter 6.

4.2 Optimal Proposal Distribution Construction

As discussed in Chapter 2, particle impoverishment is a major problem due to the random prediction sample in PF algorithm, especially in the case of high dimensional state estimation problem. Referring to the existing literatures [71, 75, 143], there are mainly two ways to select the optimal proposal distribution that brings the lowest variance of weights.

The first case is that s_t is a member of a finite set. In such case, the integrals in Equation 2.4 becomes a summation, and therefore sampling from $p(s_t \mid s_{t-1}(i), z_t)$ is possible.

Another feasible method is the analytic evaluation by assuming the distribution $p(s_t \mid s_{t-1}(i), z_t)$ is Gaussian [71, 75, 143], which indicates the dynamics are nonlinear and the measurements are linear. Such a system can be represented by Equation 4.2 and 4.3:

$$s_t = f(s_{t-1}) + v_t^{tran} \tag{4.2}$$

$$z_t = h(s_t) + v_t^{measure} \tag{4.3}$$

and $f : R^{M_x} \to R^{M_x}$ is a nonlinear transition function, observation $h : R^{M_x} \to R^{M_x}$ is a linear function, and v^{tran} and $v^{measure}$ are mutually independent.

We can obtain the proposal distribution with the assumption of Normal Distribution as Equation 4.4 and 4.5:

$$p(s_t \mid s_{t-1}, z_t) = N(s_t; m, \Sigma) \tag{4.4}$$

and

$$p(z_t \mid s_{t-1}) = N(z_t; H(f(s_{t-1})), Q + HRH^T) \tag{4.5}$$

Although in reality, the assumption of analytic evaluations cannot be held in most cases, it is possible to construct suboptimal approximations to the optimal proposal distribution by using local linearization techniques [143], or using unscented transform [75], both of which have been introduced in Chapter 2.

Finally, in application, it is often convenient to choose the proposal distribution to be the prior $p(s_t \mid s_{t-1}(i))$

$$q(s_t \mid s_{t-1}(i), z_t) = p(s_t \mid s_{t-1}(i)) \tag{4.6}$$

We obtain the weighting function with Equation 4.7

$$w_t(i) \propto w_{t-1}(i) p(z_t \mid s_t(i)) \tag{4.7}$$

This would seem to be the most common choice of proposal distribution since it is intuitive and simple to implement.

As discussed in Section 2.3, it is necessary to have a better proposal distribution therefore we attempt to develop an algorithm to determine the proposal distribution rather than the fixed model based construction methods. A common measurement of the difference between two distinct probability distributions *P* and *Q* is often referred to Kullback–Leibler Divergence (K-L divergence) [144]. As a non-symmetric measure, it represents the expected number of extra bits required to code samples from *P* when using a code based on *Q*, rather than a code based on *P*. Typically *P* represents the “true” distribution data, observations, or a precise calculated theoretical distribution. The distribution *Q* typically represents a theory, model, description, or approximation of *P*.

For probability distributions *p* and *q* of a discrete random variable the K-L divergence of distributions *Q* from *P* is defined as Equation 4. 8

$$D(p \mid q) = \sum_{i=1}^{n} p(i) \log \frac{p(i)}{q(i)} \tag{4.8}$$

where *n* samples are randomly drawn from these two distributions. From the above equation, we can anticipate that ***if two distributions are identical, the K-L divergence between them is zero***.

Suppose that we know the optimal distribution, and try to approach our proposal distribution to the optimal one by minimizing the K-L divergence. In such a situation, however, there are two different types of distributions in our problem; the optimal one is the continuous distribution while the other is discrete. To let the comparison becomes “equal”, we have to accomplish the following two procedures:

(1) Converting the continuous distribution into discrete distribution with a number of

random samples taken in the state space.

(2) Converting the discrete distribution into another discrete distribution; the converted distribution has the identical samples with the distribution from (1).

For the first problem, the conversion method works as follows:

1. Generate a random number from the standard uniform distribution; call this x_i;

2. Compute the value *probability density* from the *pdf* such that $f(x) = p$; call this p_i.

3. Normalize all p_i.

For example, we need to convert a simple normal distribution $p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ in $[0,1]$ with the interval 0.1, so we compute the following probability p_i:

$$\begin{aligned} p_i &= \{p_1, p_2, ..., p_{11}\} \\ &= \{p(0), p(0.1), ... p(1)\} \\ &= \{0.3989, 0.397, 0.391, ..., 0.242\} \end{aligned} \tag{4.9}$$

After normalization, the probability p_i becomes:

$$p_i = \{0.1069, 0.1064, 0.1048, ... 0.0648\} \tag{4.10}$$

When we turn to the second problem of adjusting the discrete distribution to another fixed sample discrete distribution, normally, a first conversion has to be made from discrete distribution to continuous distribution, and then a second conversion is made from continuous one to discrete one similar to the process presented in the previous problem. A more direct method will be presented in the following, which is adopted from the interpolation methods.

One of the simplest interpolation methods is linear interpolation. Generally, linear interpolation [145] takes two samples, say (s_a, y_a) and (s_b, y_b), and the interpolant is given by:

$$p = p_a + (y - y_a)\frac{(y_b - y_a)}{(s_b - s_a)} \text{ at the sampling point } s_a \tag{4.11}$$

For example, we have the following discrete probability density:

$$[x_i, p_i] = \{[0.1, 0.3], [0.3, 0.4], [0.4, 0.5], [0.8, 0.9], [0.9, 0.7], [1, 0.5]\} \tag{4.12}$$

From the above definition, we can get more sampling points as follows:

$$\begin{gathered}[x_i, p_i] = \{[0.1, 0.3], [0.2, 0.35], [0.3, 0.4], [0.4, 0.5], \\ [0.5, 0.6], [0.6, 0.7], [0.7, 0.8], [0.8, 0.9], [0.9.0.7], [1, 0.5]\}\end{gathered} \tag{4.13}$$

A further normalization also is needed and it is similar to the process in Equation 4.10.

Having two methods for conversion, we attempt to determine the solution of this optimization problem in the following.

Since the sampling method is obtained from the idea of interpolation methods, let us recall the error analysis of linear interpolation governed by Equation 4.14:

$$|q(s) - p(s)| \le C\left(s_b - s_a\right)^2 \text{ where } C = \frac{1}{8}\max|p''(x)| \tag{4.14}$$

where the function *p()* denotes the distribution function which we want to achieve and its second derivative exits, and s is the sampling point between s_a and s_b . According to Equation 4.14, the error is proportional to the square of the distance between the sampling points. Moreover, if the second derivative function, $p''(x)$, equals to zero, the error caused from linear interpolation will approach zero and this becomes our Theorem 4.1.

Theorem 4.1: Assuming that we need to approach an unknown distribution through a set of discrete sample distribution, the desired size of samples in different interval of $p(s)$ is proportional to $p''(x)$, saying:

$$M_{eff}^{p(a,b)} \propto \left| p''(a,b) \right| \tag{4.15}$$

According to Theorem (4.1), for example, in the situation that a Gaussian distribution, represented by Equation 4.16

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right\} \tag{4.16}$$

Then its second derivative is equal to Equation 4.17:

$$p''(x) = \frac{1}{\sigma\sqrt{2\pi}} \left\{ -\frac{1}{\sigma^2} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right] + \frac{1}{\sigma^2}\left(\frac{x-\mu}{\sigma}\right)^2 \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right] \right\} \tag{4.17}$$

In the situation when the mean $\mu = 0$ and the standard deviation $\sigma = 1$, the probability density function becomes Equation 4.18:

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2} \right) \tag{4.18}$$

So that its second derivative is represented by Equation 4.19:

$$p''(x) = \frac{1}{\sqrt{2\pi}} (x^2 - 1) \exp(-\frac{x^2}{2}) \tag{4.19}$$

To better understand the situation, the comparison of normal distribution function (Equation 4.18) and its second derivative function (Equation 4.19) is shown in Figure 4.1.

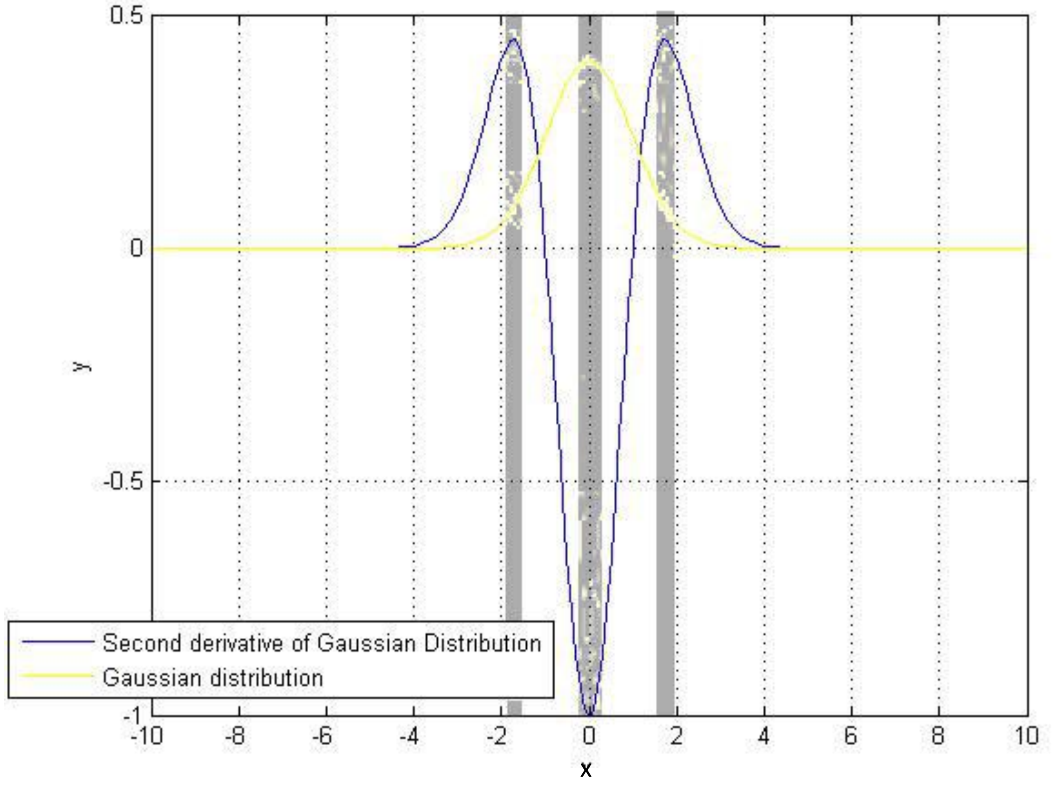


Figure 4.1 The Comparison of Normal Distribution Function and its Second Derivative Function From the second derivative of Gaussian distribution, we conclude that more samples are needed near the top of Gaussian distribution (the gray areas).

As shown in Figure 4.1, we can see that the second derivative reaches the highest in absolute value within the gray area, where the Gaussian distribution value reaches the highest and (almost) the lowest. Considering Theorem 4.1, we need more samples within these intervals if we wish that the discrete distribution approaches the Gaussian distribution based on the measurement of K-L Divergence. Moreover, considering the second derivative, the value in red highlighted central area (equals to 1) is more than twice of that in the marginal areas (smaller than 0.5). Thus we only need to take samples in the high probability area, i.e. the central gray area.

Given the relationship between the Gaussian distribution and its discrete sampling distribution, the method of construction of a sampled Gaussian distribution and elimination of particle impoverishment is straightforward and we have to move some samples around the high likelihood area based on their weights. Since most of the sensor noise are based on Gaussian distribution, or mixed Gaussian distribution, it also provides us a technique to determine characteristics of different sensors, for which more samples are always needed around the mean to construct the curve with higher second derivative rather than the linear line, no matter the noise is following a Gaussian or a non-Gaussian distribution.

4.3 ACO Solution under Combinatorial Optimization

Since our objective is to drive the proposal distribution to the optimal solution under the non-Gaussian situation and the actual model of distribution is not known beforehand. So the traditional methods, the extended Kalman Particle Filter and Unscented Particle Filter, which are both based on a fixed model are just approximately approaching the optimal solution but probably will never identical to the optimal one.

In addition to the above two methods, we propose another approach for constructing an optimal proposal distribution, in which we imagine that the particles can “move” in the state space so that better distribution can be achieved through the existing optimization methods. Referring to the formulation of the combinatorial optimization problem framework given in Section 3.2, the optimal proposal distribution problem being considered can be classified as a combinatorial optimization problem, whose details are given below.

$$Min(D(q(s_t(i) \mid s_{t-1}(i), z), p(s_t \mid s_{t-1}, z))$$

$$s.t. \begin{cases} \sum_{i=1}^{M} w(i) = 1 \\ w_t(i) = \mu \dfrac{p(z_t \mid s_t(i)) p(s_t(i) \mid s_{t-1}(i))}{q(s_t(i) \mid s_{t-1}(i), z_t)} \end{cases} \tag{4.20}$$

where *D()* is the K-L divergence between two distributions.

Because this problem when the model is not known in advance, heuristics are usually considered to be one of the possible solutions. In the following, we will attempt to apply ACO to this combinatorial optimization problem.

To solve this problem, details of the PF$_{ACO}$ algorithm based on the ACO framework are listed in the followings.

1) Construct Ant Solutions: To optimize the generic PF, we adopt ACO before the updating step in PF. An ant replaces the randomly-generated particle in the Sequential Monte Carlo concept. So we assume that we have *m* artificial ants to construct the distribution and the ants are initially distributed from the prior information (4.14), incorporating the transition information. Therefore, to build a more optimal candidate solution set, we need to feed measurement information into the distribution; the weighting function, which is related to the measurement, is also applied to evaluate the stochastic search performance as well as terminating the search when certain criterion in measurement is met. As the main algorithm of ACO, Ant System runs with a stochastic search as governed by Equation 4.21 under the constraints from the set $N(s^P)$ to maximize the evaluation function, so as to fulfill Theorem (4.1).

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]_{\alpha}[\eta_{ij}(t)]_{\beta}}{\sum_{s \in \text{all particles}} [\tau_{is}(t)]_{\alpha}[\eta_{is}(t)]_{\beta}} \tag{4.21}$$

where the term α represents pheromone value, $\tau_{ij}(t)$ denotes pheromone value corresponding to trail connecting *ant j* and *ant i;* *β* represents the heuristic information, which is defined as the reciprocal of the distance of every two particles as defined in Equation 4.22:

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \tag{4.22}$$

Note that the velocity of movement is defined as a random number between zero and the difference of the original point and the target point. It will terminate until all particles' positions converge to the high likelihood probability region (the general or local optimal solution) within a certain threshold, defined by Equation 4.23.

$$Threshold = \frac{constant\ va}{number\ of\ pa} \tag{4.23}$$

2)Update Pheromone: The pheromone value of a traversed trail is enhanced by a constant value, others are reduced according to the evaporation rate.

The updating rule of evaporation value is given in Equation 4.24. As we already have $\tau_{ij}(t)$, without loss of generality, let us define *ant j* is the one intending to move during the iteration and *ant i* is the potential moving target. Note that the candidate partial solution may lie in the trail, so this move is also considered as a kind of stochastic search. Referring to Equation 4.24, pheromone is updated during this iteration according to the target of the stochastic move. In the meanwhile, evaporation is carried out all the time together with either enhancement with the value $\Delta\tau_i(t)$ (defined as the weight of *ant i* at iteration *t*) or nothing as governed by Equation 4.24:

$$\begin{cases} \tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) & \text{j} \in \text{set of particles lie in the movement path} \\ \tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) & \text{j} \in \text{set of other particles} \end{cases} \tag{4.24}$$

where $0 < \rho \leq 1$ is the evaporation rate, $\Delta\tau$ is an enhanced value equals to the weight of *ant i*.

A pseudo-code describing the optimization algorithm is given in Algorithm 4.1

```
Algorithm 4.1: The ACO improved PF Alogorithm
Function PF$_{ACO}$
While the distance between particles and their targets are not within a certain threshold governed by Eqn. (4.31)
---Choose particle $i$ whose distance is within the threshold
---Select the moving target based on the probability Eqn.(4.29)
---Move towards the target with a velocity
---Update the parameters (e.g. pheromone value, heuristic information) of the ACO, and particle weight
End While
```

This ACO algorithm converges when P_{ij} approaches one [146], so it implies that the particle i definitely re-locates to a closer proximity of particle j. When this convergence holds during each iteration, most particles converge to this particle j, which is represented as the neighborhood of a higher likelihood (higher weights) based on Equation 4.21. In this process, two parameters determine the relative influence. If $\alpha = 0$, all particles choose to remain in their original positions so the algorithm degenerates to a generic PF; if, $\beta = 0$, particles tend to move towards neighborhood around higher likelihood, so the distribution is approaching $p(z^t | s^t)$.

Optimized by the ACO, the particle impoverishment problem is basically alleviated because it continuously compares the evaluation function, or the measurement, in every iteration, as well as it tends to stay near to its original position (the prior).

Therefore, the particle samples have a tendency to be around high mixture likelihood regions. As a result, most of the particles which are scattered far away from the true state will converge to states that represent high probability as shown in Figure 4.2. Therefore, when configured with suitable parameters (ρ, τ_0, constant value in threshold, etc.), ACO has the ability to balance between the diversity and the impoverishment of PF.

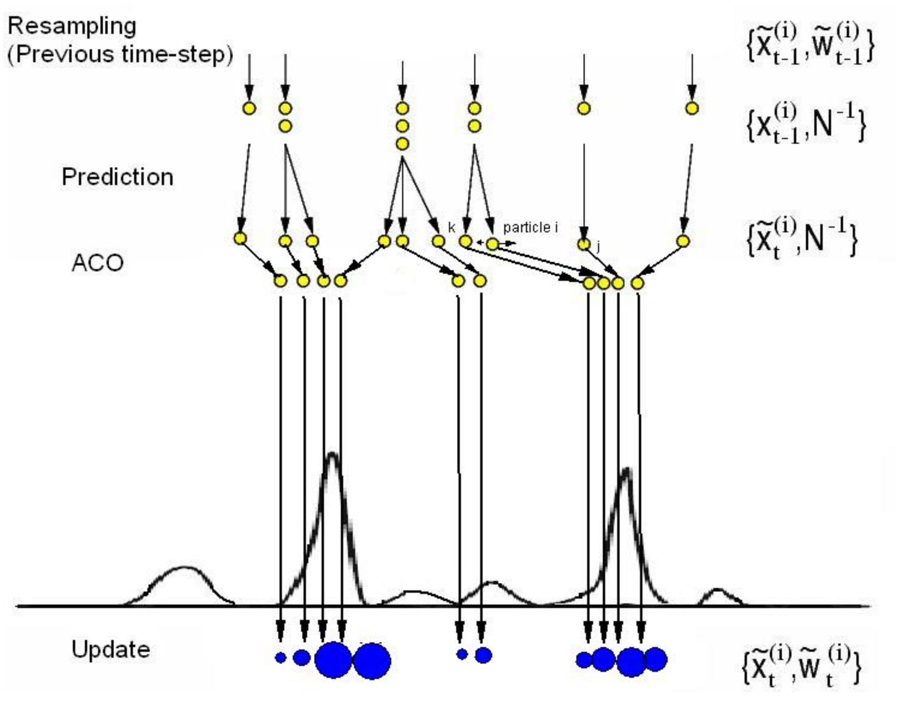

Resampling
(Previous time-step)
$\{\tilde{x}_{t-1}^{(i)}, \tilde{w}_{t-1}^{(i)}\}$
$\{x_{t-1}^{(i)}, N^{-1}\}$
Prediction
k
particle i
j
$\{\tilde{x}_{t}^{(i)}, N^{-1}\}$
ACO
Update
$\{\tilde{x}_{t}^{(i)}, \tilde{w}_{t}^{(i)}\}$

Figure 4.2 ACO Improved PF

Because particle j as a moving target has higher weight and shorter distance than the other particles, p_{ij} (denoted by the length of arrow) is larger than other probabilities. Therefore particle i moves towards particle j.

4.4 Conclusions

The construction of the proposal distribution is analyzed and reformulated in the combinatorial optimization framework. As a metaheuristic method to combinatorial optimization problems, ACO is applied to optimize the proposal distribution. In the next chapter, case studies and proof of the ACO improved PF are given.

Chapter 5 Analysis of $\mathrm{PF_{ACO}}$

5.1 Introduction

In Chapter 4, a biologically inspired method, the ACO algorithm, to optimize the particle distribution has been proposed so that particles can better approximate the optimal proposal distribution. Advantages of the proposed method are:

1) the near-optimal proposal distribution will lead to smaller variance of weights, so that the efficiency of particles can be maintained;
2) it is suitable for implementing in multi-robot systems by considering the system's characteristics as stated in Section 4.2.

In this chapter, a detailed study, including a comparison between different types of PF and a variance of weights comparison for cases with extreme variance values, will be presented in order to analyze the performance of ACO improved PF; in addition, a mathematical proof on the ACO improvement PF by measuring the K-L Divergence [140] is also given in the latter part of this chapter.

5.2 Case Studies

5.2.1 Single Variable Estimation

A nonlinear single variable economic model, given in Equations 5.1 and 5.2, adopted from [75] was employed to compare results of various methods, including Extended Kalman Filter[147], Unscented Kalman Filter [37, 148], generic PF (PF with Sequential Importance Re-sampling) [37], PF with EKF Proposal [37], PF with UKF proposal [87] and PF with ACO improvement.

$$s(t+1) = 1 + \sin(4\times10^{-2}\pi t) + 0.5s(t) + w(t) \qquad (5.1)$$

$$z(t)=\begin{cases}\dfrac{s(t)^2}{5}+u(t),\ \mathrm{t}\le 30\\ -2+\dfrac{s(t)}{2}+u(t),\ \mathrm{t}>30\end{cases} \tag{5.2}$$

where *w(t)* stands for the zero-mean white noise, and *u(t)* stands for noise with Gamma distribution[149], where the variance of *w(t)* is 1×10^{-5}, and the two parameters of Gamma distribution, *k* and *θ*, equal to 7 and 2, respectively.

From *t*=1 to 60 in a single test run, given the noise measurement, the state sequence s_t was applied to all filtering methods. In order to minimize the effect of randomness, all experiments included 30 runs. In all PF, the number of particles used was 200. Table 5.1 and 5.2 show the mean and variance of the Root Mean Square Error (RMSE) obtained from different PF algorithms. Figure 5.1 depicts the RMSE for different algorithms.

Filters	RMS Error	RMSE Percentage (EKF=100%)
EKF	0.98087	100
UKF	0.68237	69.57
Generic PF	0.77918	79.44
PF+MCMC	0.79492	81.04
PF+EKF	0.95391	97.25
PF+UKF	0.3792	38.66
PF+EKF+MCMC	0.95354	97.21
PF+UKF+MCMC	0.39387	40.16
PF+ACO	0.28153	28.70

Table 5.1 RMS value of error

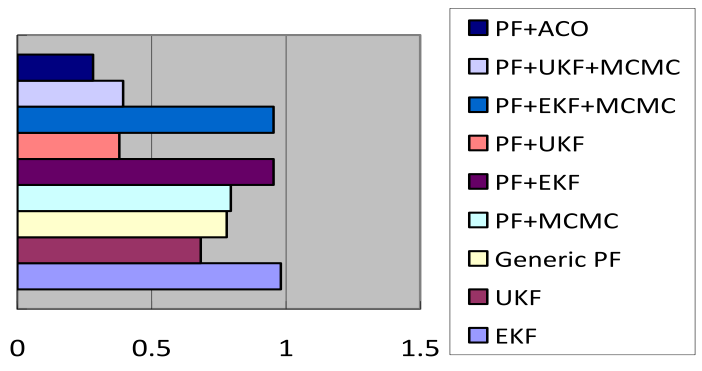


Figure 5.1 Comparison of RMSE from different filters

Filters	Variance
EKF	0.059334
UKF	0.029767
Generic PF	0.054233
PF+MCMC	0.041409
PF+EKF	0.044244
PF+UKF	0.021977
PF+EKF+MCMC	0.049126
PF+UKF+MCMC	0.01669
PF+ACO	0.001619

Table 5.2 Variance of RMS Error

From the above tables and figure, we can conclude that our PF_{ACO} can produce the best results in term of error level. In addition, the variance of our method is also the smallest comparing to other methods, implying that the PF_{ACO} can give more stable performance according to [150].

The average execution time in each run was measured and given in Table 5.3.

Filters	Time (Sec)
EKF	0.53321
UKF	0.92803
Generic PF	0.89604
PF+MCMC	1.95805
PF+EKF	5.9355
PF+UKF	11.68094
PF+EKF+MCMC	10.06142
PF+UKF+MCMC	22.88661
PF+ACO	3.18547

Table 5.3 Execution Time of Filters

According to Table 5.3, the PF_{ACO} requires a longer computational time than the Kalman Filters and generic PF, which generally will give a larger estimation error. But compared to the similar PF with an improved proposal distribution, the extended Kalman Particle Filter and Unscented Kalman Particle Filter (both with and without MCMC), PF_{ACO} takes a shorter execution time.

The following figure represents the estimation results from all filters of one run, which shows that PF_{ACO} can track the estimation accurately throughout the whole experiment.

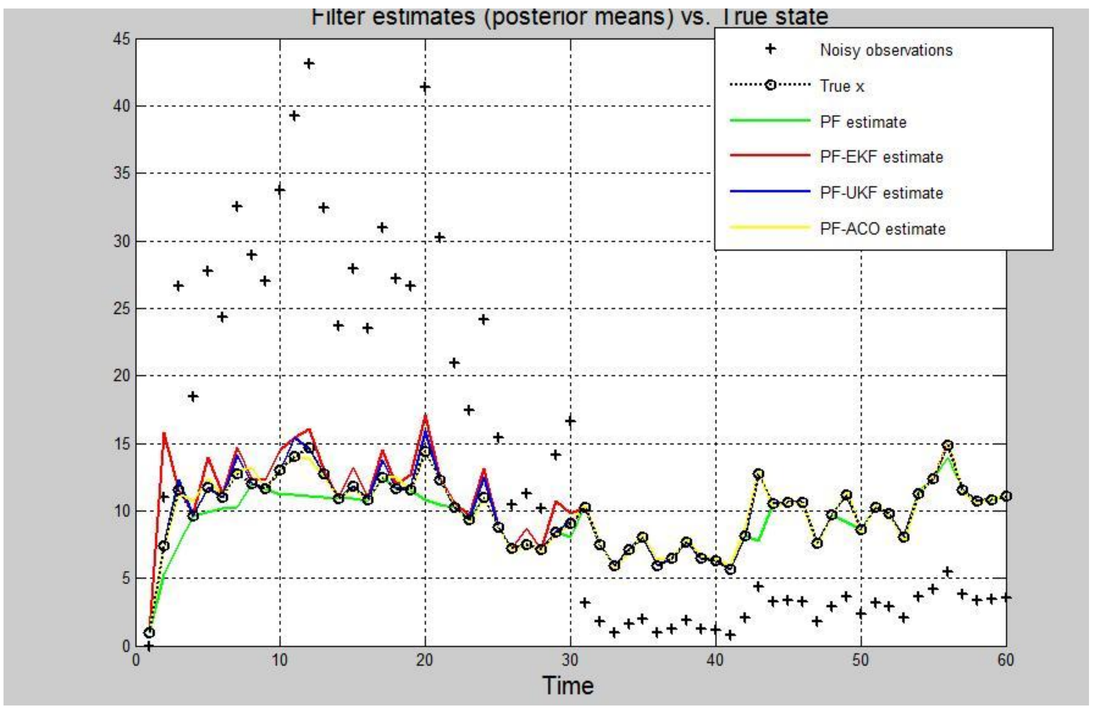
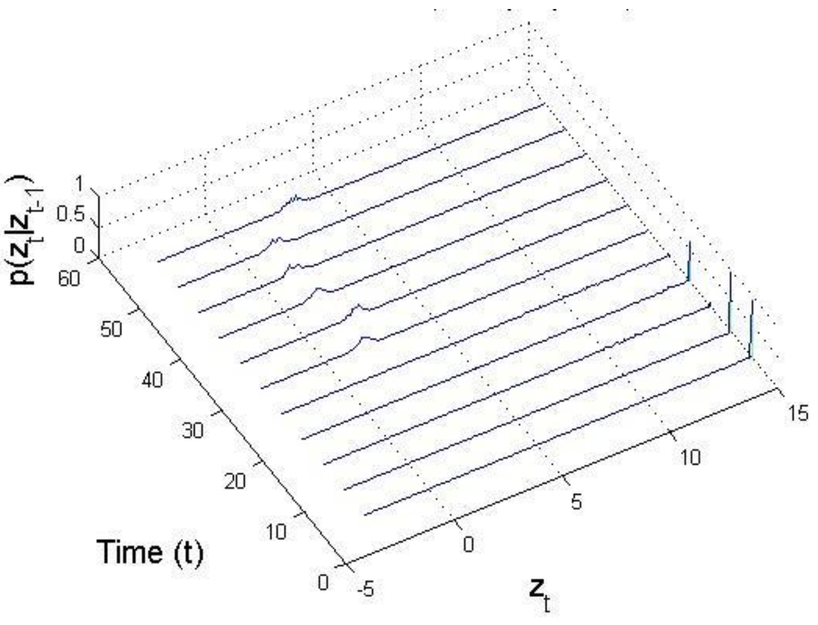


Figure 5.2 The diagram of different PF tracking result

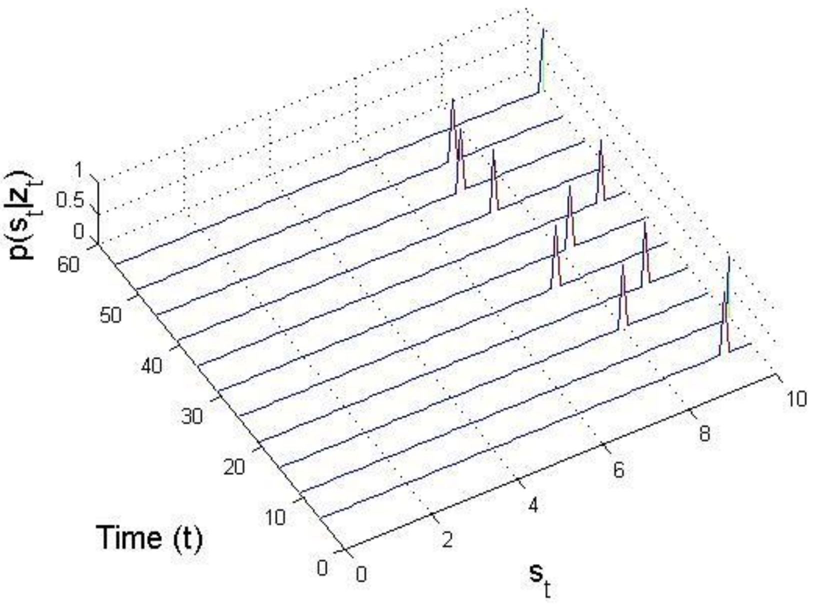
The ACO improved PF performs better than other PFs in the single variable estimation test. Besides, the extended Kalman Particle Filter performs worse than generic PF, especially from time interval 1 to 30, it may be caused from the transition function is with the white noise, which EKF probability cannot track quite well, even compared to the generic PF. After the time 30, with another measurement function, almost all PFs perform quite well.

Before we present the conditional distribution details of various kinds of filters as shown in Figure 5.3(c), an example to demonstrate the conditional distributions $p(z_t \mid z_{t-1})$ and $p(s_t \mid z_t)$ are depicted in Figure 5.3 (a) and 5.3(b). In these two examples, the curves parallel to the z_t or s_t axis denote the distribution at a specific time, i.e. t=1 to 60. So these two figures show the measurement and the posterior distributions from the generated particles. In the measurement demonstration (Figure 5.3a), a sharper distribution is preferred because it represents measurements from particles concentrate in a true measurement. In the posterior distribution (Figure 5.3b), it is desired that particles are distributed with a mixed-Gaussian model, especially in cases including a non-Gaussian estimation applications.

Figure 5.3c includes conditional distributions sequence obtained from different proposal distribution construction methods for solving the previous single variable estimation problem. The first row represents distributions of different PF, from which we can observe the different proposal distribution construction methods leading to different predicted measurements. The two distributions with a sharper peak (UKF and ACO proposal distributions), referring the accurate measurement predictions with smaller variances. Moreover, the second row in the figure shows different posterior distributions $p(s_t \mid z_t)$, representing the state distribution based on certain measurement. Unlike the measurement probability distribution, some non-Gaussian distributions, such as mixed distribution, may occur during the experiment. However, $\mathrm{PF}_{\mathrm{ACO}}$ provided the possibility to construct a mixed distribution with the particles, and it is useful in robot localization without the prior information of an initial position; the other three PF merely introduced the posterior to Gaussian distribution.

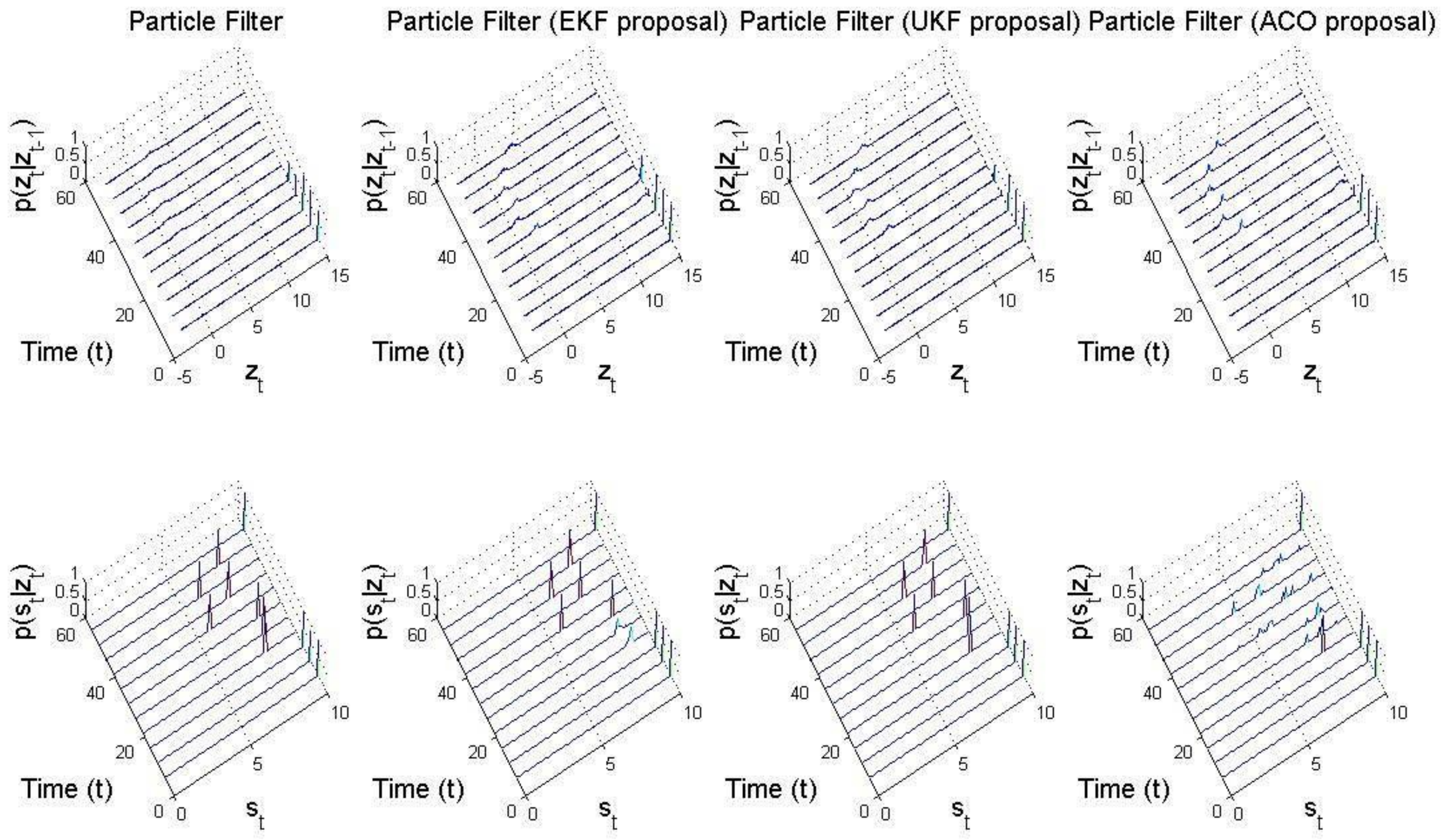


(a)



(b)

Figure 5.3 (a)(b)
Sampled distributions in a run



(c)
Figure 5.3
Sampled distributions in a run

The particles from EKF and UKF have similar distributions, while the particle measurement from PF_{ACO} concentrates sharper. The second row shows that our PF_{ACO} provides us a possibility to demonstrate the mixed-Gaussian distributions.

5.2.2 Single Robot Localization

In order to evaluate the performance of the PF_{ACO} when applied to robot's localization problem, it was implemented using a Matlab program originally developed by Vale [151] and modified by the author (Figure 5.4).

This Matlab program has the ability to simulate a mobile robot movement and its noisy measurement from a SICK laser sensor maneuvering within a map described using a matrix file. In Figure 5.4a, blue particles on the right chart represent the hypothesis of positions from generic PF, while the green particles from the PF_{ACO}. When the estimation was in progress, the blue and green particles would be re-drawn continuously as in Figure 5.4a and weights of each particle would be plotted (Figure 5.4b). The noisy observation of sensors is shown in Figure 5.4c. In the following experiment, the generic PF and PF_{ACO} were applied to estimate the robot pose (including the position and orientation) along with a series of points. (Asterisks in the Figure 5.5)

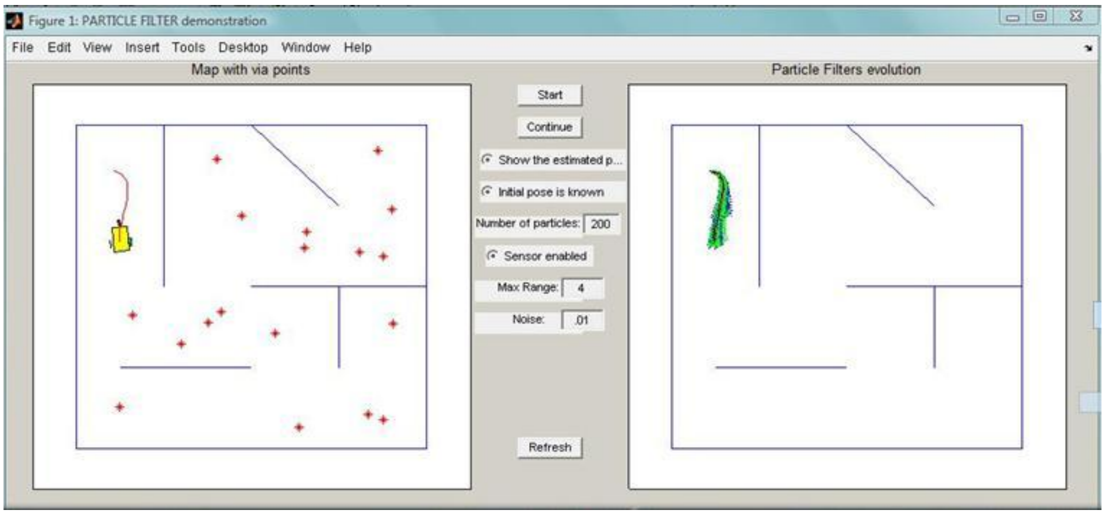


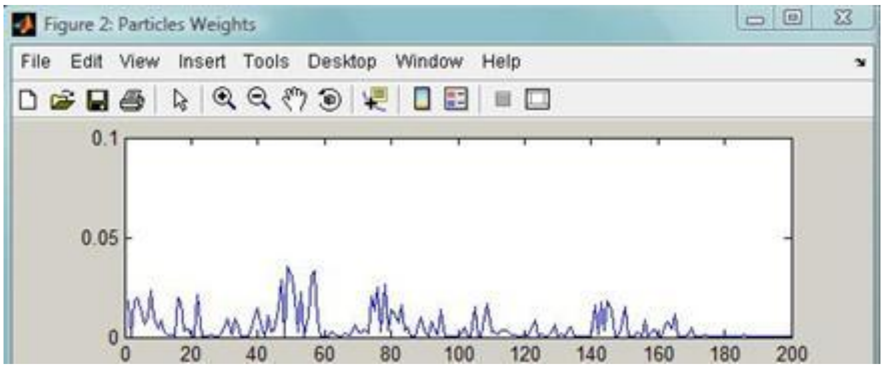
Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Start
Continue
Show the estimated p...
Initial pose is known
Sensor enabled
Refresh
Particle Filters evolution

Number of particles: 200

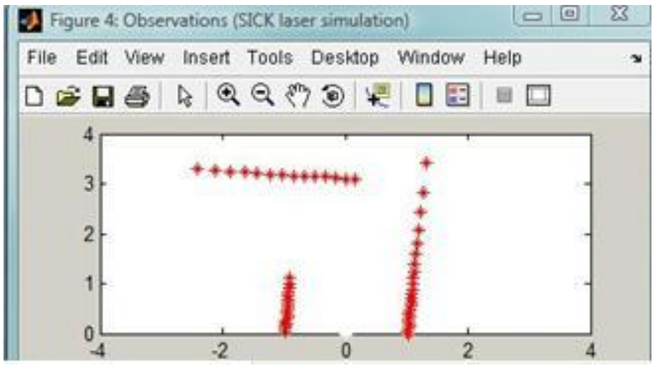
Max Range: 4

Noise: .01

(a)



(b)



(c)

Figure 5.4 Interface of Particle Filters in Robotic Localization Demonstration
In Figure 5.4a on the top, the left map shows the true map and the trajectory of the robots' navigation (red curve) along the asterisks set in advance, and the right map is the evolution demonstrations of particles representing the robot's pose while the robot is traversing in the area. In Figure 5.4 b, the graphs depict the pose particles' weight distribution of robot. Figure 5.4c shows the observation from the SICK laser sensor equipped in robot.

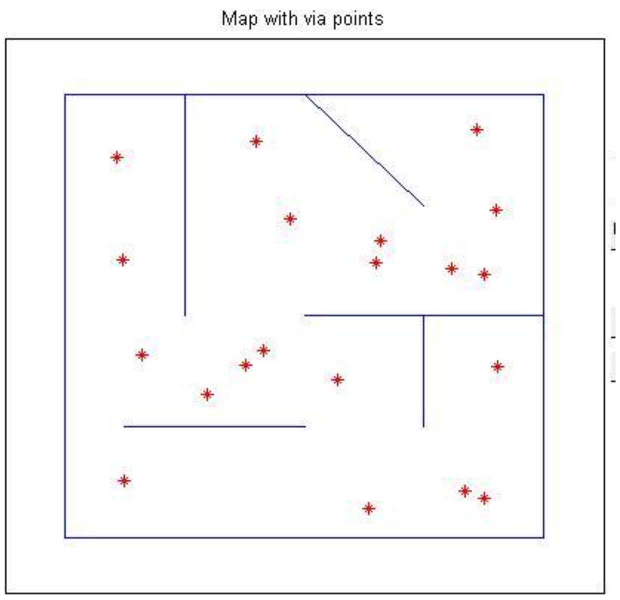


Figure 5.5 Asterisks as the navigation targets

The positions of asterisks are pre-defined and stored in a matrix. Robot will move along these asterisks during the experiments.

The generic PF, extended Kalman PF, Unscented PF, and PF_{ACO} were applied to the same localization problem. Without the prior information of initial poses, neither filter could track the poses accurately at the beginning, so some traces, standing for multiple hypotheses, appear in the particle evolution demonstration at couples of initial running steps. However, after enough information has been obtained, all filters could return to the correct trajectory.

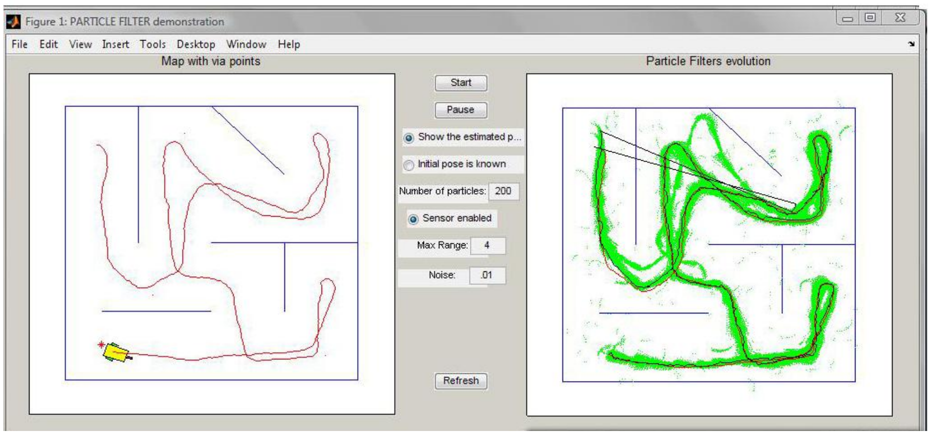


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

[✓ Show the estimated p...]
Initial pose is known
Number of particles: 200
[✓ Sensor enabled]
Max Range: 4
Noise: .01

(a)

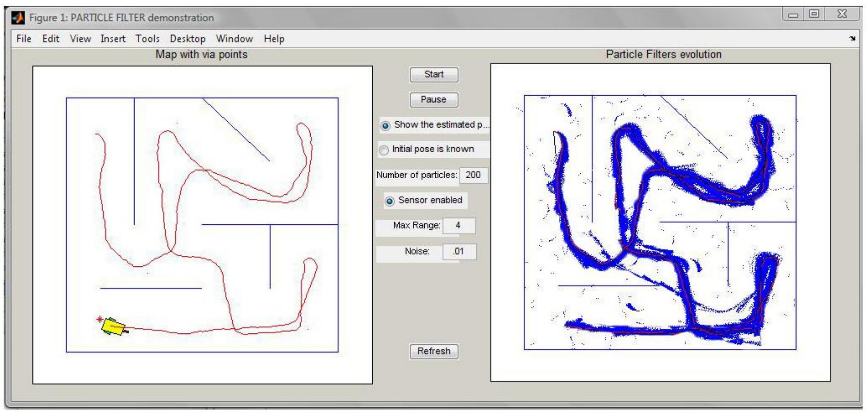


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01

(b)

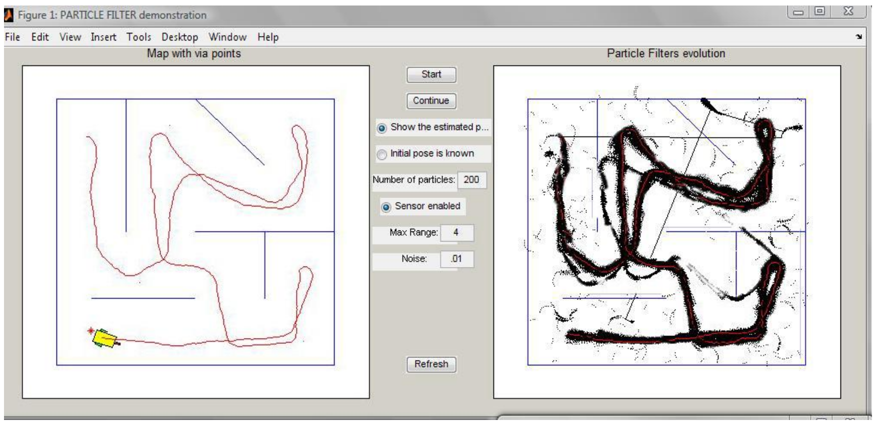


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Continue
Refresh

Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01

(c)

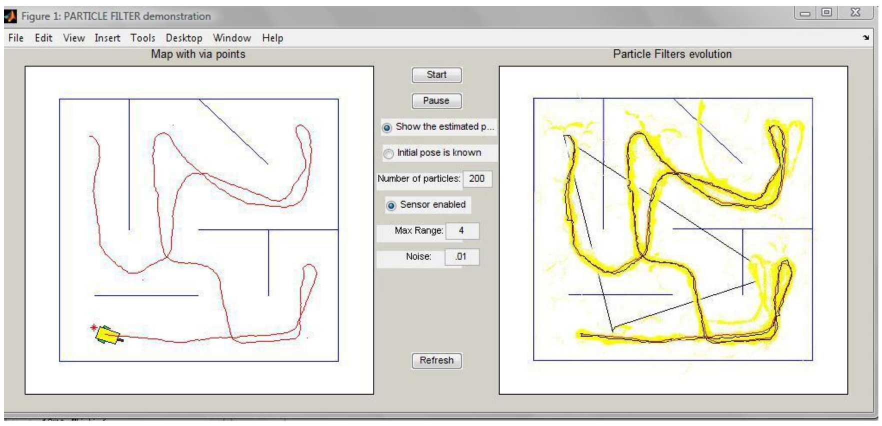


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01

(d)

Figure 5.6 Navigation Progress

(a) The localization result from generic PF, in which the particle deviation happened in upper middle of the map, and lasted for a long running process. (b) The localization result from extended Kalman PF, in which the particle deviation happened in right lower corner of the map. (c) The localization result from Unscented PF, in which the particle deviation happened from the right top corner to the middle lower part.(d) The localization result from extended Kalman PF, in which there are also some deviations. But comparatively, they are shorter than the other three PF methods.

Particle deviation refers to the fact that the particle set deviates from the true trajectory with a significant distance. In mathematics, the deviation makes this particle set negligible in Monte Carlo estimation because these particles will have a relatively small weights. Thus it can be regarded as an explanation of particle impoverishment. Figure 5.6a shows the process obtained from generic PF, and its particle deviation was almost the laragest among the four algorithms. Figure 5.6b and 5.6c were captured from the extended Kalman PF and Unscented PF localization, whose particle evolutions were so similar, and both had particle deviation with almost the same degree. However, the PF_{ACO} comparatively had smaller particle deviation than the other three PF methods.

To evaluate the results quantitatively, we compare the RMS error of the position and orientation of the robot in Table 5.4. The table also shows the percentage error of three different versions of PF using the RMS error of the generic PF as reference.

Filters	RMS Error in Position	Position Error (%)
Generic PF	4.5045	-
Extended Kalman PF	3.2918	73.08%
Unscented PF	2.8605	63.50%
PF+ACO	1.9027	43.24%
	RMS Error in Orientation	Orientation Error (%)
Generic PF	3.7746	-
Extended Kalman PF	2.3207	61.48%
Unscented PF	2.2439	59.45%
PF+ACO	0.8529	22.60%

Table 5.4 RMS Error of Four PF in single robot localization (particle number is 200)

From Table 5.4, it is obvious that PF_{ACO} has smaller errors in robot position and orientation estimation than other three methods using the same number of particles. It is because our ACO plays an important role in the optimization of particle distribution in PF, alleviating particle impoverishment from occurring.

Table 5.5 shows the execution time of the four PF, using 200 particles, and the generic PF took the shortest computation time, and PF_{ACO} took shorter time than the extended

Kalman PF and Unscented PF.

Filters	Execution Time
Generic PF	42.17231
Extended Kalman PF	60.23145
Unscented PF	91.93079
PF+ACO	53.19893

Table 5.5 Execution Time for Four PF (in Sec)

From the above results, the RMS error in all cases of the PF_{ACO} was less than that obtained from the generic PF. Thus a smaller number of particles is needed to maintain the error level, and therefore, PF_{ACO} will make robot localization problem consuming less computational time and in addition, the size dependency problem can be avoided.

In the above experiment, we merely made the estimation on the pose of robot, i.e. the localization of robot, based on the assumption that the robot had already gotten the prior information of a map.

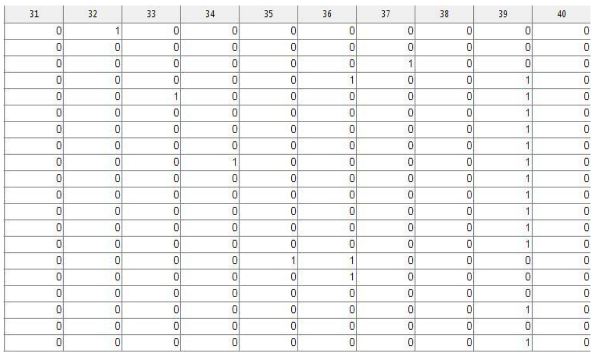
5.2.3 Single Robot SLAM

In this section, using the single robot SLAM experiment platform and map, we attempted to solve the single robot SLAM problem by four PF methods used in the above sections. Since an estimated map will be developed after the experiment, we introduce a way to investigate the grid-occupancy map and results obtained by using different filtering methods are presented in the following paragraphs. In Equation 5.3, it illustrates the method applied in evaluating the error and it is by counting the difference between the true value and estimation value of map grids one by one:

$$E_m = \sum_{i,j=1}^{L} \left| \tilde{m}_{ij} - m_{ij} \right| \tag{5.3}$$

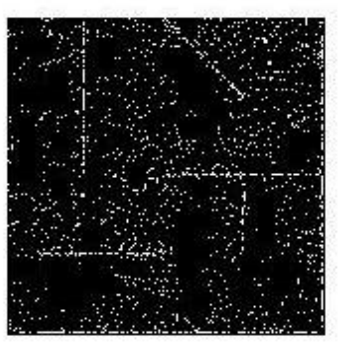
where m_{ij} is the true value of map grid *ij*, $\tilde{m}_{ij}$ is the estimation value of map grid *ij*. Thus the equation simply counts the different grids between the true map and the estimated map one by one.

There is an addition feature which is the mapping function included in the simulation program. The generated map as shown in Figure 5.6 was stored in memory continuously during the mapping process. The software can automatically compare these maps, and calculate their error (Equation 5.3) after the mapping algorithm.



31	32	33	34	35	36	37	38	39	40
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0

(a)

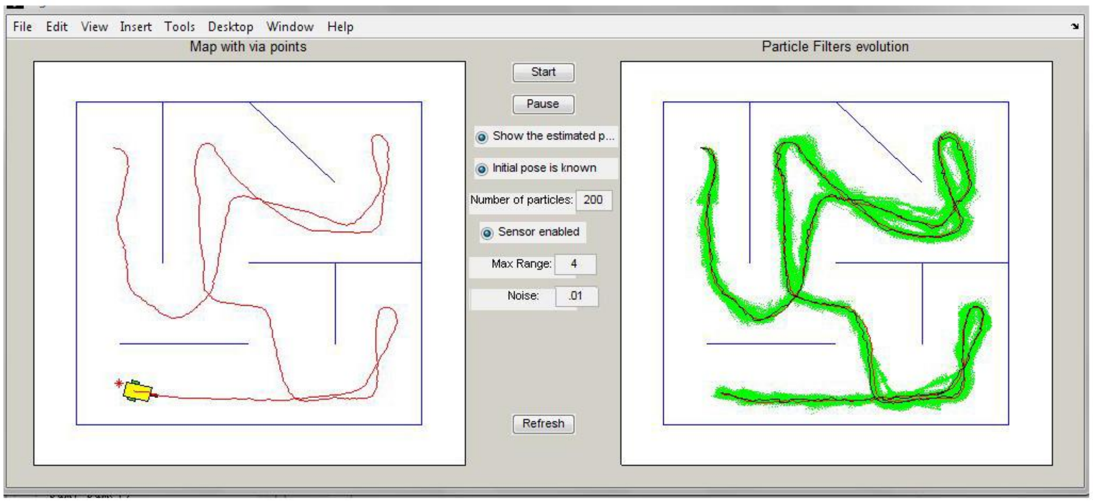


(b)

Figure 5.7 Map Re-draw

Part of the estimated map matrix stored in the memory and the map based on its information

Figures 5.8 and 5.9 illustrate the localization results and mapping results of SLAM obtained from four PF. In Figure 5.6, because the map information was not provided in advance, the multiple hypotheses of robot’s initial pose were not taken in the initial steps, so there were not many simultaneous particle traces representing the hypotheses in the map of the experiments as illustrated in Figure 5.6.


File Edit View Insert Tools Desktop Window Help
Map with via points
Start
Pause
Refresh
Particle Filters evolution

[✓ Show the estimated p...]

[✓ Initial pose is known]

Number of particles: 200

[✓ Sensor enabled]

Max Range: 4

Noise: .01

(a)

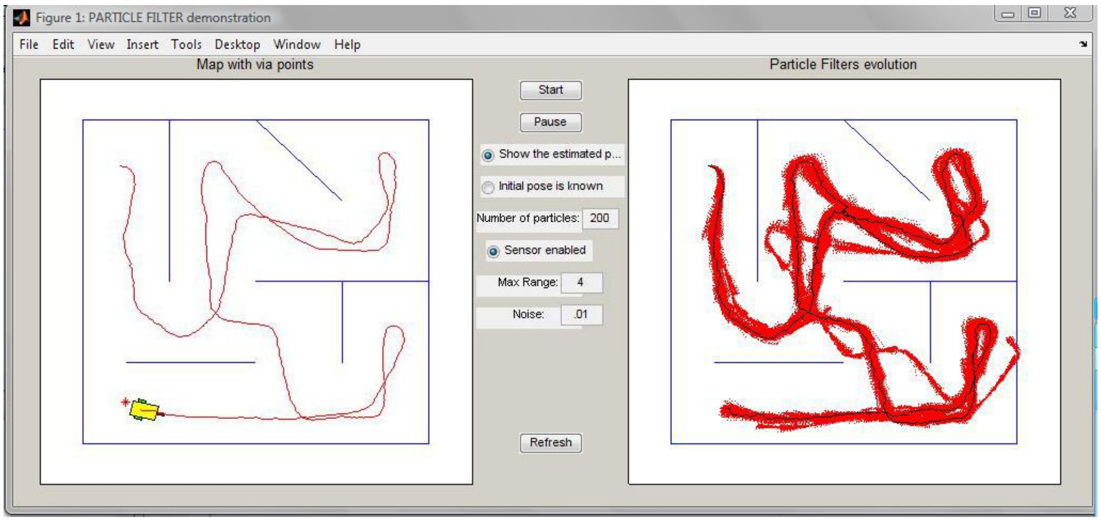


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Start
Pause
Refresh
Particle Filters evolution

[✓ Show the estimated p...]

Initial pose is known

Number of particles: 200

[✓ Sensor enabled]

Max Range: 4

Noise: .01

(b)

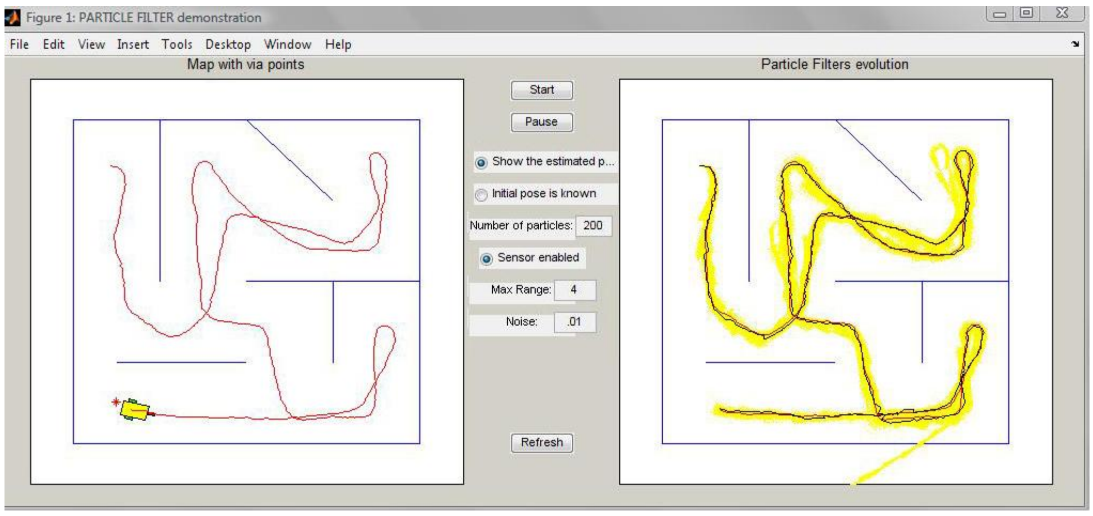


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Start
Pause
Refresh
Particle Filters evolution

[✓ Show the estimated p...]

Initial pose is known

Number of particles: 200

[✓ Sensor enabled]

Max Range: 4

Noise: .01

(c)

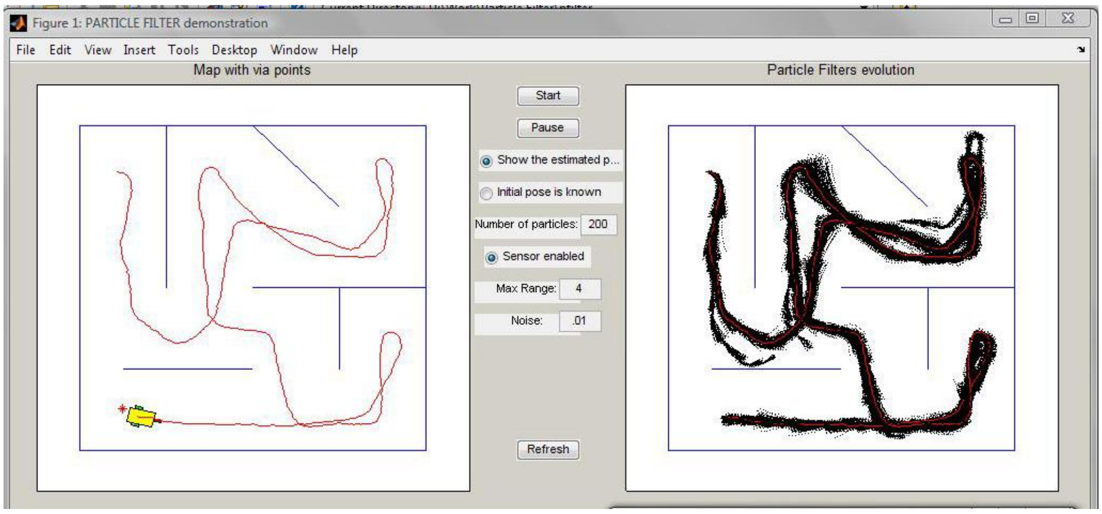


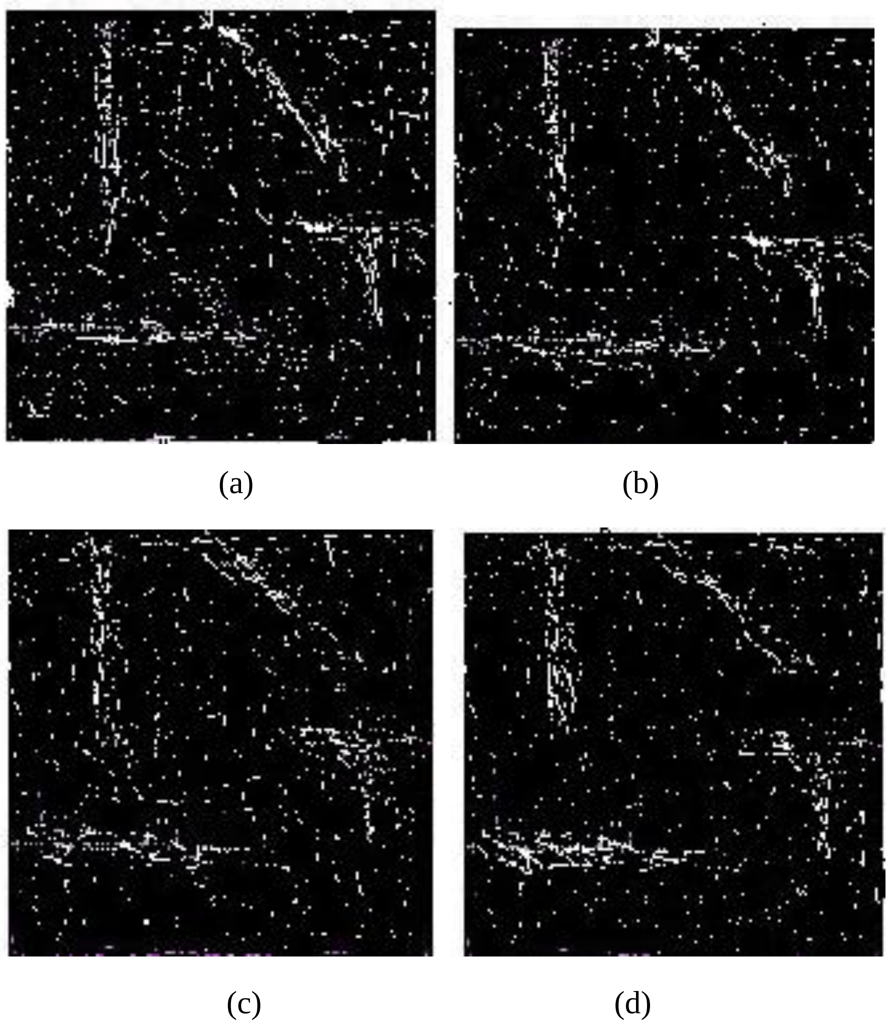
Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01

(d)

Figure 5.8 Single robot Localization results (a) Localization results of generic PF. (b) Localization results of extended Kalman PF. (c) Localization results of Unscented PF. (d) Localization results from PF$_{ACO}$.

The estimated maps as shown in Figure 5.9 are re-drawn from the map grids matrix. We can observe that the localization results depicted in Figure 5.9 are similar to those obtained from single robot localization experiments. The generic PF's mapping results has the largest error, while the "noise" in Figure 5.9b and 5.9c are within the same level. Comparatively, the PF$_{ACO}$ has the smallest errors.


(a) (b)

(c) (d)

Figure 5.9 Single robot mapping (a) Mapping results of generic PF. (b) Mapping results of extended Kalman PF. (c) Mapping results of Unscented PF. (d) Mapping results from PF$_{ACO}$. More error can be found in the generic PF mapping, while extended Kalman PF and Unscented PF are in the same error range.

The following Table 5.6 includes the error and its percentage by comparison, and Table 5.7 shows execution time. Both tables are obtained from single robot SLAM.

Filters	RMS Error in Position	Position Error (%)
Generic PF	17.5435	-
Extended Kalman PF	13.3442	76.06%
Unscented PF	13.7347	78.29%
PF+ACO	7.2483	41.32%
	RMS Error in Orientation	Orientation Error (%)
Generic PF	5.2239	-
Extended Kalman PF	5.6204	107.59%
Unscented PF	3.3570	64.26%
PF+ACO	1.4309	27.39%
	RMS Error in Mapping	Mapping Error (%)
Generic PF	10239	-
Extended Kalman PF	8924	87.16%
Unscented PF	8729	85.25%
PF+ACO	6024	58.83%

Table 5.6 RMS Error of Four PF in single robot SLAM (All particle number is 200)

Filters	Execution Time
Generic PF	494.53507
Extended Kalman PF	664.30649
Unscented PF	703.50325
PF+ACO	574.35045

Table 5.7 Execution Time of Four PF in single robot SLAM (in Sec)

These numerical results are similar to our observation from Figures 5.8 and 5.9. PF_{ACO} has the smallest error in localization and mapping, in both cases, the RMS errors are just half of those obtained from the extended Kalman PF and Unscented PF, also it is much better than generic PF. Similar to the localization problem, PF_{ACO} took shorter time to complete comparing to the other two improved PFs. Since non-linear noise is added to the motion and sensor models, but proposal distributions in generic PF, extended Kalman PF and Unscented PF are only approximations to the optimal proposal distribution, consequently, estimation result from PF_{ACO} is better.

5.3 Variance Improvement in PF_{ACO}

In this section, we will concentrate on the performance of our PF_{ACO} in cases under more extreme conditions by considering the variance of particles' weight. Large variance or small variance of weights in particle sets generated after the transition model (i.e. motion model in robotics application) often lead to diverse results. Therefore, we generated particles with different degrees of variance, so as to further investigate performances of the ACO improved PF under those cases.

We employed another single variable model defined by Equation 5.4 and 5.5.

$$s(t+1) = 6 + \sin(4e - 2\pi t) + 0.5s(t) + w(t) \tag{5.4}$$

$$z(t) = 0.2s(t)^2 + u(t) \tag{5.5}$$

where *w(t)* and *u(t)* stand for the zero-mean Gaussian process noise and measurement white noise respectively. By tuning the variance of *w(t)* and *u(t),* we can generate particles which have weights that give different variances of their weights. Then we manually positioned the particles in almost the same distance from the true value in the state space to tune their weight variance. As a result, their variance almost equals to zero. As stated in Equation 4.21, if particles are distributed with equi-distance from measurement then the probabilities of moving towards other particles are almost the same, that is, their probabilities are averagely low. However, the probability of standstill is higher by setting its default probability in ACO algorithm, so particles will remain in their initial positions. Thus the variance of the whole particle set remains zero. An experiment based on such a situation will be given in this section.
Having a large variance, it represents the situation that particles are so distant from each other. Having differences in weights, with the effort of ACO, particles with smaller weights will move towards ones with larger weights, and then the particle set will end up with a smaller variance for the weights.

The variance of weights was tuned approximately from 0.0001 to 10, with a step size of 0.1 and the improved results are presented in Figure 5.10.

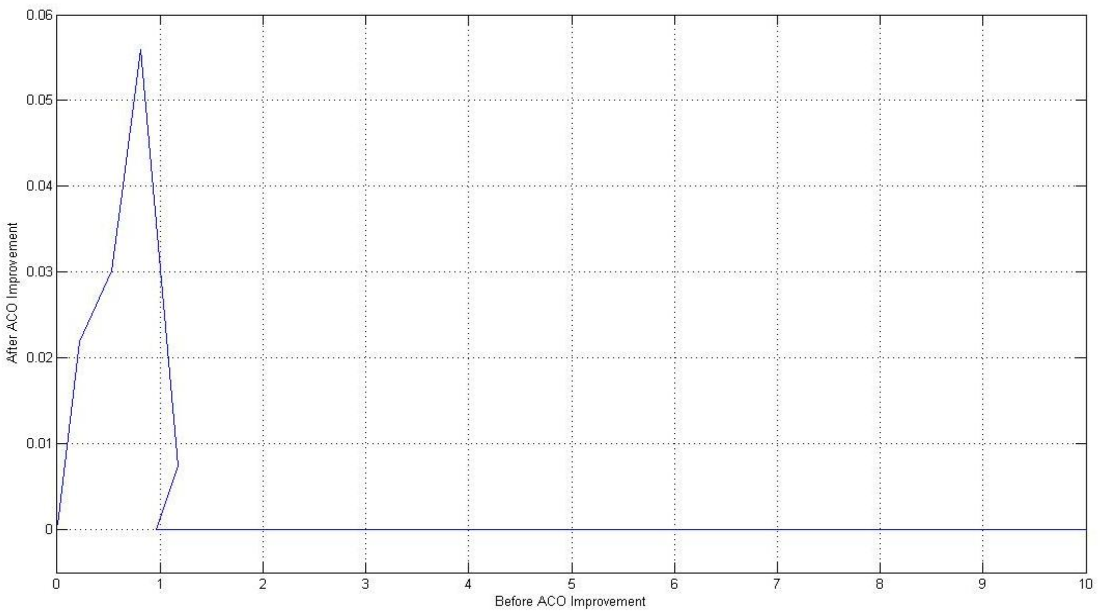


Figure 5.10 The effect of variance change before and after ACO algorithm

The figure shows that the variances above one will always decrease to zero. But the variances between zero and one only decrease to the value between 0.01 and 0.06. Generally speaking, the ACO improves the weight variance of PF.

Before ACO (variance)	After ACO (variance)
0.0001	0.0001
0.2131	0.0218
0.5310	0.0301
0.8150	0.0559
1.1754	0.0075
1.8657	6.4543×10^{-29}
4.2213	6.4543×10^{-29}
7.2852	6.4543×10^{-29}
9.5429	6.4543×10^{-29}

Table 5.8 Cases with Extreme Weight Variance of ACO Improvement

The variance improvement can be illustrated by the changes in variance value before and after the application of the ACO algorithm and results are given in Table 5.8

which represents nine runs of the same experiment with different initial settings of particles.

5.4 Optimization in PF_{ACO}

In this section, we will propose a theorem and its proof, explaining how the PF_{ACO} can produce better solution when compared to generic PF, which employs a transition function as the proposal distribution.

Theorem 5.1: With the convergence of ACO, the PF_{ACO} can always achieve the optimal proposal distribution when the ACO converges to an optimal solution.

Proof: In the generic form, a transition model is often used as the predicted proposal distribution:

$$q(s_t \mid s_{t-1}^i, z_t) = p(s_t \mid s_{t-1}^i)_{tran} \qquad (5.6)$$

However, the optimal one is as follow:

$$q(s_t \mid s_{t-1}^i, z_t)_{opt} = p(s_t \mid s_{t-1}^i, z_t) \qquad (5.7)$$

The second term $p(s_t \mid s_{t-1}^i, z_t)$ in real world presents the probability that moving to state s_t in time *t*, given the samples in previous time step s_{t-1} and the measurement z_t. This generic transition model can approximately equals to this optimal model only if the following two conditions are satisfied.

1) The motion sensor has no error in the motion detection; or
2) the motion sensor noise has similar noise variance as the observation sensor.

Nevertheless, the above two conditions are difficult to achieve in most of our experiments due to the different variances contributed by various sensors' errors. The observation sensors, e.g. laser sensors and vision sensors, are becoming more accurate, but this is not the same case for motion sensors. With different variance level, traditional transition model based on the motion sensors is not as suitable as it used to be, especially in experiments include observation sensors and motion sensors. Figure 5.11 shows a comparison of the two distributions and a mixture distribution of them which is a non-Gaussian distribution.

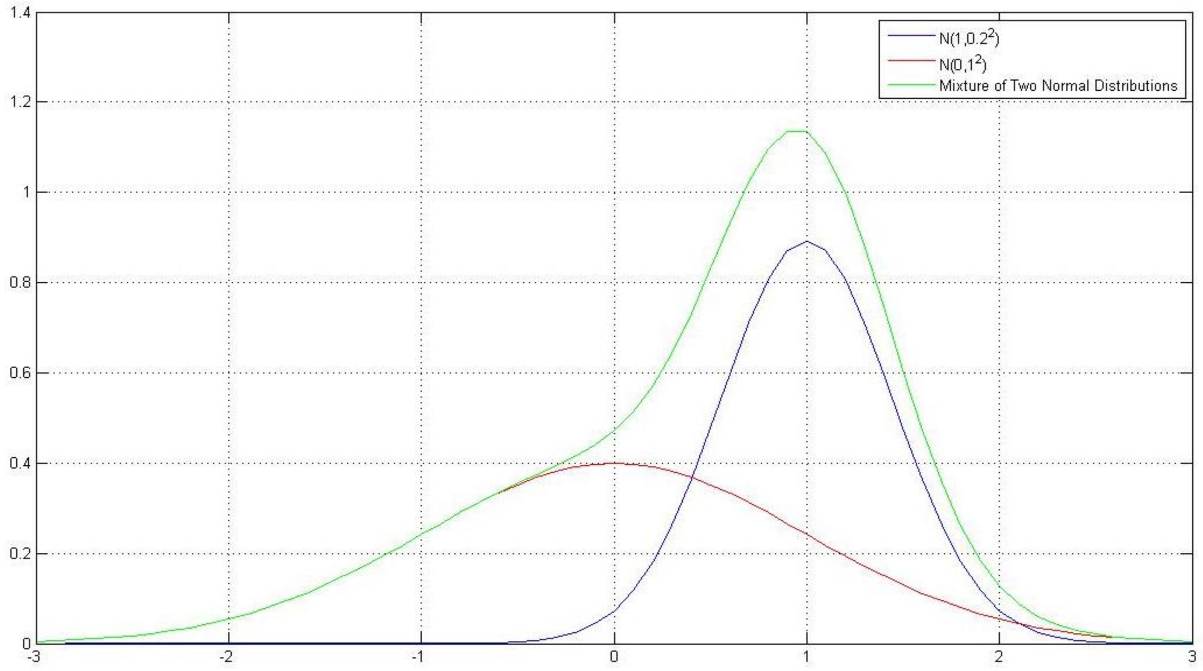


Figure 5.11 A mixture density of two different normal distributions, which having different variance values. We can tell that the mixture distribution is mainly dominated by the smaller variance normal distribution rather than the larger one.

The approximation of K-L Divergence is generated by a set of sample data set: $s_1, s_2, \ldots s_N$, based on the model density *p(s)*, so

$$D(p\,||\,q) \approx \frac{1}{N}\sum_{i=1}^{N}[\log p(s(n)) - \log q(s(n))] \tag{5.8}$$

For the generic PF, the above K-L Divergence equals to

$$D(p\,||\,q) \approx \frac{1}{N}\sum_{i=1}^{N}[\log p(s_t(n)\,|\,s_{t-1}(i), z_t) - \log q(s_t(n)\,|\,s_{t-1}(i))] \tag{5.9}$$

To evaluate the K-L Divergence, we take *N* Monte Carlo samples in state space for s_t, calculate their probability density given the condition of particle $s_{t-1}(i)$ and z_t. Note that two Monte Carlo methods are separated, in which *N* samples (denoted as index *n*) are used to calculate K-L Divergence, and *M* samples (denoted as index *i*) are used to calculate the SLAM posterior (also known as PF).

Recall that the ACO probability that drives the particle to move by the function in Equation 4.21. It is trivial to know from this equation that the ACO algorithm converges if and only if $p_{ij}(t)=1$, which indicates the necessary and sufficient conditions of ACO convergences is $d_{ij}=0$ or $\lim_{t\to\infty}\eta_{ij}(t)\to\sum_{s\in\text{all particles}}\eta_{is}(t)$. These two conditions are corresponding to the high probability density $p(x_t \mid z_t)$ (the blue curve) and $p(x_t \mid x_{t-1})$ (the red curve) as distributions depicted in Figure 5.11. Consequently, majority of particles will be located around the peak of the mixture likelihood density function.

Assuming we take samples $\hat{s}_1,\hat{s}_2,...,\hat{s}_n$ in the optimal proposal distribution, in order to approach the optimal proposal distribution according to the definition of K-L Divergence and our Theorem 4.1, we will derive the relationship of the number of samples and the optimal distribution. If it is necessary to have *M* samples ($\tilde{s}_k,\tilde{s}_{k+1},...,\tilde{s}_{k+M}$) in order to generate *N* samples ($\hat{s}_k,\hat{s}_{k+1},...,\hat{s}_{k+N}$) in the continuous optimal proposal distribution according to Theorem 4.1, the number of samples needed to be considered is proportional to the second derivative of the optimal distribution, which can be illustrated by Figure 5.12 and Equation 5.10:

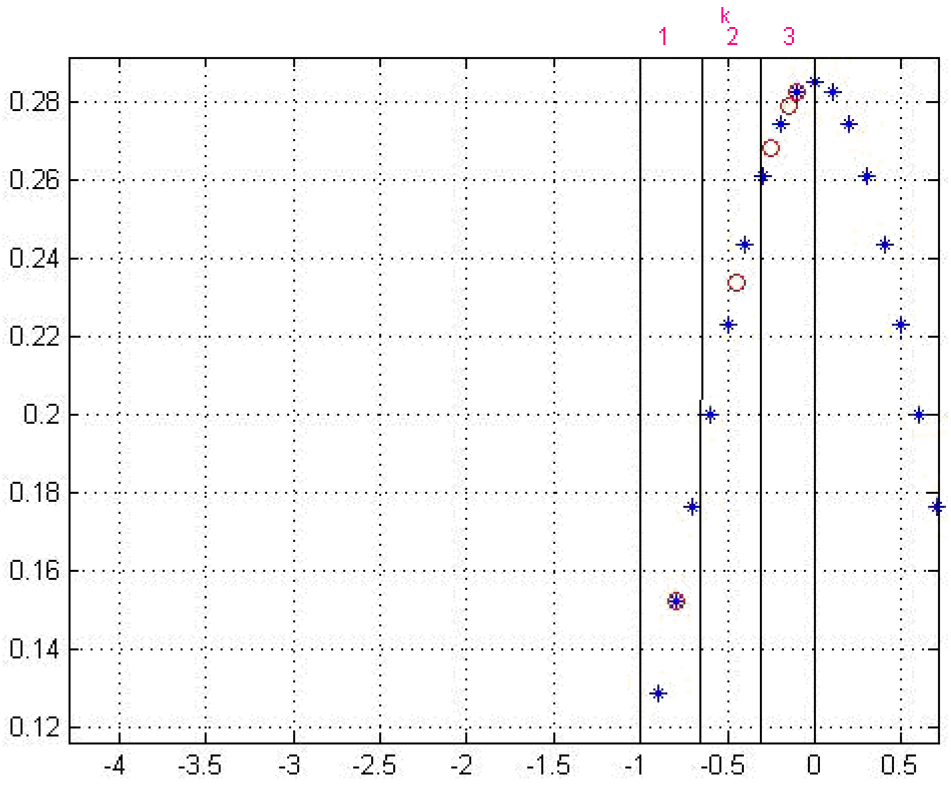


Figure 5.12 Demonstration of the samples size M and N.

Within [-1, 0], there are k=3 intervals. In the 1st and 2nd interval, M_1=1 and M_2=1 samples may be sufficient to represent the distribution because all the second derivatives in this interval are nearly equal to zero ,and M_3=3 sample are needed to re-construct the distribution. So the red samples are the most efficient and accurate representation samples. Comparatively, the blue ones are less efficient and accurate.

$$\begin{aligned} M &= \lambda N[f''(s_k)+f''(s_{k+1})+\ldots+f''(s_{k+N})]/N \\ &= \lambda \bullet [f''(s_k)+f''(s_{k+1})+\ldots+f''(s_{k+N})] \end{aligned}$$

(5.10)

In the above equation, λ is a constant, indicating that number of *M* is determined by the summation of the second derivatives of all samples in this interval.

As Figure 5.12, we uniformly take *k* samples in the optimal Gaussian distribution, and within these intervals, *M* samples are in the original discrete distribution, that is, $M_{i_k} \in \{M_1, M_2, \ldots, M_k\}$. Similarly, samples in the proposal distribution are also separated into k intervals, that is $N_{i_k} \in \{N_1, N_2, \ldots, N_k\}$

Because of the convergence of Ant Colony Optimization algorithm (Section 4.2), given a certain continuous optimal proposal distribution with *M* samples, the sample s^+_t are moved from the s_t after the ACO improvement according to the interpolation introduced in Section 4.2.

Therefore, we can compare two K-L Divergence before the ACO improvement, then Equation 5.9 becomes

$$D(p\,||\,q) \approx \frac{1}{M}\sum_{i_k=1}^{k}\sum_{n\in \text{interval k}}[\log \hat{p}(s_t(n)\,|\,s_{t-1}(i), z_t) - \log \tilde{q}(s_t(n)\,|\,s_{t-1}(i))] \quad (5.11)$$

and

$$D(p\,||\,q^+) \approx \frac{1}{M}\sum_{i_k=1}^{k}\sum_{n\in \text{interval k}}[\log \hat{p}(s_t(n)\,|\,s_{t-1}(i), z_t) - \log \tilde{q}^+(s_t(n)\,|\,s_{t-1}(i))]. \quad (5.12)$$

Let the sequence $\hat{M}_{i_k} \in \{\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_k\}$ denote the required particle number in each interval based on Equation 5.10. After sufficient iterations to achieve the optimal solution, if in an interval that the required particle number $\hat{M}_{i_k} \le N_{i_k}$, such as k = 1,2 in Figure 5.12, it is trivial that

$$D(p\,||\,q) = D(p\,||\,q^+). \quad (5.13)$$

If within the intervals that the required particle number $\hat{M}_{i_k} > N_{i_k}$, such as k = 3 in Figure 5.12, then

$$
\begin{aligned}
D(p\,||\,q) - D(p\,||\,q^{+}) &\approx \frac{1}{M}\sum_{n=1}^{M}[\log\frac{p(\hat{x}_k(n)\,|\,x_{k-1}(i), y_k)}{q(\tilde{x}_k(n)\,|\,x_{k-1}(i))} - \log\frac{p(\hat{x}_k(n)\,|\,x_{k-1}(i), y_k)}{q(\tilde{x}_k^{+}(n)\,|\,x_{k-1}(i))})] \\
&= \frac{1}{M}\sum_{n=1}^{M}[\log\frac{p(\hat{x}_k(n)\,|\,x_{k-1}(i), y_k)}{q(\tilde{x}_k(n)\,|\,x_{k-1}(i))} - \log\frac{p(\hat{x}_k(n)\,|\,x_{k-1}(i), y_k)}{p(\hat{x}_k(n)+\varepsilon\,|\,x_{k-1}(i), y_k)}] \\
&\to \frac{1}{M}\sum_{n=1}^{M}[\log\frac{p(\hat{x}_k(n)\,|\,x_{k-1}(i), y_k)}{q(\tilde{x}_k(n)\,|\,x_{k-1}(i))} - 0] \qquad (5.14.1) \\
&> 0
\end{aligned}
$$

(5.14)

The step (5.14.1) comes from the convergence of Ant Colony Optimization (Section 4.2). So within the intervals that the required particle number $\hat{M}_{i_k} > N_{i_k}$

$$D(p\,||\,q) > D(p\,||\,q^{+})$$

Given a number ε, with enough iteration, we can always achieve arbitrarily small K-L Divergence. Overall, when the K-L Divergence takes summation in all the intervals, we can conclude that $D(p\,||\,q) > D(p\,||\,q^{+})$.

The above theorem shows that the proposal distribution can ultimately achieve optimal one with Ant Colony Optimization when the number of iterations in a qualitative way. A quantitative study to the performance of PF$_{\mathrm{ACO}}$ will be shown in the following theorem and proof.

The main purpose of following proof is to bound the error introduced by the sample-based representation of PF. To derive this bound, we assume that the optimal distribution is given by a discrete, piecewise constant distribution [152] such as a discrete density or a multi-dimensional histogram. For such a representation, we can determine the number of samples so that the K-L Divergence between the maximum likelihood estimate based on the samples and the optimal distribution does not exceed a pre-defined probability. The optimization of PF can be achieved if its probability is larger than that of generic PF with the same amount of particle samples.

Theorem 5.2: With a given ε, if we need the efficient particle size *n* to approach the true distribution with an upper bound $\varepsilon\ \varphi$ on the K-L Divergence with probability $1-\delta$ in generic PF, then in PF$_{ACO,}$ the probability is always 1.

Proof :

Given the true distribution $p(s_t \mid s_{t-1}(i), z_t)$, after it is sampled to *k* bins, let the vector $\underline{S} = (S_1, S_2, ..., S_k)$ denote the number of samples drawn from each bin and $\underline{S}$ is distributed according to a multinomial distribution:

$$\underline{S} \sim Multinomial_k(n, \underline{p}) .$$

where $\underline{p} = p_1 \cdots p_k$ specifies the probability of each bin. The maximum likelihood estimate of $\underline{p}$ is given by $\underline{\hat{p}} = n^{-1}\underline{S}$. The K-L Divergence of these of two distributions $\underline{p}$ and $\underline{\hat{p}}$ is

$$D(\underline{\hat{p}} \,||\, \underline{p}) = \sum_x X_j \log(\frac{\hat{p}_j}{p_j} \tag{5.15}$$

Furthermore, when $\underline{p}$ is the true distribution, the likelihood ratio statistic λ_n for testing $\underline{p}$ is given as:

$$\begin{aligned} \log \lambda_n &= n \sum_x \hat{p}_j \log(\frac{\hat{p}_j}{p_j}) \\ &= n \sum_x \hat{p}_j \log \varepsilon_j \end{aligned} \tag{5.16}$$

After some arrangement by logarithm rules, we can derive $\lambda_n = \prod_x \varepsilon_j^{\hat{p}_j \cdot n}$. Please be noted that λ also can be arbitrarily small due to the assumption that ε is an enough small number.

From the Equations 5.15 and 5.16, it can be proved that the likelihood ratio statistic is *n* times the K-D Divergence between the proposal distribution and the true distribution:

$$\log \lambda_n = nD(\underline{\hat{p}} \,\|\, \underline{p})$$

(5.17)

We can see from literature [153] that the likelihood ratio converges to a chi-square distribution with *k-1* degree of freedom

$$2\log \lambda_n \rightarrow_d \chi^2_{k-1}$$

(5.18)

when $n \rightarrow \infty$.

Let $P_p(D(\underline{\hat{p}} \,\|\, \underline{p}) \le \varepsilon)$ denote the probability that the relative entropy between the true distribution and the proposal distribution is less than or equal to ε. The relationship between this probability and the number of samples is as follows:

$$\begin{aligned} P_p(D(\underline{\hat{p}} \,\|\, \underline{p}) \le \varepsilon) &= P_p(2nD(\underline{\hat{p}} \,\|\, \underline{p}) \le 2n\varepsilon) \\ &= P_p(2\log \lambda_n \le 2n\varepsilon) \\ &= P(\chi^2_{k-1} \le 2n\varepsilon) \end{aligned}$$

(5.19)

By replacing the second step of Equation 5.19 with the likelihood ratio statistic, and by using the convergence result stated in Equation 5.18, the quantiles δ of the chi-square distribution are given by

$$P(\chi^2_{k-1} \le \chi^2_{k-1,1-\delta}) = 1-\delta$$

(5.20)

If we choose *n* such that $2n\varepsilon$ is equal to $\chi^2_{k-1,1-\delta}$, we can combine the above two equations together and get

$$P_p(D(\underline{\hat{p}} \,\|\, \underline{p}) \le \varepsilon) = 1-\delta \qquad (5.21)$$

To explicitly summarize, if we choose the number of samples as

$$n = \frac{1}{2\varepsilon}\chi^2_{k-1,1-\delta} \tag{5.22}$$

Then we can guarantee that with the probability *1-δ*, the relative entropy between the proposal distribution and the true distribution is less than *ε* with a finite particle size *n*.

Note that Equation 5.19 can be modified as follows after the ACO improvement is successfully applied:

$$P_p(2\log\lambda_n \le 2n\varepsilon) = P_p(2\log\prod_x {\varepsilon_j}^{\hat{p}_j n} \le 2n\varepsilon) \tag{5.23}$$

When we maintain Equation 5.22, because $\prod_x {\varepsilon_j}^{\hat{p}_j n}$ is arbitrarily small, so equation $2\log\prod_x {\varepsilon_j}^{\hat{p}_j n} \le 2n\varepsilon$ is always true. Therefore the above probability is always one. It means that provided any *ε*, with the ACO improvement, we can always achieve the K-L Divergence with an upper bound of *ε* with a particle number *n*.

5.5 Conclusions

In this chapter, a number of experiments have been presented to examine the performance of PF_{ACO} comparing with other filtering methods. The results show that the novel method tracks more accurately than other methods and takes shorter computational time than other improved proposal distribution methods both in single variable estimation and in single robot localization problems. Moreover, the variance of particle weights decreases after the ACO improvements in experiments. In Section 5.3, two theorems about the ACO optimization have been introduced. First, it will converge to the optimal proposal distribution after infinite iterations. Second, with a pre-defined thresold, if the generic PF need an efficient particle size to approach the optimal distribution with a certain probability, then the PF_{ACO} has larger probability to achieve optimal with the same threshold and same particle number. In next chapter, the PF_{ACO} will be applied to the multi-robot SLAM problem.

Chapter 6 PF_{ACO} Application on Multi-robot SLAM

6.1 Introduction

In Chapter 4, the PF_{ACO} algorithm was introduced and it was applied to solve the single variable estimation problem as well as the single robot localization problem. However, as mentioned in Section 4.1, one of the advantages of our PF_{ACO} is that it is easy to be implemented in a multi-robot system and fully utilize the available computational power provided by the robots. Therefore for problems related to multi-robot, such as multi-robot SLAM, the time required to identify the solution can be reduced when comparing to a single robot system, with the assumption that an optimal communication system with no data loss and no time delay[154, 155] is available.

Because researchers had solved several key problems related to the implementation of distributed generic PF [156], in the following sections, we will focus on the realization of the novel PF_{ACO} using distributed computing algorithm. In addition, the coordinating strategy using distributed or ordinary PF algorithms of a multi-robot system adopted for solving the multi-robot SLAM problem will also be discussed.

6.2 Distributed Implementation of PF_{ACO}

The multi-robot system is simulated by the same simulator program introduced in Section 5.2. It is further modified and enhanced with the multi-robot function, to make it applicable to multi-robot SLAM simulation, experimental details will be introduced in Chapter 7. Furthermore, the Parallel Computing toolbox is applied in this program. This toolbox can simulate a parallel computing process and speedup the program using multi-core processors, in which each core is called a *worker* in the program. In order to make use of the Parallel Computing Toolbox, we have to establish "*the Matlab parallel language worker pool*" with parallel workers. For example, if we have a dual-core processor, the following command is necessary:

matlabpool open 2

which indicating that we will open two pools of MATLAB sessions for parallel computation.

We can then further modify the program to make it suitable for parallel computing. For example, the *parfor-loop*, a parallel version of a *for-loop*, enables the execution of statements in the loop using a maximum of M Matlab workers to simulate the parallel programming. The statement is automatically divided into M parts and distributed to workers (cores), so all we need to do is to simply modify some commands in the original program. In the following, we only convert the ordinary loops to parallel computing loops, because in the PF_{ACO}, loops are the main programming structure, also it is reasonable to distribute loops to multi-robot's processors in the real applications.

However, in Matlab's *parfor* command requirements, the computation included in those loops must be independent[157, 158], implying that the data being modified in one loop cannot be modified in another simultaneous loop. Although some techniques regarding to distributed computing with share parameters has been developed, we are able to comfortably split the data into any number of arbitrary parts and fully utilize the computational power with independent parameters.

Thus we analyzed every step of the PF_{ACO} to derive a suitable approach to tackle the problem. The PF_{ACO} includes three steps, which are:

1) Particle generations, also referred as prediction;

2) Particles movement, based on the ACO algorithm;

3) Re-sampling.

By analyzing the three steps involved in the PF_{ACO}, in the prediction step, it is trivial to modify the existing algorithm by using distributed techniques as particles are generated independently. For the re-sampling step, although it used to be a problem years ago, however, literatures in recent years [156] presented several methods to tackle this problem. Therefore, in the rest of this section, we will concentrate on the implementation of the ACO in a distributed approach.

Referring to Algorithm (4.1) , we can anticipate that the particles' state as well its weight can be calculated by different robots in a multi-robot environment in each iteration, except for some shared parameters, e.g. the parameter *d* representing the distance between particles. However, we could modify Algorithm (4.1), so that the parameters update stage is performed outside the loops, as shown in Figure 6.1.

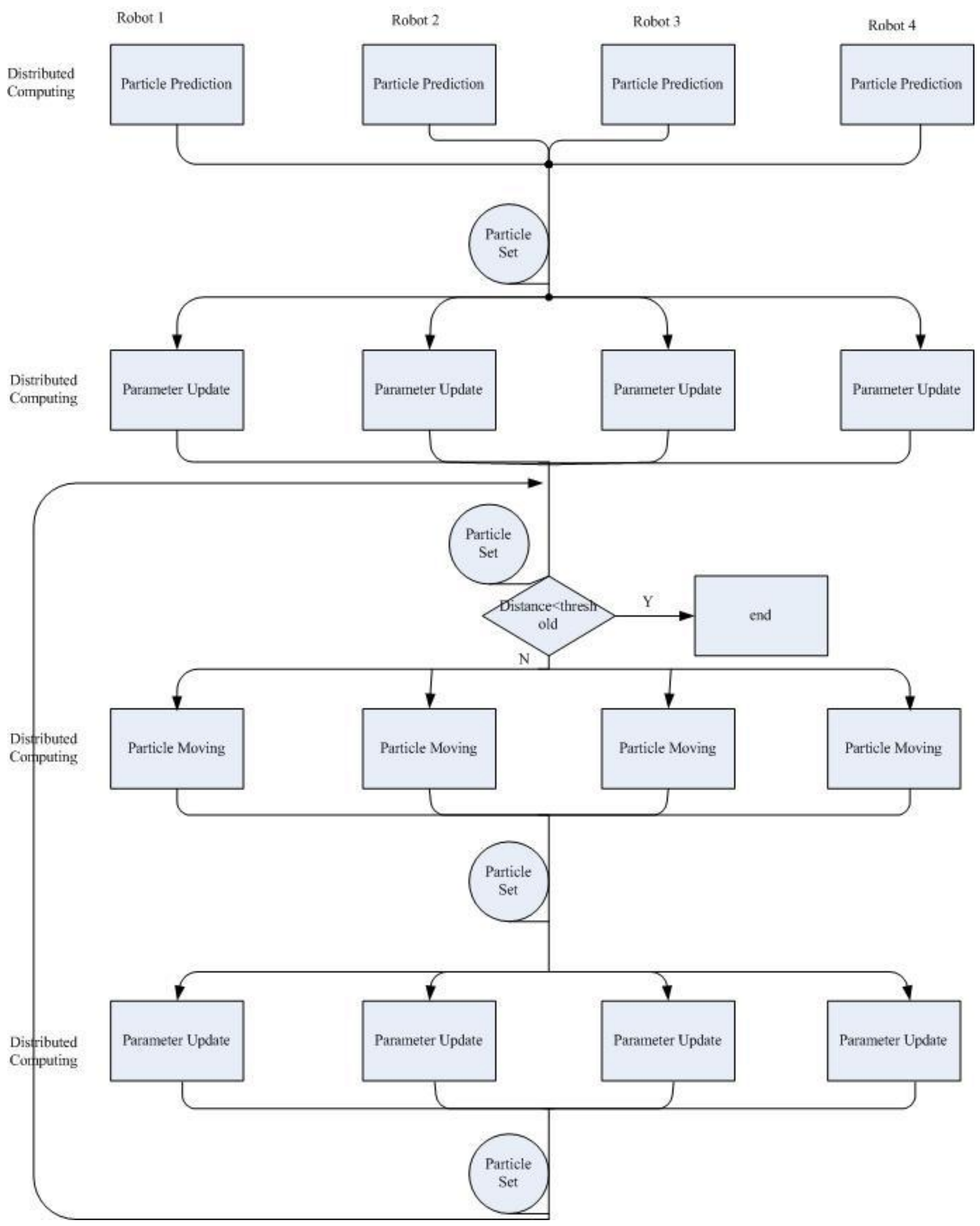


Figure 6.1 Flow-chart of ACO improved PF ran in four robots (also similar with different size of the troop). The whole process is divided into 4 sub-processes, each of which is implemented by the processors in robots.

As shown in Figure 6.1, the Algorithm 4.1 is divided into 4 parts, which can be computed in different processors. Loops in each part can be automatically pre-processed by the Matlab program, and then assigned to different processors. We assume that there is a control centre to communicate with robots and assist to pre-process some data. After all the tasks in loops are finished, data is transferred to the control centre and then, continues the program.

6.3 Coordination Strategy for Multi-robot SLAM

We have to clarify the scenario in which the multi-robot system employs the distributed or ordinary PF to solve the SLAM problem. Previous studies [159-161] on multi-robot SLAM often assumed that the relative poses of robots, or at least their initial poses, are known in advance, so under such an assumption, the single robot SLAM problem, which only includes the pose (the position and angle) of one robot, can be easily augmented into larger dimension estimation problems, and a multi-robot SLAM can be implemented by simply incorporating another set of robot poses into the filter. However, an important scenario of a multi-robot system is “encountering”, which means a robot detects another robot, implying that their relative pose can be obtained, so the assumption about initial poses are no longer necessary [161-163]. From the moment they encounter and know their relative pose, we can choose whether they both can start to jointly estimate the same map area.

In our experimental platform, we assumed that a robot can always distinguish its peers from other static objects (such as walls, obstacles, etc.), although this data association problem is still another major problem in SLAM[77, 164]. Furthermore, as stated in [162], for simplicity it is assumed that once a robot detects another robot within its laser sensing area, the exact relative poses of each other are also known.

In the following experiments, the robots also do not know the initial positions, or their relative positions. They simultaneously explore the area as depicted in Figure 6.2.

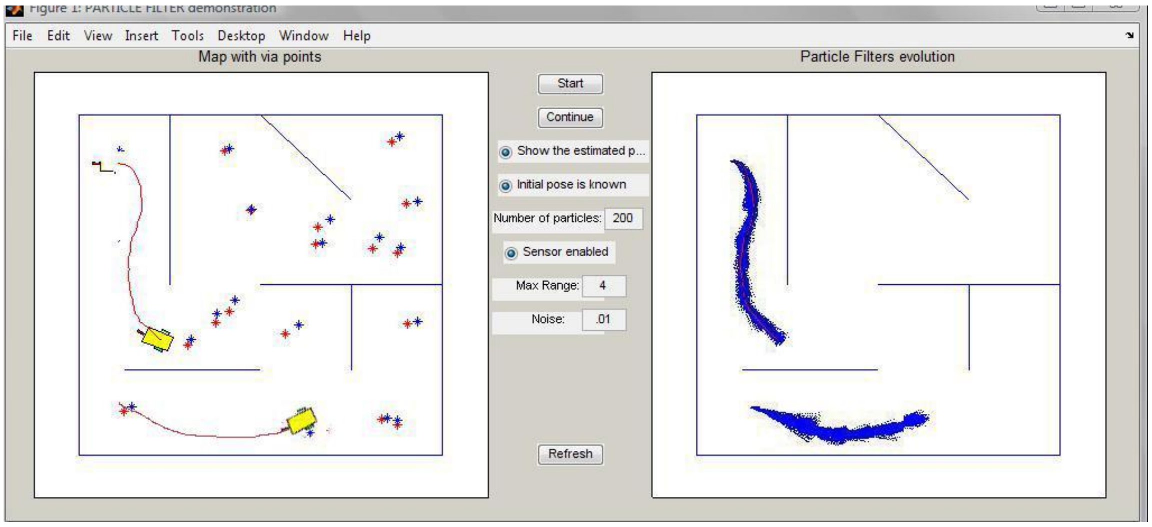


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Continue
Refresh

Show the estimated p...

Initial pose is known

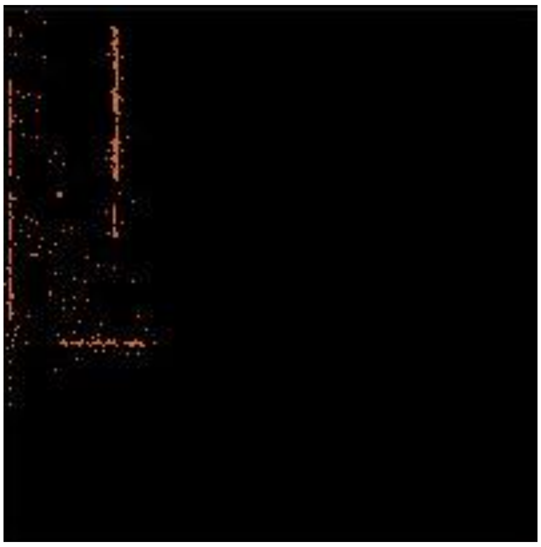
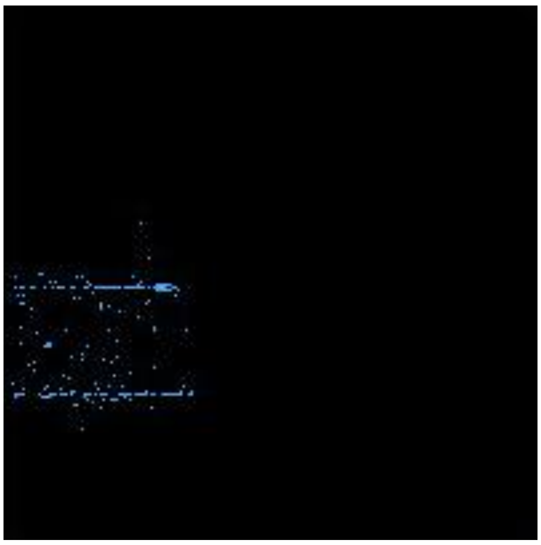
Number of particles: 200

Sensor enabled

Max Range: 4

Noise: .01

(a)

(b) (c)

Figure 6.2 Two Robots before Encounter (a)The navigation of two robots in the map; (b)The map estimated by robot A (the one at the top of the map); (c) The map estimated by robot B (the one at the bottom of the map).

The two map and their poses estimations are totally independent as the single robot SLAM are run in two robots.

These figures are derived from the simulation program which is further developed based on the localization version introduced in Chapter 4. There are mainly two differences between the single robot SLAM version and multi-robot SLAM version:

1) There are two sets of asterisks representing landmarks which help the robots in

navigation; the landmarks are only used for simulation and experimental use in order to make sure that robots in different algorithms run with the same trajectory rather than a random run which is the case in passive SLAM;

2) Two robots move independently within the map along the asterisks; the maps are jointly estimated by two robots.

At the beginning of the navigation, the SLAM posterior estimation problems are totally independent, that is, their measurement and estimation states are not co-related. Thus, before the encounter, the PF are being executed separately in each robot just like the single robot system and individual maps are generated by each robot peer as shown in Figure 6.2b and 6.2c.

As shown in Figure 6.3a, when two robots encounter, they will exchange the map data. The relative pose either detected using sensors or received from the other robot provides us a larger map that is obtained by linking together the two robots' separate maps (Figure 6.3). Consequently, the robots continue to navigate until one of the robots finds itself traversing the area which other robot had visited before (Figure 6.4). The distributed PF becomes active to solve the multi-robot SLAM. However, in this process, the requirements in the localization problem and the mapping problem of multi-robot SLAM are different so that it is being discussed in the following.

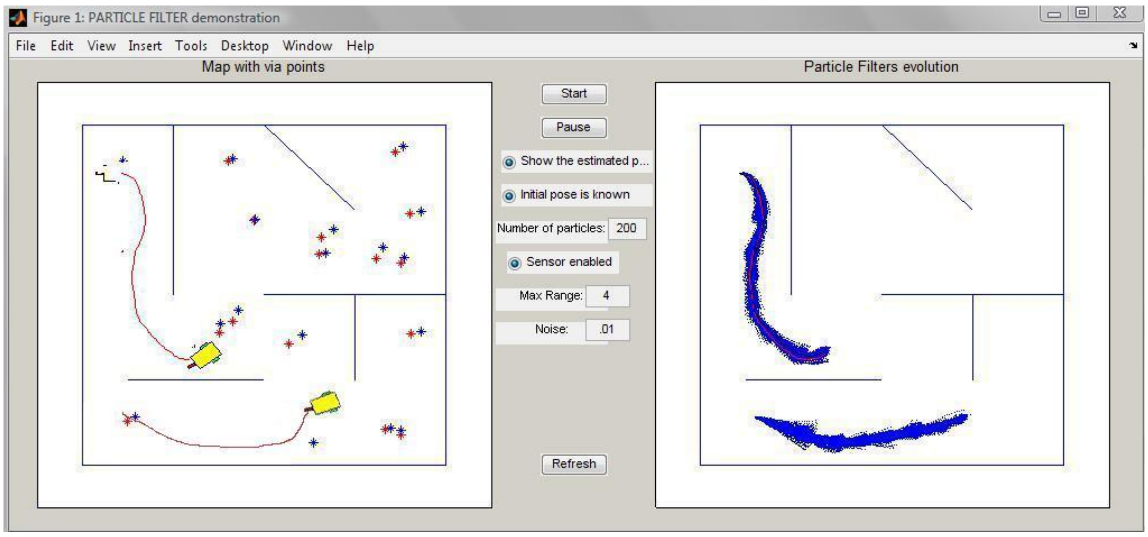
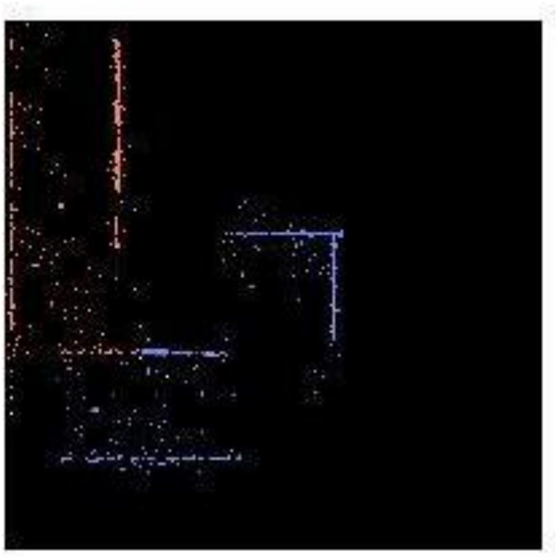


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

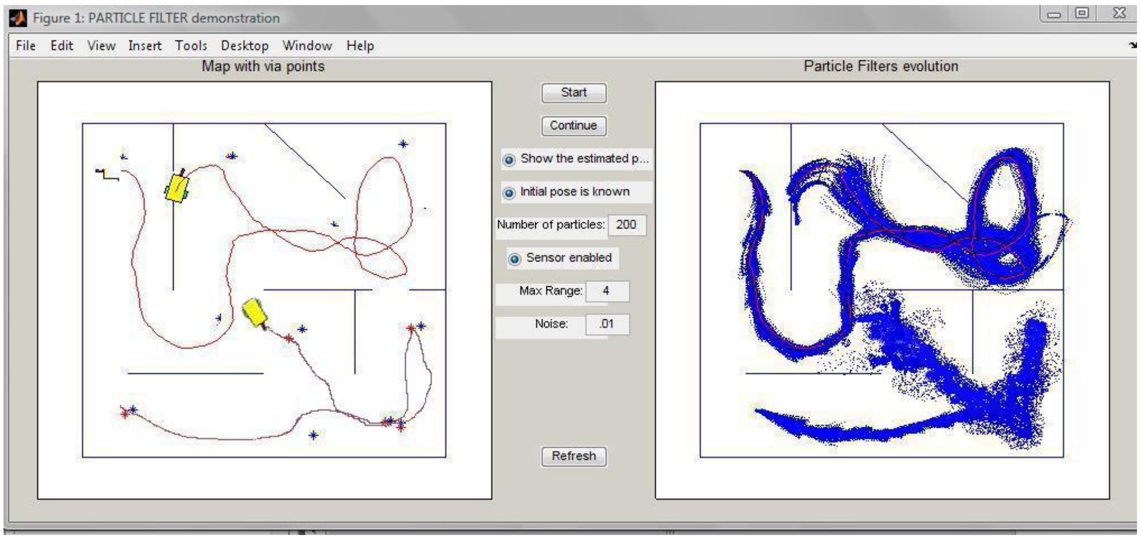
Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01

(a)

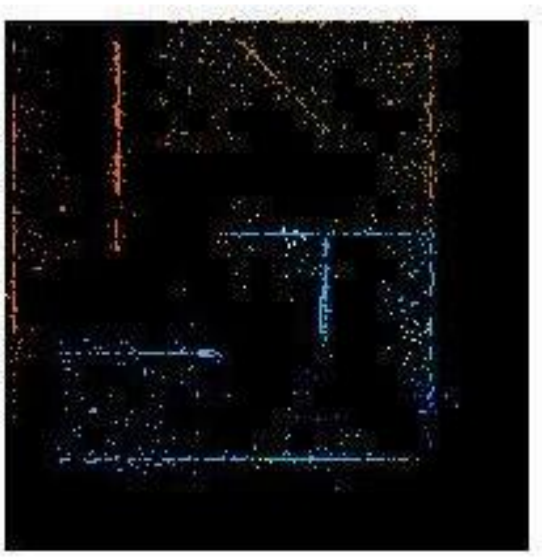


(b)

Figure 6.3 Two robots encounter (a) Two robots are encounter (b) Two estimated map is combined as a larger one as their relative poses are known.



(a)



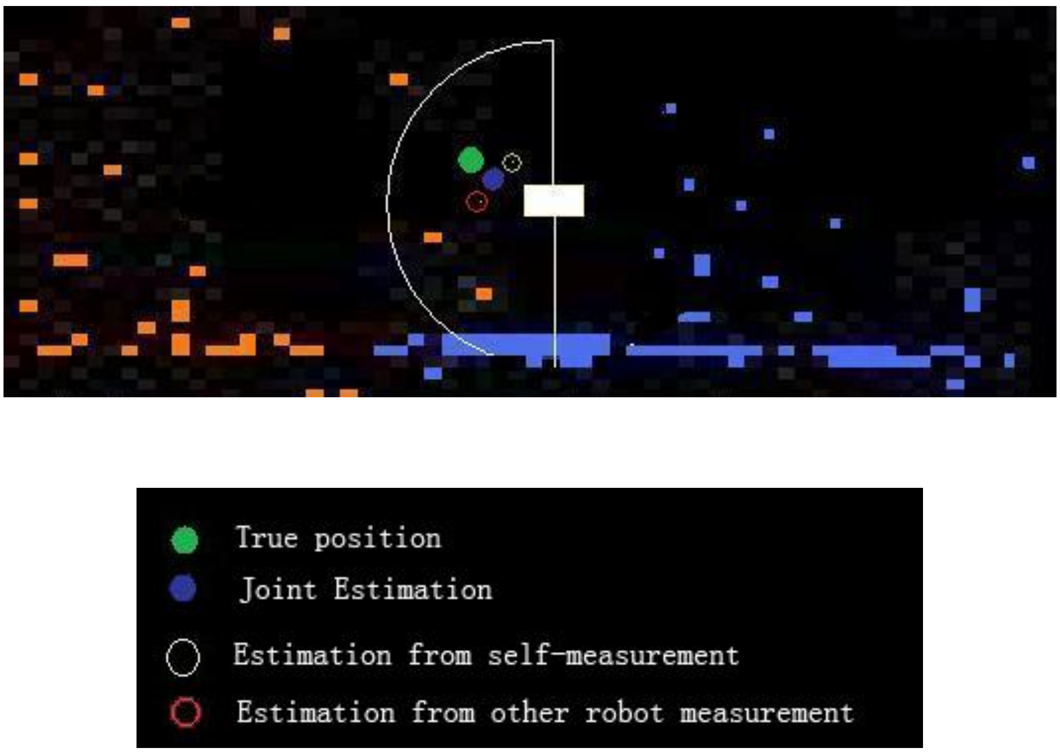
(b)

Figure 6.4 Start of Multi-robot SLAM (a) The system starts the multi-robot SLAM when a robot is traversing the area another robot visited before; (b) a map is almost generated at the moment. So we have two options: 1) coordinately estimate the rest of the map 2) improve the accuracy of the existing map

For the localization problem of a multi-robot system, because robots are always navigating in the environment, the poses of robots are always being updated throughout the whole process. Therefore, the PF running in each robot have to work individually and deal with different problems. Nevertheless, the relative pose after the encounter does offer us a reference for correction with the knowledge of both robots' poses as governed by Equation 6.1.

$$p(s_B) = \alpha p(s_B \mid z_B) + \beta p(s_B \mid s_A, \Delta_{AB}) \qquad (6.1)$$

The estimation of the pose of *robot B* is determined by its own measurement and estimation from another robot (say *robot A).* The importance coefficient α and β are defined based on their estimation accuracy, such as sensor error, where $0 < \alpha < 1$, $0 < \beta < 1$ and $\alpha + \beta = 1$. In Figure 6.5, it illustrates the result by applying Equation 6.1 to estimate the current pose of robot B. It is obvious that more accurate result can be obtained by considering information both from *robot A* and *robot B*.



True position
Joint Estimation
Estimation from self-measurement
Estimation from other robot measurement

Figure 6.5 A joint estimation demonstration of the robot position (the angle is omitted). Neither of the self-measurement and other robot measurement is exactly accurate to the true position. But the joint estimation reduces some of the estimation error.

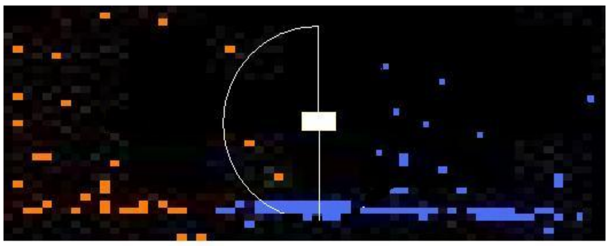
The mapping function of multi-robot SLAM can be jointly performed by two robots' as long as their relative poses are already known. Because the mapping result data is static due to the map grid invariance, if one map grid is already estimated by one robot, certainly another robot can estimate again to construct a more accurate map. Or it is not necessary to update the map again if we only wish to accomplish the mapping in the shortest time instead of emphasizing the accuracy of the map, so these robots can deal with the same map estimation problem, share the workloads and speed up the map estimation by distributed PF. Therefore, there exist two approaches in our joint mapping strategy, one is the efficient approach, and the other is the accuracy approach.

While operating based on the efficient approach, the mapping strategy is given below: when one robot (let us call it *robot A*) is traversing the area that *robot B* has visited before, and it realizes that its sensors are receiving data that *robot B* obtained before (it is easy to calculate since their relative poses are known) then *robot A* will stop constructing the map and commence to solve the PF which is running in *robot B*, indicating that the two robots are sharing the same problem. The PF_{ACO} distributed running in *robot A* and *robot B* is as shown in Figure 6.1.

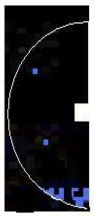
In the accuracy approach, the robots still run the PF separately after encounter. However, the major difference between single robot's mapping comparing to multi-robot is in the prediction step. When *robot A* is traversing the area where *robot B* has visited before, instead of a simple transition function, the map generated from *robot B* is also considered. The prediction function is given in Equation 6.2:

$$m_{t+1}^{A}(i) = \alpha_m f(m_t^{A}(i)) + \beta_m m^{B}(i) \tag{6.2}$$

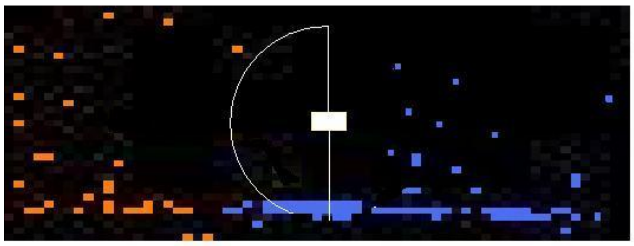
where α_m and β_m both are predefined constants, which indicate different accuracy levels of equipments, such as the range sensors, embodied in each robot. $f(m_t^{A}(i))$ denotes the map estimated by *robot A* after the transition model, where $m^{B}(i)$ denotes the map estimation by *robot B*. For the grid-occupancy map in our experiment, the conditions $0 < \alpha < 1$, $0 < \beta < 1$ and $\alpha_m + \beta_m = 1$ are defined. Figure 6.6 shows the results of the algorithm obtained by applying the accurate mapping option.



(a)



(b)



(c)

Figure 6.6 Demonstration of the accurate map estimation

(a) The joint map estimated by two robots, in which the orange part is estimated by robot B (not included in the figure), and the blue part is estimate by robot A (shown in the figure); (b) The map estimation from laser sensor data in robot A; (c) The joint estimation after process by Eqn. 6.2

6.4 Conclusions

In Section 6.1, the distributed version of our PF_{ACO} has been presented. In order to be computed distributive, the computing loop of our algorithm has to be independent. Therefore certain modifications of the algorithm are applied. In the second section of this chapter, we mainly focused on the application of multi-robot SLAM, especially how to deal with the mapping problem with multiple robots. After encounter, the relative poses of robots are known, which can be used as a correction reference to localization. Also, two different requirements in mapping can be adopted namely efficiency or accuracy, two different algorithms are introduced in order to satisfy the different requirements. In the next chapter, we will conduct several multi-robot SLAM experiments based on the PF_{ACO} and encounter strategy based on a Matlab simulation program.

Chapter 7 Multi-robot SLAM Simulation

7.1 Introduction

In Chapter 5, we studied experimental results of PF_{ACO} and other filtering methods when applied to a single variable estimation problem, single robot localization and SLAM problem. In Chapter 6, we also proposed the distributed version of PF_{ACO} for the SLAM problem and its application in a multi-robot system. In this chapter, we will further examine the following issues:

1) the PF_{ACO} performance comparing to other PF methods in solving the multi-robot SLAM problem;
2) different outcomes due to the accurate and efficient approaches in the mapping.

The simulation program of multi-robot SLAM has some differences comparing with the single robot version and the localization version. The interface of this program is shown in Figure 7.1.

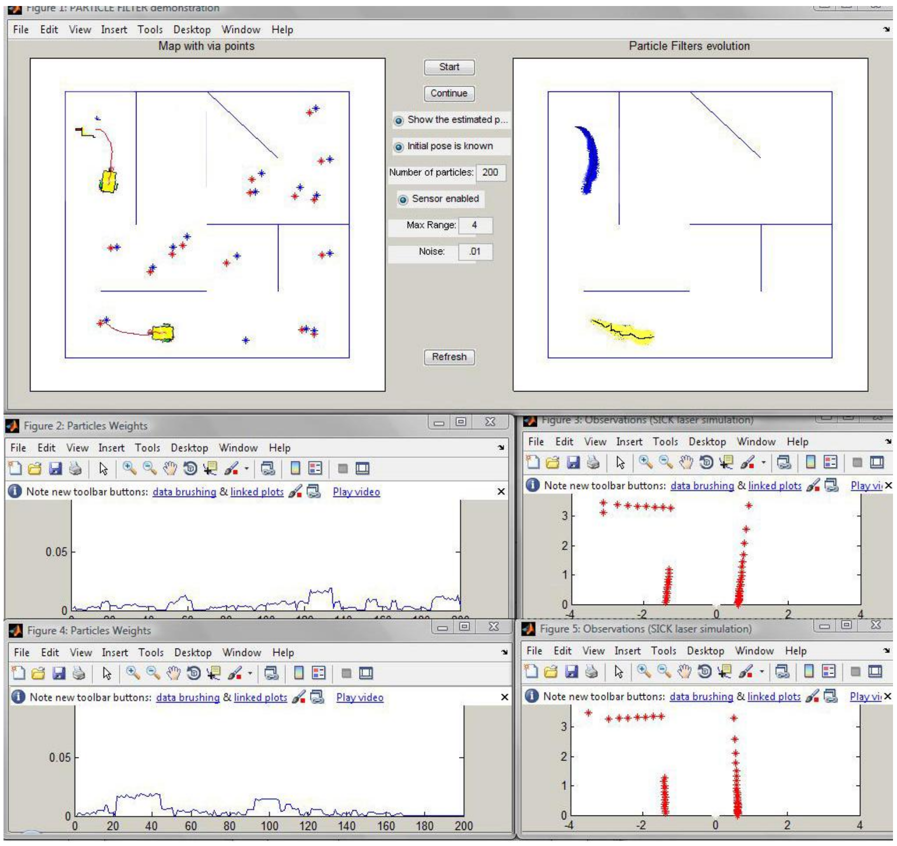


Figure 1: PARTICLE FILTER demonstration
Map with via points
Particle Filters evolution
Start
Continue
Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01
Refresh
Figure 2: Particles Weights
Figure 3: Observations (SICK laser simulation)
Figure 4: Particles Weights
Figure 5: Observations (SICK laser simulation)
Note new toolbar buttons: data brushing & linked plots
Play video

(a)

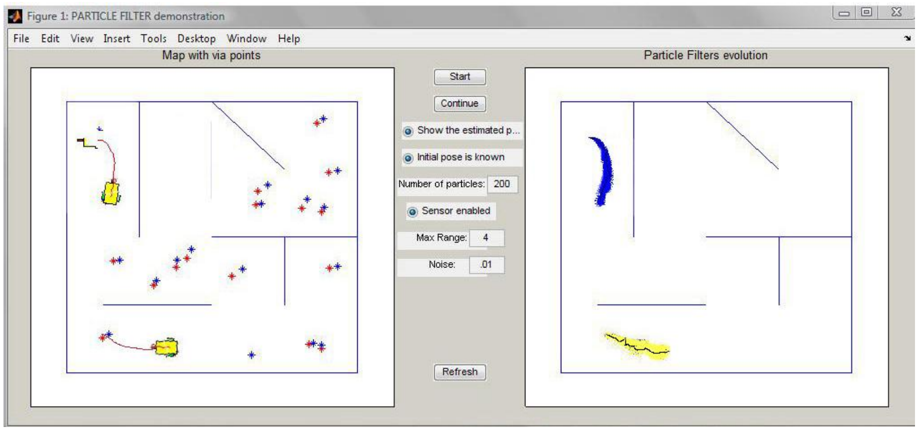


Figure 1: PARTICLE FILTER demonstration
Map with via points
Particle Filters evolution
Start
Continue
Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01
Refresh

(b)

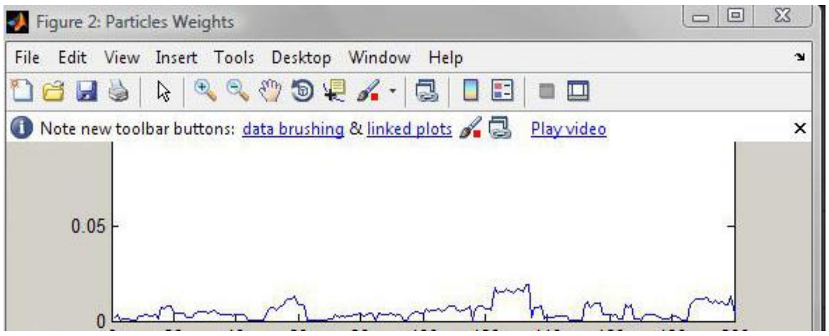
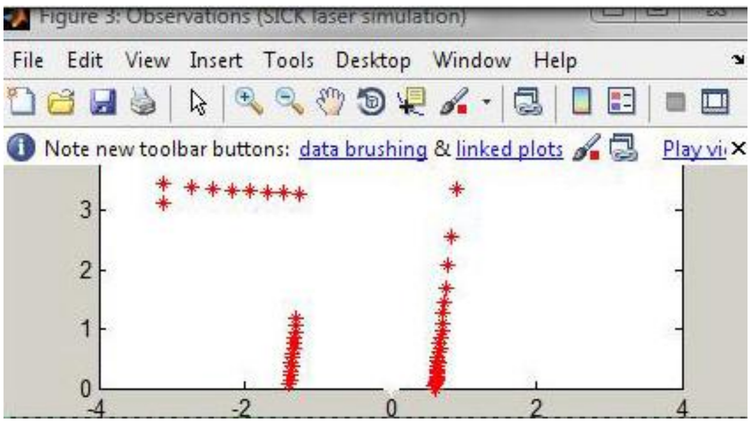
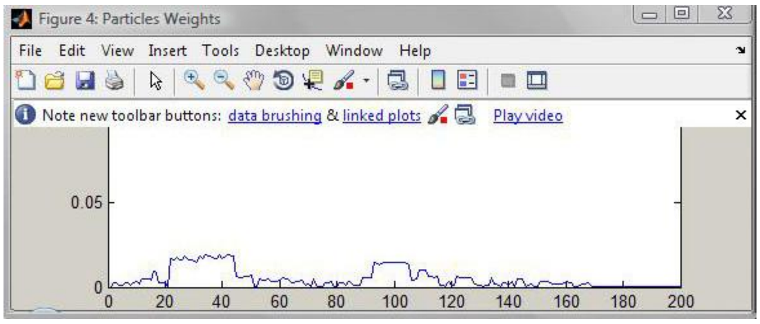


Figure 2: Particles Weights
Note new toolbar buttons: data brushing & linked plots
Play video
0.05
0

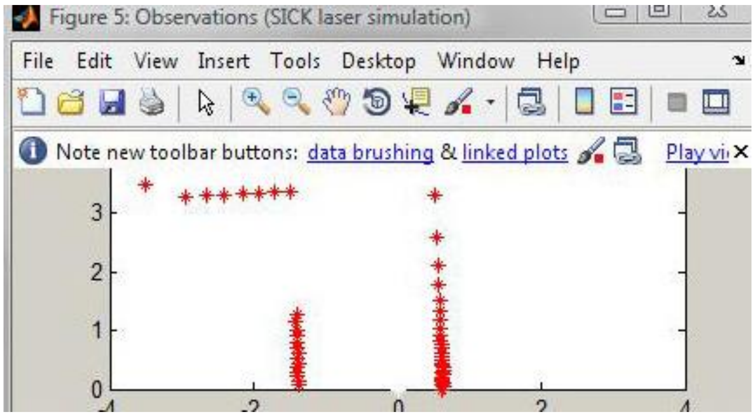
(c)



(d)



(e)



(f)

Figure 7.1 Multi-robot SLAM simulation program in Matlab

(a) A complete view of the interface; (b) The navigation of two robots, in which the left map shows the true map and the trajectory of the robots' navigation (red curve) along the asterisks set in advance, and the right map is the evolution demonstrations of particles representing the robot's pose while the robot is traversing in the area. (c) the graphs depict the pose particles' weight distribution of robot A. (d) the observation from the SICK laser sensor equipped in robot A.(e) the graphs depict the pose particles' weight distribution of robot B. (f) the observation from the SICK laser sensor equipped in robot B.

In Figure 7.1, the interface shows the localization information, including the particles weights representing the robot poses (Figure 7.1c and 7.1e), and the sensing results by the SICK laser sensors. Besides the information provided in the interface, the mapping information of two robots is simultaneously stored with the binary matrix as shown in Figure 7.2a, in which the value of one represents the obstacle sensed from the laser sensor, and value of zero represents the empty space. Therefore, the mapping information, as shown in Figure 7.2b, can be re-produced by a simple drawing command in Matlab.

7.2Multi-robot SLAM

In this section, a comparison of four PF algorithms including the generic PF, the Extended Kalman Particle Filter, the Unscented Particle Filter and PF_{ACO}, in solving the multi-robot localization problem is provided. Four SLAM methods were tested and all tests were conducted with the same initial particle set, so that the random factors can be minimized during the comparison. Since some of the algorithms, i.e. the generic PF and the extended Kalman Particle Filter, are difficult to be modified to fully utilize distributed computing power, in order to compare their estimation results and execution time, our PF_{ACO} on multi-robot SLAM is set to run in the accuracy mapping approach, that is, in each PF estimation, the maps are estimated twice by two robots.

In the following experiments, four PF methods are employed to solve the same multi-robot SLAM problem. Similar to previous experiments, when the two robots are navigating along the asterisks, the laser sensor will provide information for the PF SLAM estimators.

Figures 7.2 and 7.3 show the estimation of map and poses from four PF algorithms.

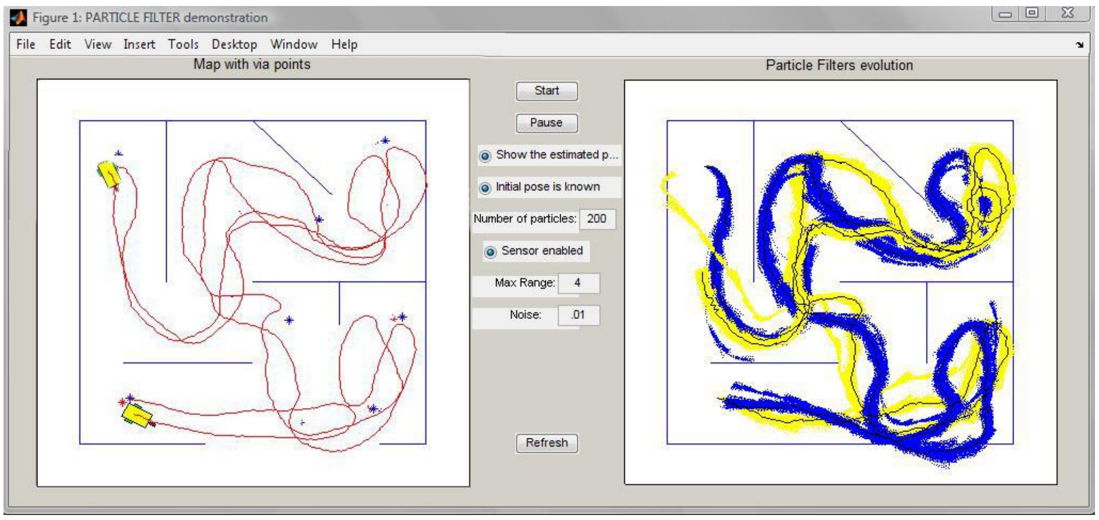


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

Show the estimated p...

Initial pose is known

Number of particles: 200

Sensor enabled

Max Range: 4

Noise: .01

(a)

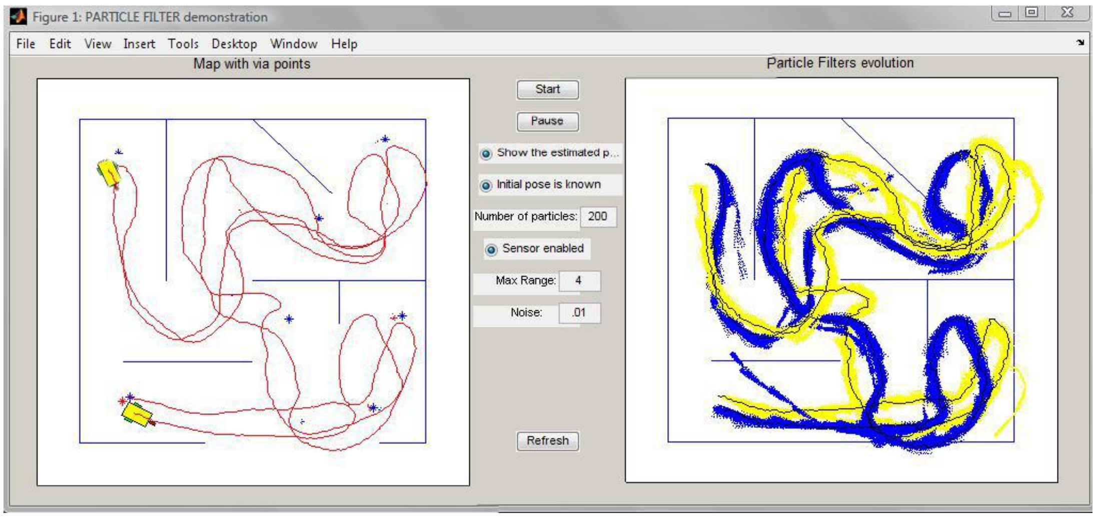


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

Show the estimated p...

Initial pose is known

Number of particles: 200

Sensor enabled

Max Range: 4

Noise: .01

(b)

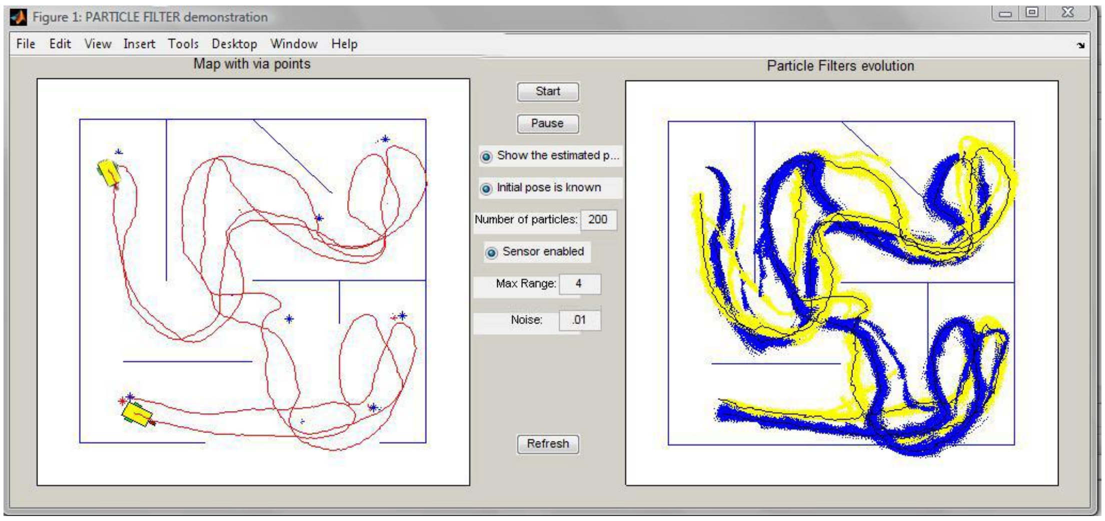


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

Show the estimated p...

Initial pose is known

Number of particles: 200

Sensor enabled

Max Range: 4

Noise: .01

(c)

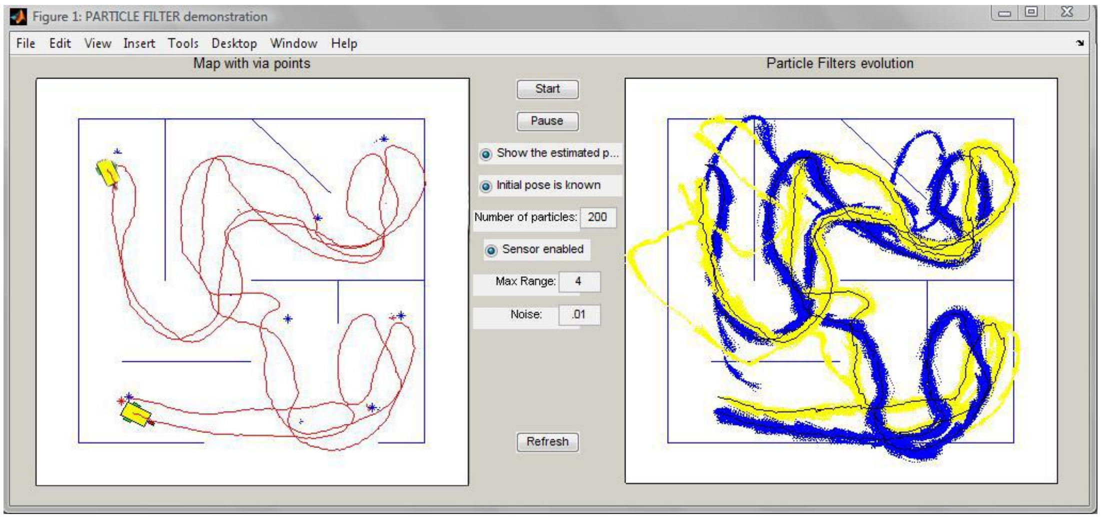


Figure 1: PARTICLE FILTER demonstration
File Edit View Insert Tools Desktop Window Help
Map with via points
Particle Filters evolution
Start
Pause
Refresh

Show the estimated p...
Initial pose is known
Number of particles: 200
Sensor enabled
Max Range: 4
Noise: .01

(d)

Figure 7.2 The poses estimation from four PF methods. Given the same motion trajectory and observation, different results are obtained by PF methods:

(a) *In the estimation from the generic PF, some deviation of the trajectory can be found. Also there exists some error accumulation which was in the right top corner. (b) Estimation from Extended Kalman PF; (c) Estimation from the Unscented PF, (d) Estimation from ACO improved PF*

In Figure 7.2, though all four methods have some deviations, such as particles were located outside the map area, in the particle evolutions, the estimated paths were generally fit the robot trajectory. From Figure 7.2a, the generic PF produced some path estimation errors in the upper right corner of the trajectory. The other three PF methods estimated the trajectory satisfactorily.

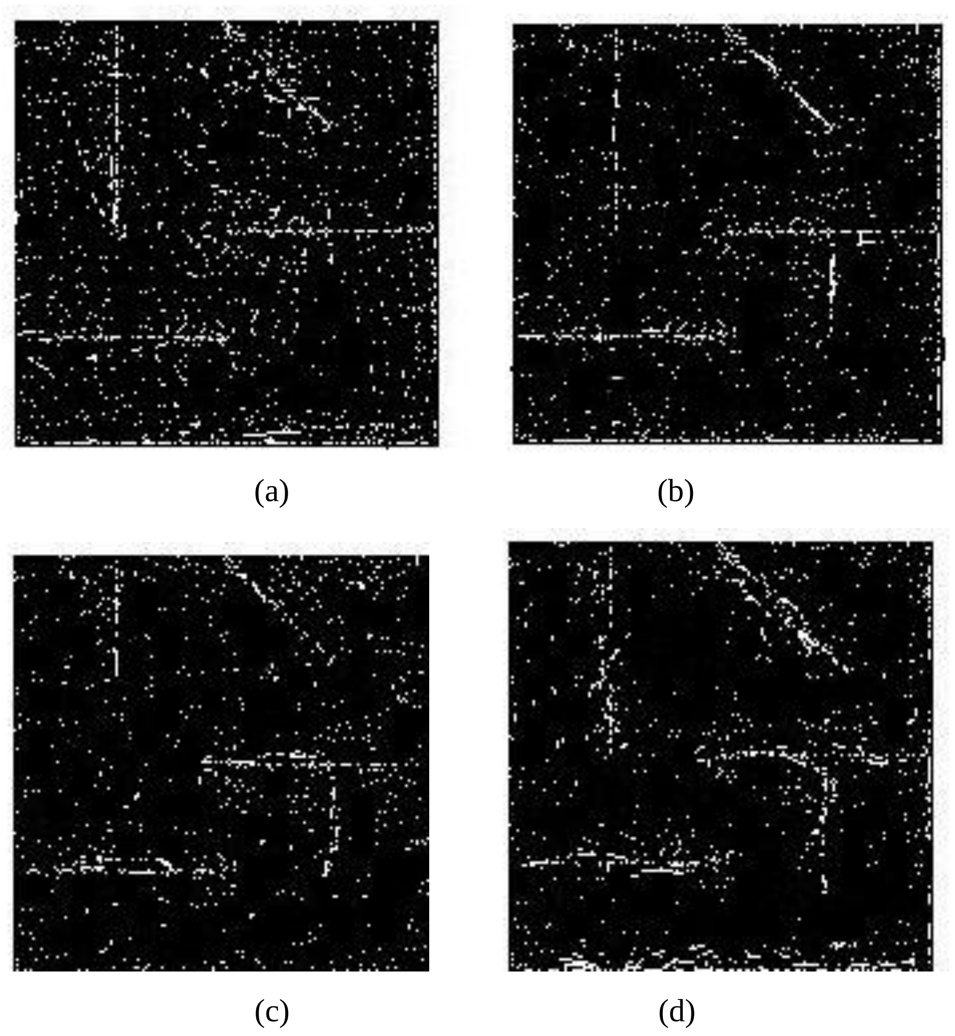


Figure 7.3 Maps estimated by PF methods

(a) Estimated map from generic PF; (b) Estimated map from Extended Kalman PF; (c) Estimated map from Unscented PF; (d) Estimated map from ACO improved PF. We can see that the ACO improved PF have less deviation in map estimation than other three methods.

Referring to Figure 7.3, comparison between different PF methods for mapping is presented. By observation, the map created by generic PF, as shown in Figure 7.3a, has the largest errors in estimation and a numerical measurement will be presented in the following section. With the same particle size, the Extended Kalman PF and Unscented PF represented by Figure 7.3b and 7.3c have less deviation in the mapping. Comparatively, in Figure 7.3d, the PF_{ACO} estimated obstacles having least deviation among the four methods in Figure 7.3 a~7.3c. Although these four estimated map is not quite satisfactory due to the large noise added, further enhancement techniques can be used afterwards to produce better maps.

The mapping results are now analyzed quantitatively. After the alignment between the estimated map and the original map, error estimation is conducted. That is the mapping error (*E_m*) and the pose error (*E_{pos}*) is represented as follows:

$$E_m = \sum_{i,j=1}^{L} \left| \tilde{m}_{ij} - m_{ij} \right| \tag{7.1}$$

$$E_{pos} = \sum_t \left| \tilde{X}_R - X_R \right| + \left| \tilde{Y}_R - Y_R \right| \tag{7.2}$$

$$E_{ang} = \sum_t \left| \tilde{\omega} - \omega \right| \tag{7.3}$$

where L is the number of grids in one edge. In Equation 7.1, $\tilde{m}_{ij}$ is the estimated value of grid *ij*, while m_{ij} is the true value. Similarly, $\tilde{X}_R$ and $\tilde{Y}_R$ are the estimated value of the position in axis X and Y; X_R and Y_R are the true value of position in Equation 7.2; $\tilde{\omega}$ is the estimated angle and ω is the true angle in Equation 7.3.

	Generic PF	Error percentage	Extended Kalman PF	Error (%)
E_m	9703	-	6281	64.73%
E_{pos}	120.6237	-	100.2955	83.15%
E_{ang}	31.3138	-	40.9202	130.68%
	Unscented PF	Error percentage	PF+ACO	Error (%)
E_m	6034	62.19%	4723	48.68%
E_{pos}	88.9311	73.73%	80.2138	66.50%
E_{ang}	30.5793	97.65%	26.3901	84.27%

Table 7.1 Multi-robot SLAM error and error percentage (comparing with generic PF)

To study the computational time of different PF methods, we compare the cumulative computational time of determining values of three states, i.e. the map, the position and the angle, and the results are listed in Table 7.2.

	Generic PF	Extended Kalman PF	Unscented PF	PF+ACO
Position	119.41	194.53	196.37	141.52
Angle	57.93	87.99	89.28	97.15
Map	620.78	740.79	831.92	799.25
Total	798.12	1023.31	1117.57	1037.92

Table 7.2 The cumulative running time (in seconds) of three states

The generic PF takes the shortest time to complete the computation, while the other three methods are in the same range. But the PF_{ACO} requires shorter time to accomplish the map and position states than the Unscented PF because the PF_{ACO} has advantages solving high dimension problem.

Table 7.2 shows that the execution time of PF_{ACO} is shorter than the Unscented PF, and is in the same range of that obtained for extend Kalman PF. The execution time in Matlab simulation seems quite long compared to most of the mobile robot application requirements. However, it mostly results from that Matlab is not an optimized programming language, and it includes numerous parts that are not necessary in robot applications, for example, the GUI to demonstrate the results. Therefore, the execution of the algorithms can be shortened when applied in real cases.

7.3 Mapping Approaches

After a comparison of different PF methods, we will focus on the different maps resulted from two mapping approaches as discussed in Section 6.3 namely the efficient and the accurate approaches. Figure 7.4 depicts the map created from the same experimental data but under these different mapping approaches.

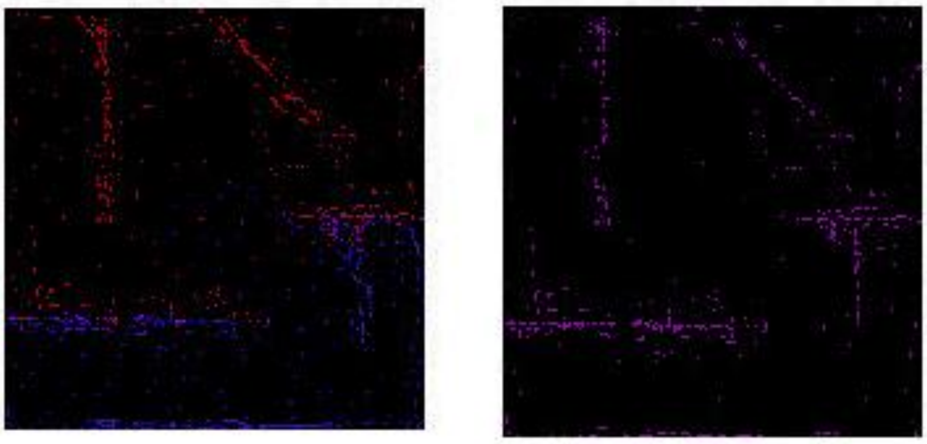

(a) (b)

Figure 7.4 The maps estimated under two different approaches

(a) The map under efficient approach, in which two maps (the blue and red map) are simply linked together, accept some purple grids representing the obstacle estimated by both robots; (b) the map under accurate approach as stated in Equation. 6.2 (both α and β equal to 0.5), which has less deviation with the original map. But some obstacles (wall) are not continuous.

Table 7.3 shows the error which is defined by Equation 7.1 and accumulated execution time of the map estimation.

	Accurate	Efficient
E_m	1037	5006
Execution Time (Sec)	820.13	529.27

Table 7.3 Multi-robot SLAM Results from the two mapping approaches

From Table 7.3, we can anticipate that the accurate approach is more suitable in robot navigation because it allows the robots to acquire its surrounding more precisely and determine its action autonomously. However, the efficient approach is also useful if we do not need such an accurate map, especially if we can process the estimated map (Figure 7.3a) with some image processing filtering methods [165], the "noise" in this map can be possibly eliminated. Furthermore, the processing time of the efficient approach is much shorter than the accurate one.

7.4 Conclusions

In this chapter, two experiments, concentrating on the comparison of different PF applications in multi-robot SLAM as well as comparing the two distinct mapping approaches, have been presented. Based on results derived from the first experiment, we can conclude that PF_{ACO} leads to least deviation in mapping and localization among the four methods, and its total processing time is in the same level of the other two proposal distribution methods (EKF, UKF proposal distributions). The second comparison shows the different results characterized in the mapping error and the cumulative running time from two approaches. These two approaches can be utilized depending on actual application requirements.

Chapter 8 Conclusions and Future Works

8.1 Conclusions

The SLAM (Simultaneous Localization and Mapping) problem is a fundamental problem in the development of autonomous mobile robot system and therefore, it is attracting numerous researchers to work on this field. Between two main branches of SLAM solution namely the Kalman Filters and Particle Filters, in this thesis, we mainly focus on the Particle Filters approach for solving the multi-robot SLAM problem due to its advantage in estimating non-Gaussian and non-linear models both of which are relevant to the SLAM problem. However, multi-robot SLAM is a high-dimensional estimation problem and when applying Particle Filtering to obtain a solution, the most straightforward method is to enlarge the particle size so that the particle impoverishment problem can be minimized, but on the other hand, it also increases the computation burden.

In order to tackle this problem, some researchers focused on deriving methods to establish the proposal distribution to get closer to the optimal distribution. Rather than applying the traditional modeling methods to approach the optimal solution, such as the extended Kalman Particle Filter and Unscented Particle Filter, we proposed a biologically inspired method, the Ant Colony Optimization, to direct the proposal distribution to approximate the optimal solution.

We first conducted an analysis of the optimization of PF proposal distribution. Based on this framework, we rewrote it as a combinatorial optimization problem. So it is possible to be solved by Ant Colony Optimization (ACO) as a novel algorithm namely the PF_{ACO}. The mathematical proof showed that the ACO can drive the proposal distribution to approach the optimal solution and improve the efficient particle size. Therefore, fewer particles are needed to re-construct the optimal solution.

Because the theoretical theorems are ideal cases with infinite iterations, a threshold in iteration is set in experiments to better balance the optimized result and efficiency. Furthermore, from our experimental results, it substantiated that the estimation results produced by ACO improved PF are more accurate than generic Particle Filters and other Particle Filters with improved proposal distribution such as extended Kalman Particle Filter and Unscented Particle Filter.

Consequently, PF_{ACO} was applied in solving the multi-robot SLAM problem. We implemented the algorithm using a simulation program written in Matlab. Also, to efficiently utilize the algorithm for a multi-robot system and make full use of the computational power in each robot, we designed a modified version of our algorithm for distributed computing.

Being different from traditional multi-robot SLAM solution, in our experiment, we do not need to obtain the prior information about the relative poses in advance. Our solution is this: when the navigation starts, SLAM is accomplished by every robot independently as a single SLAM problem before robots encounter. Once the encounter occurs, the relative poses information is obtained, and then the encounter strategies are applied. When the robots encounter, the map stored in each robot obtained from the single SLAM is linked and combined into a larger map which is shared by each robot. Then a mutual correction of the poses information can be conducted to establish more accurate poses estimation. Finally, there are two possible mapping approaches: the accurate approach and the efficient approach.

The result of mapping with efficient approach as if the whole estimated map is linked by partial maps generated by individual robots. Because the robot determines that it is not necessary to update the map again if we only wish to accomplish the mapping in the shortest time instead of the accuracy of a map grid. On the other hand, with the accurate approach, when a map-grid is already estimated by one robot, it is estimated again by another robot and considered jointly to construct a more accurate map.

For this project, the following conclusions are made.

1) As a metaheuristic method to combinatorial optimization problems, ACO can be applied to optimize the proposal distribution. By optimizing the particle distribution, the experimental results show that the PF_{ACO} tracks more accurately than other methods and takes shorter computational time than other improved proposal distribution methods when using the same number of particles.

2) To fully utilize the computational power of robots, a key issue in distributed computing is to partition processing tasks included in loops into different processors. However, the computation included in those loops must be independent, implying that the data being modified in one loop is not modified in another. The PF_{ACO} algorithm was modified in order to implement the algorithm into a multi-robot system.

3) Two mapping approaches lead to different results. The efficient approach roughly estimates the environment map, while the accurate approach estimates the map with two robots. They can be selected according to different application requirements.

8.2 Further Development

The experiments carried out in this thesis are based on the Matlab simulation program, and they are based on the assumptions that optimal communication and accurate sensors are available. Therefore, several technical problems should be resolved in applications using real robots.

1) The relative pose assumption is not held in application, so in practice the relative pose is also needed to be estimated, and an extra dimension in the filters will be included.

2) In our experiments, we assumed that there is a central control between robots to

assign the computational tasks and enabling real-time and optimal communication within the multi-robot system. In practice, noises from various sources or communication error (such as communication lost between robots) may be introduced. So a fault-tolerant and flexible communication technique [166] together with an error checking method is necessary.

3) The Matlab experiments often take longer time than the practical experiments because they are not optimized as demonstration modules (such as the GUI) are included but in practical applications such modules may not be necessary. Therefore, the Matlab experiments are only for comparison between different filter algorithms and optimizing the program and building it in a real experiment is a better way to evaluate the execution time in practice.

4) Finally, the map in Matlab simulation is built on simple planar boundaries. If it is modified into a more complicated map presentation, although the complexity in grid occupancy map based PF remains the same, not depending on the kind of obstacles, a suitable sensor, e.g. vision sensor, is needed to detect the obstacles with tiny or different shape. Consequently, more sensor model may be needed in the Matlab simulation program.

Inspired from this thesis, some future research directions of the multi-robot SLAM can be made.

1) From the idea of tuning the particle distribution, better solution by various optimization methods can also be developed. Since we already established the combinatorial problem based on the relationship between Monte Carlo samples and optimal proposal distribution, other optimization methods and meta-heuristic, such as Evolutionary Computation, Simulated Anneal, can also be applied to construct the better proposal distribution function.

However, the main problem of these methods is their computational complexity. In some cases, the real-time requirement of the estimation problems, such as multi-robot SLAM, is critical. Thus a proper decision to trigger and terminate the metaheuristic methods is also necessary to be considered by some Machine Learning methods.

Furthermore, other model based methods, e.g. extended Kalman Particle Filter, Unscented Particle Filter, can be combined with the above metaheuristic methods, so that the accuracy and estimation time can be improved.

2) The so-called efficient mapping approaches may not be the most efficient due to the fact that we cannot actively control our robots. This subject is called Active SLAM, which is the combination of trajectory planning and SLAM.

In our future investigation, the multi-robot Active SLAM will be conducted in order to explore an unknown environment and build its map more efficiently. The multiple robots coordinate control is not trivial at all. In the current stage, the Maximum Entropy method, as well as Reinforcement Learning, is considered to select the most optimal navigation path.

References

[1] E. Sahin, "Swarm Robotics: From Sources of Inspiration to Domains of Application," *Lecture Notes in Computer Science - Swarm Robotics*. Berlin / Heidelberg : Springer, 2005, pp. 10-20.

[2] M.F.Ercan, L.Partawijaya, and Y.F.Fung, "Collective Search and Exploration with a Robot Swarm," *IEEE Region 10 Conference TENCON,* pp. 1-4, 2006.

[3] A.A.G.Requicha, "Nanorobots, NEMS, and nanoassembly," *Proceedings of the IEEE,* vol. 19, no. 11, pp. 1922-1933, 2003.

[4] M.Yim, D.G.Duff, and K.D.Roufas, "PolyBot: a modular reconfigurable robot." *Proceeding of IEEE International Conference of Robotics and Automation* , pp. 514-520, 2000.

[5] A.Davids, "Urban Search and Rescue Robots: from Tragedy to Technology," *IEEE Intelligent Systems and Their Applications,* vol. 17, no. 2, pp.81-83, 2002.

[6] A.Birk and S.Carpin, "Rescue Robotics - a Crucial Milestone on the Road to Autonomous Systems," *Advanced Robotics,* vol. 20, no. 5, pp.595-605, 2006.

[7] W.Burgard, M.Moors, D.Fox et al., "Collaborative Multi-Robot Exploration," *IEEE International Conference on Robotics and Automation,* vol. 1, pp. 476-481, 2000.

[8] D.Fox, W.Burgard, H.Kruppa et al., "A Probabilistic Approach to Collaborative Multi-robot Localization," *Autonomous Robots,* vol. 8, no. 3, pp.325-344, 2000.

[9] L.Hugues, "Collective Grounded Representations for Robots," *Proceedings of International Conference on Distributed Autonomous Robotics Systems* , pp. 79-88, 2000.

[10] L.Iocchi, D.Nardi, M.Piaggio et al., "Distributed coordination in heterogeneous multi-robot systems," *Autonomous Robots,* vol. 15, no. 2, pp. 155-168, 2004.

[11] L.Whitcomb, D.R.Yoerger, H.Singh et al., "Advances in underwater robot vehicles for deep ocean exploration - Navigation, control, and survey operations," *Proceedings of the 9th International Symposium* , pp. 439-448, 2000.

[12] J.J.Leonard, A.A.Bennett, C.M.Chriatopher, et al., "Autonomous Underwater Vehicle Navigation," *Autonomous Underwater Vehicle Navigation,* MIT Marine Robotics Laboratory Technical Memorandum 98-1, 1998.

[13] R.Russell, "Robotic location of underground chemical sources," *Robotica,* vol. 22, no. 1, pp.109-115, 2004.

[14] G.Hirzinger, B.Brunner, J.Dietrich, et al., "ROTEX-the first remotely controlled robot in space," *Proceedings of IEEE International Conference on Robotics and Automation,* vol. 3, pp. 2604-2611, 1994.

[15] M.Dissanayake, P.Newman, S.Clark, et al., "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation,* vol. 17, no. 3, pp.229-241, 1994.

[16] R.Brooks, "Aspects of mobile robot visual map making," *IEEE International Conference on Robotics and Automation,* vol. 2, pp. 824-829, 1985.

[17] J.Guivant, E.Nebot, and S.Baiker, "Localization and Map Building Using Laser Range Sensors in Outdoor Applications," *Journal of Robotic Systems,* vol. 17, no. 10, pp.565-583, 2000.

[18] J.E.Guivant, F.R.Masson, and E.M.Nebot, "Simultaneous localization and map building using natural features and absolute information," *Robotics and Autonomous Systems,* vol. 40, no. 2-3, pp.79-80, 2002.

[19] S.H.Cho, S.Lee, and W.Yu, "Use of Range Sensor Information for Improving Positioning Accuracy," *Proceedings of 17th World Congress the Internatonal Federation of Automatic Control ,* pp. 1669-1674, 2008.

[20] C.Fouque, P.Bonnifait, and D.Betaille, "Enhancement of global vehicle localization using navigable road maps and dead-reckoning," IEEE/ION Position, Location and Navigation Symposium, pp. 1286-1291, 2008.

[21] G.Dissanayake, S.Sukkarieh, E.Nebot, et al., "The Aiding of a Low-cost Strapdown Inertial Measurement Unit Using Vehicle Model Constraints for Land Vehicle Applications," *IEEE Transactions on Robotics and Automation,* vol. 17, no. 5, pp.731-747, 2001.

[22] S.Thrun, "Robotic Mapping: A survey," *Exploring artificial intelligence in the new millennium,* 1 ed. San Francisco: Morgan Kaufmann, 2003, pp. 1-35.

[23] A.Eliazar and R.Parr, "DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks," *Proceedings of the 18th international joint conference on Artificial intelligence*, pp. 1135-1142, 2003.

[24] A.Eliazar and R.Parr, "DP-SLAM 2.0," *Proceeding of IEEE International Conference of Robotics and Automation* , pp. 1314-1320, 2004.

[25] A.Nuchter, H.Surmann, K.Lingemann, et al., "6D SLAM with an Application in autonomous mine mapping," *Proceeding of IEEE International Conference of Robotics and Automation* , pp. 1998-2003, 2004.

[26] N.Karlsson, E.Bernardo, J.Ostrowski, et al., "The vSLAM Algorithm for Robust Localization and Mapping," *Proceedings of IEEE International Conference on Robotics & Automation* , pp. 24-29, 2005.

[27] R.Ouellette and K.Hirasawa, "A Comparison of SLAM Implementations for Indoor Mobile Robots," *IEEE/RSJ International Conference of Intelligent Robots and Systems*, pp. 1479-1484, 2007.

[28] S.Williams, G.Dissanayake, and H.F.Durrant-Whyte, "Towards Terrain-aided Navigation for Underwater Robotics," *Advanced Robotics*, vol. 15, no. 5, pp.533-549, 2001.

[29] S.Williams, P.Newman, J.Rosenblatt, et al., "Autonomous underwater navigation and control," *Robotica*, vol. 19, no. 5, pp.481-496, 2001.

[30] J.Kim and L.Ong, "Decentralized approach to unmanned aerial vehicle navigation: without the use of the global positioning system and preloaded maps," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 218, no. 6, pp. 399-416, 2004.

[31] S.Thrun, D.Hahnel, D.Ferguson, et al., "A system for volumetric robotic mapping of abandoned mines," *IEEE International Conference on Robotics and Automation* , vol. 3, no. 3, pp. 4270-4275, 2003.

[32] M.Montemerlo, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association," Doctorial Thesis, Carnegie Mellon University, 2003.

[33] A.Cooper, "A Comparison of Data Association Techniques for Simultaneous Localization and Mapping," Master Thesis, Massachusetts Institute of Technology, 2005.

[34] M.Isard and A.Blake, "*Contour Tracking by Stochastic Propagation of Conditional Density," Lecture Notes in Computer Science - Computer Vision*. Berlin / Heidelberg : Springer, pp. 343-356, 1996.

[35] Y.Bar-Shalom, X.R.Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York: John Wiley & Sons, Inc, 2001.

[36] S.J.Julier and J.K.Uhlman, "A New Extension of the Kalman Filter to Nonlinear Systems." *Proceedings of the Conference of Signal processing, sensor fusion, and target recognition VI* , pp. 182-193, 1997.

[37] M.S.Arulampalam, S.Maskell, N.Gordon, et al., "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing,* vol. 50, no. 2, pp.174-188, 2002.

[38] 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.

[39] M.Montemerlo, S.Thur, D.Koller, et al., "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association," *Eighteenth National Conference on Artificial Intelligence,* pp. 593-598, 2002.

[40] S.Thrun, C.Martin, Y.Liu, et al., "A Real-time Expectation Maximization Algorithm for Acquiring Multi-planar Maps of Indoor Environments with Mobile Robots," *IEEE Transactions on Robotics & Automation,* vol. 20, no. 3, pp.433-442, 2003.

[41] D.Anguelov, R.Biswas, D.Koller, et al., "Learning Hierarchical Object Maps of Non-stationary Environments with Mobile Robots," *Proceedings of the Conference on Uncertainty in Artificial Intelligence* , pp. 10-17, 2002.

[42] S.Thrun, "Particle Filters in Robotics," *Proceedings of the 17th Annual Conference on Uncertainty in AI,* pp. 511-551, 2002.

[43] S.Thrun, D.Fox, W.Burgard, et al., "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence,* vol. 128, no. 1, pp.99-141, 2001.

[44] N.Vlassis, B.Terwijn, and B.Krose, "Auxiliary Particle Filter Robot Localization from High-dimensional Sensor Observations," *Proceeding of IEEE International Conference of Robotics and Automation*, vol. 1, no. 1, pp. 7-12, 2002.

[45] D.Whitley, "A genetic algorithm tutorial," *Statistics and Computing,* vol. 4, no. 2, pp.65-85, 1994.

[46] N.J.Gordon, D.J.Salmond, and A.F.M.Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *IEE Proceedings F Rader and Signal Processing* , vol. 140, no. 2, pp. 107-113, 1993.

[47] G.Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *Journal of computational and graphical statistics,* vol. 5, no. 1, pp.1-25, 1996.

[48] C.Stachniss, G.Grisetti, and W.Burgard, "Recovering particle diversity in a Rao-Blackwellized Particle Filter for SLAM after actively closing loops," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* , pp. 18-22, 2005.

[49] N.Gordon, D.Salmond, and C.Ewing, "Bayesian state estimation for tracking and guidance using the bootstrap filter," *Journal of Guidance, Control, and Dynamics,* vol. 18, no. 6, pp.1434-1443, 1995.

[50] H.Durrant-Whyte and T.Bailey, "Simultaneous Localization and Mapping (SLAM): Part 1, the Essential Problem," *IEEE Robotics and Automation Magazine* , vol. 13, no. 2, pp. 99-110, 2006.

[51] T.Bailey and H.Durrant-Whyte, "Simultaneous Localisation and Mapping (SLAM): Part II State of the Art," *IEEE Robotics and Automation Magazine* , vol. 13, no. 3, pp. 108-117, 2006.

[52] B.Ristic, S.Arulampalam, and N.Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Application,* Boston, MA: Artech House, 2004.

[53] R.Brown and P.Hwang, *Introduction to Random Signals and Applied Kalman Filtering,* 3 ed, New York: Wiley, 1997.

[54] A.Gelb, *Applied Optimal Estimation*, Cambridge, MA: MIT Press, 1974.

[55] A.H.Jazwinski, *Stochastic Processing and Filtering Theory,* New York: Academic Press, 1970.

[56] R.Smith and P.Cheeseman, "On the Representation and Estimation of Spatial Uncertainty," *The International Journal of Robotics Research,* vol. 5, no. 4, pp.56-68, 1986.

[57] S.Ahn, M.Choi, J.Choi, et al., "Data Association Using Visual Object Recognition for EKF-SLAM in Home Environment," *IEEE/RSJ International*

Conference on Intelligent Robots and Systems , pp. 2588-2594, 2006.

[58] M.Bosse, P.Newman, J.Leonard, et al., "An Atlas for scalable mapping," *Proceeding of IEEE International Conference of Robotics and Automation* , vol. 2, pp. 1899-1906, 2003.

[59] S.Thrun, D.Koller, Z.Ghahamani, et al., "Simultaneous Mapping and Localization with Sparse Extended Information Filters: Theory and Initial Results," *Algorithmic Foundations of Robotics*. Berlin / Heidelberg: Springer, 2002, pp. 363-380.

[60] Y.Liu and S.Thrun, "Results for Outdoor-SLAM Using Sparing Extended Information Filters," *Proceeding of IEEE International Conference of Robotics and Automation,* vol. 1, pp. 1227-1233, 2003.

[61] J.Guivant and E.Nebot, "Optimization of Simultaneous Localization and Map-building Algorithm for Real-time Implementation," *IEEE Transactions on Robotics & Automation,* vol. 17, no. 3, pp.242-257, 2001.

[62] J.Guivant and E.Nebot, "Solving Computational and Memory Requirement of Feature-based Simultaneous Localization and Mapping Algorithms," *IEEE Transactions on Robotics & Automation,* vol. 19, no. 4. pp.749-754, 2003.

[63] D.Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Conference on Decision Making including 17th Symposium Adaptive Process* , vol. 17, pp. 1202-1211, 1978.

[64] D.Crisan, "Particle Filters: a Theoretical Perspective." *Sequential Monte Carlo Methods in Practice*. Berlin / Heidelberg: Springer, 2001, pp. 17-42.

[65] J.Liu and R.Chen, "Sequential Monte Carlo Methods for Dynamical Systems," *Journal of the American Statistical Association*, vol. 93, no. 443, pp.1032-1044, 1998.

[66] T.Kollar, "Optimal Robot Trajectories using Reinforcement Learning," Master Thesis, Massachusetts Institute of Technology, 2007.

[67] B.Efron, *The Jackknife, Bootstrap, and other Resampling Plans*, Philadelphia: SIAM, 1982.

[68] D.Crisan and A.Doucet, "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on Signal Processing,* vol. 50, no. 3,

pp.736-746, 2002.

[69] A.Kong, J.Liu, and W.H.Wong, "Sequential Imputations and Bayesian Missing Data Problems," *Journal of the American Statistical Association,* vol. 89, no. 425, pp.278-288, 1994.

[70] A.Doucet and N.Gordon, "Simulation-based Optimal Filter for Maneuvering Target Tracking," *SPIE Signal and Data Processing of Small Tragets,* vol. 3809, no. 1, pp.241-255, 1999.

[71] A.Doucet, S.Godhill, and C.Andrieu, "On sequential Monte Carlo methods for Bayesian filtering," *Statistics and Computing,* vol. 10, no. 3, pp.197-208, 2000.

[72] N.Bergman, "Recursive Bayesian Estimation," *Linköping Studies in Science and Technology*. Dissertations No. 579, Department of Electrical Engineering, Linkoping University, 1999.

[73] Y. Rui and Y.Chen, "Better Proposal Distributions: Object Tracking Using Unscented Particle Filter," *Computer Vision and Pattern Recognition,* vol. 2, pp.786-793, 2001.

[74] S. Thrun, D. Fox, and W. Burgard, "Monte Carlo Localization With Mixture Proposal Distribution." *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence ,* pp. 859-865, 2000.

[75] R. Van der Merwe, A. Doucet, N. De Freitas, et al., "The Unscented Particle Filter," *Advances in Neural Information Processing Systems*, Cambridge, MA: The MIT Press, 2000.

[76] W.Madow, "On the Theory of Systematic Sampling, II," *Annual of Mathematical Statistics,* vol. 20, no. 3, pp.333-354, 1949.

[77] M.Montemerlo and S.Thrun, "Simultaneous Localization and Mapping with Unknown Data Association using FastSLAM," *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1985-1991, 2003.

[78] D.Crisan, D.Moral, and T.Lyons, "Discrete filtering using branching and interacting particle systems," *Markov Processes and Related Filds,* vol. 5, no. 3, pp.293-318, 1999.

[79] D.B.Rubin, "Using the SIR Algorithm to Simulate Posterior Distribution," *Bayesian Statistics,* vol. 3, pp.395-402, 1988.

[80] A.Smith and A.Gelfand, "Bayesian Statistics without Tears: A

Sampling-resampling Perspective," *American Statistician,* vol. 46, no. 2, pp.84-88, 1992.

[81] N.Gordon, "Bayesian Methods for Tracking," Imperial College, University of London, 1993.

[82] J.Carpenter, P.Clifford, and P.Fearnhead, "Building Robust Simulation-based Filters for Evolving Data Sets," Department of Statistics, Oxford University, 1999.

[83] T.Higuchi, "Monte Carlo Filter using the Genetic Algorithm Operators," *Journal of Statistical Computation and Simulation,* vol. 59, no. 1, pp.1-23, 1997.

[84] D.Crisan and A.Doucet, "Convergence of Generalized Particle Filters," CUED/F-INFENG/TR 381, Cambridge Unverisity, 2000.

[85] R.Douc and O.Cappe, "Comparison of resampling schemes for particle filtering," *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis* , pp. 64-69, 2005.

[86] M. Bolic, P.M. Djuric, and S. Hong, "Resampling Algorithms and Architectures for Distributed Particle Filters," *IEEE Transactions on Signal Processing,* vol. 53, no. 7, pp.2442-2550, 2005.

[87] R.Merwe and A.Doucet, "The Unscented Particle Filter." CUED/F-INFENG/TR 380, Cambridge University Engineering Department, 2000.

[88] C.Andrieu, J.de Freitas, and A.Doucet, "Sequential MCMC for Bayesian Model Selection." *IEEE Higher Order Statistics Workshop,* pp. 130-134, 1999.

[89] W.R.Gilks and C.Berzuini, "Following a Moving Target-Monte Carlo Inference for Dynamic Bayesian Models," *Journal of the Royal Statistical Society.Serial B,* vol. 63, no. 1, pp.127-146, 2001.

[90] S.N.MacEachern, M.Clyde, and J.S.Liu, "Sequential Importance Sampling for Nonparametric Bayes Models: The Next Generation," *The Canadian Journal of Statistics,* vol. 27, no. 2, pp.251-267, 1999.

[91] G.Casella and E.George, "Explaining the Gibbs Sampler," *The American Statistician,* vol. 46, no. 2, pp.167-174, 1992.

[92] S.Chib and E.Greenberg, "Understanding the Metropolis-Hastings Algorithm," *The American Statistician,* vol. 49, no. 4, pp.327-335, 1995.

[93] J.Liu and R.Chen, "Blind Deconvolution via Sequential Imputations," *Journal of the American Statistical Association,* vol. 90, no. 430, pp.567-576, 1995.

[94] V.S.Zaritskii, V.B.Svetnik, and L.I.Shimelevich, "Monte-carlo Techniques in Problems of Optimal Information Processing," *Automation and Remote Control,* vol. 36, no. 3, pp.2015-2022, 1975.

[95] D.Avitzour, "A Stochastic Simulation Bayesian Approach to Multitarget Tracking." *IEE Proceedings of Radar, Sonar and Navigation*, vol. 142, no. 2, pp. 41-44, 1995.

[96] E.R.Beadle and P.M.Djuric, "A Fast Weighted Bayesian Bootstrap Filter for Nonlinear Model State Estimation," *IEEE Transactions on Aerospace and Electronic System,* vol. 33, no. 1, pp.338-343, 1997.

[97] C.Berzuini, N.G.Best, W.R.Gilks, et al., "Dynamic Conditional Independence Models and Markov Chain Monte Carlo Methods," *Journal of the American Statistical Association,* vol. 92, pp.1403-1412, 1997.

[98] A.Doucet, "On Sequential Simulation-based Methods for Bayesian Filtering." CUED/F-INFENG/TR 310, Department of Engineering, Cambridge University, 1998.

[99] P.Muller, "Monte Carlo Integration in General Dynamic Models," *Contemporary Mathematics*, vol. 94, no. 446, pp.590-599, 1991.

[100] M.Pitt and N.Shephard, "Filtering via Simulation: Auxiliary Particle Filters," *Journal of the American Statistical Association,* vol. 94, no. 446, pp.590-599, 1999.

[101] M.Montemerlo, S.Thrun, D.Koller et al., "FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges," *International Joint Conference on Artificial Intelligence,* pp. 1151-1156, 2003.

[102] A.Doucet, "Monte Carlo Methods for Bayesian Estimation of Hidden Markov Models: Applications to Radiation Signals," Doctorial Thesis, University Paris-Sud, 1997.

[103] J.Crassidis, "An Overview of Particle Filters with Applications to Aerospace Systems." Cambridge, MA.: *Technical Presentation at C.S. Draper Lab,* 2005,

[104] F.Daum and J.Huang, "Curse of Dimensionality and Particle Filters," *Proceedings of IEEE Aerospaces Conference* , vol. 4, pp. 1979-1993, 2003.

[105] A.Doucet, N.de Freitas, K.Murphy, et al., "Rao-blackwellised Particle Filters for Dynamic Bayes Networks," *Proceedings of 16th Annual Conference Uncertainty in AI* , pp. 176-183, 2000.

[106] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, Cambridge, MA: The MIT Press, 2004.

[107] M. Dorigo, V.Maniezzo, and A.Colorni, "Positive Feedback as a Search Strategy," Italy: Dipartimento di Elettronica, Politecnico di Milano, vol. 91-016, 1991.

[108] M. Dorigo, "Optimization, Learning and Natural Algorithms," Dipartimento di Elettronica, Politecnico di Milano, 1992.

[109] M. Dorigo, V.Maniezzo, and A.Colorni, "The Ant Systems: Optimization by a Colony of Cooperating Agents," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26, no. 1, pp.29-41, 1996.

[110] E.L.Lawler, J.K.Lenstra, A.H.G.Rinnooy Kan, et al., *Traveling Salesman Problem*, Chichester, UK.: John Wiley & Sons, 1985.

[111] G.Reinelt, "The Traveling Salesman: Computational Solutions for TSP Applications." *Lecture Notes in Computer Science*, Berlin, Germany :Springer-Verlag, 1994.

[112] R.Rubinstein, "The Cross-Entropy Method for Combinatorial and Continuous Optimization," *Methodology and Computing in Applied Probability,* vol. 1, no. 2, pp.127-190, 1999.

[113] D.T.Pham, A.Ghanbarzadeh, E.Koc, et al., "The Bees Algorithm: A Novel Tool for Complex Optimisation Problems," *Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems* , pp. 454-459, 2006.

[114] H.A.A.Bahamish, R.Abdullah, and R.A.Salam, "Protein Conformational Search Using Bees Algorithm," *Second Asia International Conference on Modeling & Simulation*, pp. 911-916, 2008.

[115] C.M.Fonseca and P.J.Fleming, "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, vol. 3, no. 1, pp.1-16, 1995.

[116] L.C.C.Fung, "Combined fuzzy-logic and genetic algorithm technique for

thescheduling of remote area power system," *IEEE Power Engineering Society Winter Meeting,* pp. 1069-1074, 2000.

[117] M.L.den Besten, T. Stutzle, and M. Dorigo, "Ant colony optimization for the total weighted tardiness problem," *Proccedings of 6th International Conference of Parallel Problem Solving from Nature,* vol. 1917, pp. 611-620, 2000.

[118] G.Di. Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," *Journal of Artificial Intelligence Research,* vol. 9, pp.317-365, 1998.

[119] L.M.Gambardella and M. Dorigo, "Ant Colony System hybridized with a new local search for the sequential ordering problem," *INFORMS Journal on Computing,* vol. 12, no. 3, pp.237-255, 2000.

[120] L.M.Gambardella, E.D.Taillard, and G.Agzzi, "MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows," *New Ideas in Optimization,* London, UK: McGraw Hill, 1999, pp. 63-76.

[121] D.Merkle, M.Middendorf, and H.Schmeck, "Ant colony optimization for resource-constrained project scheduling," *Proceedings of the Geneticand Evolutionary Computation Conference,* pp. 893-900, 2000.

[122] T. Stutzle and M. Dorigo, "ACO algorithms for the quadratic assignment problem." *New Ideas in Optimization,* London, UK: McGraw Hill, 1999, pp. 33-50.

[123] V.Cerny, "A thermodynamical approach to the traveling salesman problem," *Journal of Optimization Theory and Applications,* vol. 45, no. 1, pp.41-51, 1985.

[124] S.Kirkpartick, C.D.Gelatt Jr., and M.P.Vecchi, "Optimizaition by Simulated Annealing," *Science,* vol. 220, pp.671-680, 1983.

[125] F.Glover, "Tabu Search - Part I," *ORSA Journal on Computing,* vol. 1, no. 3, pp.190-206, 1989.

[126] F.Glover, "Tabu Search - Part II," *ORSA Journal on Computing,* vol. 2, no. 1, pp.4-32, 1990.

[127] F.Glover and M.Laguna, *Tabu Search*. Norwell, MA: Kluwer Academic Publishers, 1997.

[128] H.R.Lourenco, O.Martin, and T. Stutzle, "Iterated Local Search," *Handbook*

of Metaheuristics. The Netherlands: Kluwer Academic, pp. 321-353. 2002.

[129] L.J.Fogel, A.J.Owens, and M.J.Walsh, *Artificial Intelligence Through Simulated Evolution*. New York: John Wiley & Sons, 1966.

[130] J.Holland, *Adaptation in Natural and Artificial Systems*: Ann Arbor: University of Michigan Press, 1975.

[131] H.P.Schwefel, *Numerical Optimization of Computer Models*. New York: John Wiley & Sons, 1981.

[132] M. Dorigo, G.Di Caro, and L.M.Gambardella, "Ant Algorithms for Discrete Optimization," *Artificial Life,* vol. 5, no. 2, pp.137-172, 1999.

[133] M. Dorigo, G.Di Caro, and L.M.Gambardella, "The Ant Colony Optimization metaheuristic." *New Ideas in Optimization*. London, UK: McGraw Hill. 1999, pp. 11-32.

[134] T. Stutzle and H.H.Hoos, "MAX-MIN Ant System," *Future Generation Computer Systems,* vol. 16, no. 8, pp.889-914, 2000.

[135] K.Socha, J.Knowles, and M.Sampels, "A MAX-MIN ant system for the university timetabling problem," vol. 2463, pp. 1-13, 2002.

[136] M. Dorigo and L.M.Gambardella, "Ant Colonies for the Traveling Salesman Problem," *BioSystems,* vol. 43, no. 2, pp.73-81, 1997.

[137] M. Dorigo and L.M.Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," *IEEE Transactions on Evolutionary Compuatation,* vol. 1, no. 1. pp.53-66, 1997.

[138] L.M.Gambardella and M. Dorigo, "Solving Symmetric and asymmetric TSPs by ant colonies," *Proceedings of the IEEE International Conference on Evolutionary Computaion ,* pp. 622-627, 1996.

[139] T.Stutzle and M.Dorigo, "A short convergence proof for a class of ant colony optimizationalgorithms," *IEEE Transactions on Evolutionary Compuation,* vol. 6, no. 4, pp.358-365, 2002.

[140] R.Gary, *Entropy and Information Theory*. New York: Springer, 1990.

[141] G. Grisettiyz, C. Stachniss, and W. Burgard, "Improving Grid-based SLAM

with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling," *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2432-2437, 2005.

[142] L.E. Parker, "Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets," *Autonomous Robots,* vol. 12, no. 3, pp.231-255, 2002.

[143] P. Del Moral, "Measure valued processes and interacting particle systems. Application to nonlinear filtering problems," *Annals of Applied Probability,* vol. 8, no. 2, pp.438-495, 1998.

[144] J.R.Hershey and P. A. Olsen, "Approximating the Kullback Leibler divergence between Gaussian mixture models," *IEEE International Conference on Accoustic, Speech and Signal Processing* , vol. 4, pp. 317-320, 2007.

[145] P.J. Davis, *Interpolation and approximation*: New York: Dover Publications, 1963.

[146] T.Stutzle and M.Dorigo, "A short convergence proof for a class of Ant Colony Optimization algorithms," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 4, pp.358-365, 2002.

[147] G.Welch and G.Bishop, "An Introduction to the Kalman Filter," vol. 95-041, 2001. Univ. North Carolina at Chapel Hill.

[148] S.J.Julier and J.K.Uhlmann, "Unscented Filtering and Nonlinear Estimation," *Proceedings of the IEEE,* vol. 92, no. 3, pp. 401-422, 2004.

[149] E.Lukacs, "A Characterization of the Gamma Distribution," *The Annals of Mathematical Statistics,* vol. 26, no. 2, pp.319-324, 1955.

[150] R.Roll, "A Mean/Variance Analysis of Tracking Error," *The Journal of Portfolio Management,* vol. 18, no. 4, pp.13-22, 1992.

[151] A. Vale and M.I.Ribeiro, "A probabilistic approach for the localization of mobile robots in topological maps," *Proceedings of the 10th Mediterranean Conference on Control and Automation* , 2002.

[152] D.Koller and R.Fratkina, "Using Learning for Approximation in Stochastic Processes." *Proceedings of the International Conference on Machine Learning* , pp. 287-295, 1998.

[153] J.A.Rice, *Mathematical statistics and data analysis*, Belmont, CA: Duxbury Press, 1995.

[154] P.Alfredo Toriz, L.Abraham Sanchez, and A.L.Maria, "Coordinated Multi-robot Exploration with SRT-Radial," *Proceedings of the 11th Ibero-American conference on AI: Advances in Artificial Intelligence* , pp. 402-411, 2008.

[155] W.Burgard, M.Moors, C.Stachniss et al., "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, pp.376-386, 2005.

[156] M.Bolic, P.M.Djuric, and S.Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Processing,* vol. 53, pp.2442-2450, 2004.

[157] Mathworks Inc., "Distributed Computing Toolbox of Matlab," 2006.

[158] M.Wolfe and C.W.Tseng, "The Power Test for Data Dependence," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, pp.591-601, 1992.

[159] R.Simmons, D.Apfelbaum, W.Burgard et al., "Coordination for multi-robot exploration and mapping," *Proceeding of 7th International Symposium on Experimental Robotics* , pp. 852-858, 2000.

[160] S.Thrun and Y.Liu, "Multi-Robot SLAM with Sparse Extended Information Filers." *The Eleventh International Symposium*, pp. 254-265, 2005.

[161] J.Ko, B.Stewart, D.Fox, et al., "A Practical, Decision-theoretic Approach to Multirobot Mapping and Exploration," *IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 3232-3238, 2003.

[162] A.Howard, "Multi-robot Simultaneous Localization and Mapping using Particle Filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp.1243-1256, 2006.

[163] N.Roy and G.Dudek, "Collaborative Robot Exploration and Rendezvous: Algorithms, Performance Bounds and Observations," *Autonomous Robots,* vol. 11, no. 2, pp.117-136, 2001.

[164] T.Bailey and H.Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics & Automation Magazine,* vol. 13, no. 3, pp.108-117, 2006.

[165] R.C.Gonzalez and R.E.Woods, *Digital Image Processing,* 3 ed. Upper Saddle River, NJ: Prentice Hall, 2007.

[166] P. E. Rybski, A. Larson, H. Veeraraghavan et al., "Communication strategies in multi-robot search and retrieval: Experiences with mindart," *Distributed Autonomous Robotic Systems 6*. pp.317-326, 2007.