2010 11th Int. Conf. Control, Automation, Robotics and Vision
Singapore, 7-10th December 2010

Mapping Large Scale Environments By Combining Particle Filter and Information Filter

Mahesh Mohan
IIIT Hyderabad
Hyderabad, India
mahesh_mohan@research.iiit.ac.in

K MadhavaKrishna
IIIT Hyderabad
Hyderabad, India
mkrishna@iiit.ac.in

***Abstract*—This paper presents two approaches to combine two popular mapping strategies, namely Particle Filters and Information Filters. The first method describes how the Particle Filter can be incorporated into the Information Filter framework by building local submaps using the Particle Filter and combining them using a Information Filter to obtain a global map. Using the Particle Filter locally reduces the linearization errors and is useful in handling ambiguous data associations, while the Information Filter keeps track of the uncertainty over long periods of time, thereby avoiding FastSLAM's tendency to become overconfident. The second method shows how the Information Filter can be used in the Particle Filter framework as a simple means of remembering the filter's uncertainty. This can then be used to repopulate particles while closing loops. This not only handles non linearities but is also robust for loop closing because, unlike the Particle Filter, the Information Filter does not exhibit forgetfulness of the trajectory's past.**

***Index Terms*—SLAM, Information Filter, Particle Filter**

I. Introduction

The idea behind SLAM is to estimate the position of the robot and the positions of the landmarks, given noisy control and measurement data. In other words, the aim is to estimate the distribution given by,

$$p(x_{0:t}, m|z_{1:t}, u_{1:t}) \tag{1}$$

Here, x_t is the state of the robot, z_t is the measurement, u_t is the control action performed at time t respectively. m denotes the map. This version of the SLAM problem is known as the full SLAM problem since we estimate both the map and the entire robot trajectory. In the online SLAM problem we only estimate the current pose of the robot instead of the entire trajectory. This is given as,

$$p(x_t, m|z_{1:t}, u_{1:t}) \tag{2}$$

The complete list of approaches to address this widely studied problem is beyond the scope of this paper, but the popular approaches can be roughly divided into 3 general categories. The first set of approaches are based on the Extended Kalman Filter Framework. These approaches generally, linearize the motion and measurement models and use this in a Kalman Filter framework to estimate the robot's current position and the map simultaneously. Despite its theoretical elegance, these methods have 3 major problems, namely, divergence due to linearization errors, inability to recover from errors in data association and a measurement update step of complexity $O(n^2)$. Here n is the number of landmarks in the map. Needless to say, this becomes expensive when the map grows large.

The second category of methods have been collectively described as GraphSLAM in [1]. They are offline methods that accumulate both control and measurement information into a graph, without resolving it into an estimate of the robot's position or the map. The robot's trajectory and the map may be obtained through an additional inference step. Thus they can be seen as the solution to the full SLAM problem, instead of the online SLAM problem that is solved by the EKF framework. The posterior being estimated is factorized as:

$$\begin{aligned} p(x_{0:t}, m|z_{1:t}, u_{1:t}) &= \eta p(x_0) \\ &\qquad \prod_t p(x_t|x_{t-1}, u_t) p(z_t|x_t, m) \end{aligned}$$

Examples of this approach are [7], [5], [1]. The creation of the graph can be seen as using the Information Filter as described in [1].The advantages of this approach include, the ability to undo data misassociations, linearizing multiple times and consequently ability to produce better maps of larger orders of magnitude, than can be handled by EKF and its variants. Disadvantages are, size of the graph grows linearly with time and the retrieval of the path and the map requires an additional inference step. Regardless of the nature of the graph maintained, this step is computationally, expensive enough to prohibit regular inference. Methods like SEIF based SLAM [1], attempt to solve the online SLAM problem using the Information Filter, their pros and cons are given in [8].

The third category of approaches treats the SLAM problem as a Markov Chain Monte Carlo process. It relies on the important observation that the landmark poses are independent given the robot's trajectory. This leads to an elegant refactorization of the posterior, which reduces the complexity of both the measurement and prediction updates. The basic idea is to represent the belief at every step by a weighted set of samples instead of defining it over the entire map. The prediction update then produces hypotheses for the robot pose by sampling based on the prediction model. These hyotheses are weighted by using the measurement model and the final posterior is obtained by resampling based on these weights. the advantages of this approach are, it is faster and it can handle nonlinearities better, since no linearization is done before resampling. However, this approach is susceptible to sample impoverishment, where a majority of the trajectories are weeded out due to resampling and only a few distinct trajectories are retained. This causes problems when the environment contains large loops.

Thus it is evident that no single category has a failproof answer to the SLAM problem. Each of them come with their own inherent advantages and disadvantages. The natural step to take next, would be to see if a combination of these approaches results in methods that ameliorate their individual weaknesses. One such hybrid approach is Hybrid SLAM [3], that proposes a combination of a Particle Filter based front end and an EKF based back end. The resulting method does not suffer from sample impoverishment that plagues Particle Filters and also fairly resistant to linearization errors (especially in cluttered environments) that plagues EKF based approaches. However, everytime the local submap generated by the Particle Filter front end is merged into the global map, hard data association decisions must be taken that cannot be undone. This restricts the size of the map that can be acquired.

The next section describes a method to compare the severity of the sample impoverishment problem for different Particle Filtering methods. We then propose that the Particle Filter and the Information filter can be combined in two ways. The first method is to use a Particle Filter to build local submaps and finally obtain the global map by fusing these local submaps using an Information Filter. This is described in detail in Section 3. The second method is to improve the sampling density of a Particle Filter by sampling from an Information Filter based likelihood function. This is described in Section 4.

II. Forgetfulness In FastSLAM

This section aims to take a look at the sample impoverishment problem that plagues FastSLAM. The reader is refered to [?] for a detailed explanation. Broadly speaking, the problem lies in the fact that, FastSLAM uses a finite number of samples to represent a posterior over the space of all possible trajectories (which grows exponentially). As a result, the resampling process causes a loss in diversity in the trajectories maintained and the sampled representation quickly becomes optimistic. The result is a set of trajectories that are largely similar to each other, in terms of their ancestry. The aim here is to describe a method to compare the forgetfulness of different Particle Filter schemes.

An indicator of the severity of the sample impoverishment problem is the probability that a given trajectory survives k resamplings. If the probability is linearly dependent on all the previous states and observations, the problem is not so severe.

Definition 1. *The Particle filter (using n samples) is said to have* forgotten *its history from steps 1 to $t-k$ if,*

$$x_j^1 = x_j^2 = ... = x_j^n, for 1 \leq j \leq t-k$$

or in other words, if,

$$\Sigma_j^t = 0, for 1 \leq j \leq t-k$$

Lemma 1. *In FastSLAM, the probability that a trajectory $x_{0:t}$ and map m survives after k steps is $\propto \prod_{i=1}^{k} p(z_i|x_i)^{k-i+1}$*

Proof: For the posterior, $p(x_{0:t}, m|u_{1:t}, z_{1:t})$, there are two ways to generate a sampled set representation. Firstly, we could sample directly from the true posterior (which of course is impossible in practice). This true posterior is represented as $p_{true}(x_{0:t})$. Secondly, we could use the Particle Filter factorization where the posterior is given as,

$$p_{FS}(x_{0:t}) = res(p(x_t|x_{t-1}, u_t)p_{FS}(x_{0:t-1}))$$

Now, the probability that a particular trajectory and map $x_{0:t}, m$ is chosen at the end of the kth step is given for the first method by,

$$p_{true}(x_{0:k}) = p(x_0) \prod_{i=1}^{k} p(z_i|x_i)p(x_i|x_{i-1}, u_i)$$

whereas for the second method it is given by,

$$p_{FS}(x_{0:k}) = res(p(x_k|x_{k-1}, u_k)p_{FS}(x_{0:k-1}))$$

Under the assumption that the posterior obtained by FastSLAM is equivalent to the true posterior, we may assume that the two methods are equivalent. Thus we have,

$$p_{FS}(x_{0:t}) \quad \Leftrightarrow \quad p_{true}(x_{0:t})$$

In other words, resampling based on weights at step k in FastSLAM is equivalent to actually sampling from the true posterior at step k. The probability that a trajectory $x_{0:t}$ survives at time t (denoted as $p_{FS}^{survival}(x_{0:t})$)is given by,

$$\begin{aligned} p_{FS}^{survival}(x_{0:t}) &= res(p(x_t|x_{t-1}, u_t)p_{FS}(x_{0:t-1})) \\ &= res(p(x_t|x_{t-1}, u_t)p_{FS}^{survival}(x_{0:t-1})) * \\ &\quad res(p(x_{t-1}|x_{t-2}, u_{t-1})p_{FS}^{survival}(x_{0:t-2})) \\ &= (p(x_0)\prod_{i=1}^{k} p(z_i|x_i)p(x_i|x_{i-1}, u_i)) * \\ &\quad (p(x_0)\prod_{i=1}^{k-1} p(z_i|x_i)p(x_i|x_{i-1}, u_i)) * \\ &\quad ...(p(x_0)) \\ &= \prod_{j=0}^{k}\prod_{i=0}^{j} \rho_i \\ &\propto \prod_{i=1}^{k} p(z_i|x_i)^{k-i+1} \end{aligned}$$

where, $\rho_i = p(z_i|x_i)p(x_i|x_{i-1}, u_i)$. Note, the chaining is important in the context of resampling based schemes (and will not occur in the context of methods like EKF), since the probability of survival of a trajectory at step t is not indicative of whether it will actually be retained at step t. For example, consider trajectories $x_{0:t}^{[1]}$ and $x_{0:t}^{[2]}$. Suppose that, the first trajectory had a better weight at time t, but the second had a better weight at time $t+1$. This would imply that the second trajectory would not have been sampled at time t. Hence the probability of survival of a trajectory at step t is not indicative of whether it will actually be retained at step t. ∎

This leads us to the following definition of forgetfulness that allows us to compare the forgetfulness of various Particle Filter based methods.

Definition 2. *A Particle Filter based method A is said to be more* forgetful *than a Particle Filter based method B, if the probability of survival of a hypothesis is lower in A than in B, i.e.*

$$p_A^{survival}(x_{0:t}) \leq p_B^{survival}(x_{0:t}) \tag{3}$$

This definition will come in handy when we are comparing the performance of the proposed Particle Filter based schemes.

III. Particle Filter in an Information Filter Framework

The Information Filter operates by building a graph of relative constraints between successive robot poses and the landmark positions. It stores this information in the Information matrix Ω and Information vector ξ. In order to reduce the size of Ω, ξ, in practice, multiple scans are combined to form a local submap. In this section, we propose a method that uses the Particle Filter to build such local submaps. Once these local submaps have been built, their relative transformations are obtained by scan correlation. These are then incorporated into Ω, ξ using a separately defined measurement model.

A. Intuitive description

In the Information Filter framework, the full SLAM problem is refactored as,

$$\begin{aligned} p(x_{0:t}, m|z_{1:t}, u_{1:t}) &= \eta p(x_0) \prod_t p(x_t|x_{t-1}, u_t) \\ & \quad p(z_t|x_t, m) \end{aligned}$$

Many implementations accumulate data obtained from multiple scans into a single local submap in order to reduce the size of the information matrix. The rationale can be explained by considering a compound motion model and a compound measurement model. The normal motion model returns $p(x_t|x_{t-1}, u_t)$. On the other hand, a compound motion model returns $p(x_t, m_{tk}|x_{t-k}, u_{t-k+1:t}, z_{t-k+1:t})$, where m_{tk} is the submap observed from $t-k$ to t. More specifically, this can be seen as describing a motion model

$$p(X_t|X_{t-1}, U_t)$$

from X_{t-1} to X_t, where the state X_{t-i} is given by, x_{t-i*k} and U_t is given by the function, $com(u_{t-k+1:t}, z_{t-k+1:t})$. Here the exact nature of com is not important, as we shall see. What is important,however is the observation that the above motion model can be approximated by a Gaussian.

Before we proceed, we note that we have 3 things, $x_{t-k}, u_{t-k+1:t}, z_{t-k+1:t}$ and we have to estimate the posterior over x_t. This is similar to the online SLAM problem for a subset of the total set of states. This may be solved using any of the methods in the literature. We advocate the use of a Particle Filter, since it is robust with respect to nonlinearity and data association ambiguity. The actual problem arises from the fact that the Particle Filter maintains the posterior as a set of weighted samples. The procedure to convert the set of weighted samples into a Gaussian, given in [3], is described below.

Each particle in the posterior P_i with weight w_i, reresenting landmarks $[\mu_{L_1}, \Sigma_{L_1}, \mu_{L_2}, \Sigma_{L_2}, ...]$ and trajectory $traj$, can be represented by a Gaussian Mixture Model given by,

$$P_i = < w_i, \mu_i, \Sigma_i >$$

where, $\mu_i = [traj, \mu_{L_1}, \mu_{L_2} ....]$ and $\Sigma_i = [\Sigma_{traj}, \Sigma_{L_1} ....]$

Hence, if the posterior contain n particles, they can be considered to be n GMMs. These can then be combined using moments matching to obtain a single Gaussian whose mean μ and variance Σ are given by,

$$\begin{aligned} \mu &= \sum_p (w_p \mu_p) \qquad (4) \\ \Sigma &= \sum_p (w_p(\Sigma_p + (\mu_p - \mu)(\mu_p - \mu)^T)) \end{aligned}$$

Thus we see that, regardless of the nature of U_t, we can obtain a Gaussian for the model, $p(X_t|X_{t-1}, U_t)$. The conversion of the particle set into a Gaussian posterior in case of unknown correspondences is also discussed in [3]. The choice of k depends upon the environment, the motion model, the measurement model and the number of particles being used. Naturally, the lesser the noise in the motion and measurement models, the larger k can be.

The above procedure yields a local submap along with a being a motion model. The compound measurement model depends on the nature of the local submap. If it is a landmark based map, then each submap can be viewed as a GMM, since each landmark is represented by a Gaussian. If it is a raw data based map, it can be converted into a GMM as shown in [4].Either way, once we reduce the local submaps to a sum of Gaussians representation, the required measurement model can be obtained by the procedure outlined in [4].

B. Comparisons

1) With Standard GraphSLAM: The standard implementation of the Information Filter to the SLAM problem is GraphSLAM, as described in [13]. The exact method described therein, is an offline method. With respect to standard GraphSLAM, the proposed method results in lower linearization errors.

This is because, in the standard approach to GraphSLAM, the action and measurement models are linearized with respect to the estimated state. Hence, the error due to linearization depends on the error in the estimated state (especially on the uncertainty in heading). This may grow with time, resulting in poor map estimates. In the proposed method, by using the Particle Filter to estimate the local submaps, we are reducing the number of linearizations of the action and measurement models (only once per local submap). Also, the heading uncertainty within each submap can be assumed to be small, thereby improving the quality of the maps produced.

Algorithm 1 Method A

```
$\Omega \leftarrow \infty, \xi \leftarrow 0$
$ps \leftarrow initializeParticles(x_0)$
for all t do
    $ps \leftarrow PFactionUpdate(ps, u_t)$
    $ps \leftarrow PFmeasurementUpdate(ps, z_t)$
    if stepNum = k or loop detected then
        $\mu, \Sigma, LS_{cur} \leftarrow obtRelative\mu And\Sigma(ps)$
        $ps \leftarrow initializeParticles(\mu)$
        $\Omega, \xi \leftarrow IFactionUpdate(\Omega, \xi, \mu, \Sigma)$
        $\hat{z}_t, Q_t \leftarrow scan\_align(LS_{cur}, LS)$
        $\Omega, \xi \leftarrow IFmeasurementUpdate(\hat{z}_t, Q_t)$
        $LS \leftarrow append(LS, LS_{cur})$
    end if
    if loop detected then
        $\Omega, \xi \leftarrow reduce(\Omega, \xi)$
        $\Sigma_{traj} \leftarrow \Omega^{-1}$
        $\mu_{traj} \leftarrow \Sigma^{-1}\xi$
    end if
    $stepNum \leftarrow stepNum + 1$
end for
```

2) With Standard FastSLAM: The standard implementation of the Particle Filter to the SLAM problem is FastSLAM, as described in [1]. The main advantage of the proposed method with respect to standard FastSLAM is lesser overconfidence.

Lemma 2. *Splitting a large trajectory into multple subtrajectories and estimating each subtrajectory serparately results in lesser forgetfulness than estimating a single long trajectory, i.e.*

$$p^{survival}_{submaps}(x_{0:t}) \geq p^{survival}_{noSubmaps}(x_{0:t}) \tag{5}$$

Proof: Consider the case, where we split the trajectory into smaller trajectories of size k' each. In this case, the probability that the trajectory defined by $x_{0:k}$ survives is given by the product of the probabilities that each sub trajectory survives in the submap,

$$\begin{aligned} p^{survival}_{submaps}(x_{0:k}) &= p^{survival}_{submap1}(x_{0:k'})p^{survival}_{submap2}(x_{k':2k'})..(6) \\ &= \prod_{i=0}^{k/k'}\prod_{j=0}^{k'} \rho_{j+i*k'} \\ &> \prod_{j=0}^{k}\prod_{i=0}^{j} \rho_i \end{aligned}$$

■

Since, the probability that a trajectory survives is higher in the proposed method, we can assume that the proposed method has a lesser tendency to be forgetful and consequently, its particle set diversity is higher. This implies that the proposed method can be used to close larger loops than traditional FastSLAM.

IV. Information Filter in a Particle Filter Framework

The Particle Filter represents the required posterior using a set of weighted samples. It is a well known fact that this sample set exhibits exponential forgetting of its past due to the resampling process. Hence, we use the Information matrix and the Information vector as a means of remembering the past uncertainty about the relative transformation between successive robot poses. The sample set can then be repopulated by using the knowledge of this uncertainty, by Gibbs resampling.

A. Intuitive description

The major problem in using the Particle Filter for building large scale maps is its tendency to become overconfident very soon. In other words, the Particle Filter quickly loses diversity in the trajectories maintained, thereby reducing its ability to close large loops. The reason for this is the exponential growth of the trajectory space, (since FastSLAM works in trajectory space and not state space).

Thus we see, sample impoverishment causes the particle set to cease to be representative of the actual posterior being estimated (it actually makes it highly optimistic [9]). This further means that the weighting scheme becomes less and less accurate as time progresses, leading to its inability to close large loops.

The method described in the previous section addressed this problem by breaking up the map into submaps. The Particle Filter was used to estimate only the local submap, thereby making the method online only with respect to the current local submap. Though this is sufficient in most cases, there might arise a need to actually maintain an online pose and map estimate with respect to the global map (for example, if the robot is exploring the environment).

Therefore, in this section, we use the Particle Filter to build a global map. The ”forgetfulness” of the Particle Filter is counteracted by maintaining a corresponding Information matrix Ω and Information vector ξ. They maintain the relative transformation between the robot's poses. In contrast to the previous section, here, we do not obtain the trajectory by the standard inference step of the Information Filter. Instead we obtain the relative transformation between poses by inverting the appropriate submatrix of Ω. The relative transformations between all the given poses forms the set of conditional distributions for the full SLAM posterior. So we can simply use Gibbs resampling to obtain samples from the full posterior. This will ensure that the particle set is a better representation of the actual posterior. Before we describe the method, we describe the Gibbs resampling method, with the appropriate modifications to suit our purposes.

B. Gibbs Resampling

Gibbs sampling or Gibbs sampler is an algorithm to generate a sequence of samples from the joint probability distribution of two or more random variables. The purpose of such a sequence is to approximate the joint distribution, or to compute an integral (such as an expected value). Gibbs sampling is a special case of the MetropolisHastings algorithm, and thus an example of a Markov chain Monte Carlo algorithm.

Gibbs sampling is applicable when the joint distribution is not known explicitly, but the conditional distribution of each variable is known. The Gibbs sampling algorithm generates an instance from the distribution of each variable in turn, conditional on the current values of the other variables. It can be shown that the sequence of samples constitutes a Markov chain, and the stationary distribution of that Markov chain is just the sought-after joint distribution.

Suppose, we have to generate samples from a function $P(x_{0:t})$. Gibbs resampling is a sampling strategy that is used when it is impossible to obtain samples from $P(x_{0:t})$ directly and $P(x_{0:t})$ is defined in terms of its conditional distributions. In our case, $x_{0:t}$ is the trajectory being estimated and $P(x_{0:t})$ is the complete posterior over the trajectory space,$p(x_{0:t}|z_{1:t}, u_{1:t})$. Hence, in order to apply Gibbs resampling, we need an expression for its conditionals $p(x_i|x_1, x_2...x_t, z_{1:t}, u_{1:t})\ \forall i$. Note that this essentially denotes the relative transformation between x_{i-1} and x_i when all other states are known. This reduces to $p(x_i|x_{i-1}, u_{1:i-1}, z_{1:i-1})$, which can be retrieved from the diagonal elements of the reduced Information matrix Ω.

The procedure to generate the samples, given $p(x_i|x_{i-1})\ \forall i$ is given as,

$$\begin{array}{rcl} x_2^k & \sim & p(x_2^k|x_1^{k-1}, x_3^{k-1}...x_t^{k-1}, u_{1:t}, z_{1:t}) \\ & \sim & p(x_2^k|x_1^{k-1}, u_{1:t}, z_{1:t}) \\ x_3^k & \sim & p(x_3^k|x_2^k, u_{1:t}, z_{1:t}) \\ . & & \\ . & & \\ x_t^k & \sim & p(x_t^k|x_{t-1}^k, u_{1:t}, z_{1:t}) \end{array}$$

The states, $x_{1:t}^k$ represent one trajectory sample. This process is repeated to generate the n trajectory samples required.

C. Algorithm Description

In order to make the estimation of the full SLAM posterior an online process, FastSLAM samples from the proposal distribution, $p(x_t|x_{t-1}, u_t)p(x_{1:t-1}|u_{1:t-1}, z_{1:t-1})$, and assigns an importance weight to each sample based on the measurement model, $p(z_t|x_t)$. This makes the process much faster, but introduces the exponential dependence of the filter on its previous states, discussed in Section 2. To avoid this dependence on the previous states, we need to directly sample from the complete posterior.

Now three questions arise, first, how do we build the complete posterior? Second, how do we sample from it? And lastly, when do we resample from the complete posterior?

The complete posterior $p(x_{0:t}, m|z_{1:t}, u_{1:t})$ can be broken down into a product of its conditionals, as,

$$p(x_{0:t}, m|z_{1:t}, u_{1:t}) = \prod_i p(x_t, m_t|x_{t-1}, u_t)p(z_t|x_t) \quad (7)$$

Thus we can obtain samples from the complete posterior by sampling from the conditionals.

In order to obtain the conditionals, we can use the Information matrix Ω and Information Vector ξ (constructed using the same procedure described in the previous section). It follows from the method of their construction that the reduced form of Ω and ξ can be used to obtain the entire trajectory and its variance. However, we note that such an inference is quite expensive since it involves the inversion of quite a large matrix. Also to be noted is that, by appropriate transformations, the relative transformation between successive states can be obtained at a significantly lower cost. To see how, consider the reduced form of Ω and ξ, which can be visualized as a graph whose nodes are the states (x_0 to x_t) in the robot's trajectory and the edges are the constraints between them. Suppose the state x_t is dependent on x_2, indicating a loop. Hence in order to obtain the transformation between x_2 and x_t, the entire submatrix (from indexes 2 to t) needs to be inverted. But in order to obtain the relative transformation between x_3 and x_4, we need to invert only that submatrix (from indexes 3 to 4).

Thus we see, states that are not loop states (i.e the first and last states of a loop) will have their corresponding off diagonal elements in Ω as zero. Thus, by making appropriate transformations (to shift x_3 to the origin, in this case), we can obtain the relative transformation between x_3 and x_4 by inverting their corresponding submatrix. This can be done in $O(1)$ time.

But what about loop states? How do we obtain the relative transformations for those states? In this case, there is no option but reduce the matrix further for all the non loop states that form a part of the loop, till all the intermediate states are replaced by the relative transformations. At the end of this step, the 4 states in the loop will remain, the two loop states (x_2 and x_t), the state immediately after the start of the loop (x_3) and the state immediately before the end of the loop (x_{t-1}). All other intermediate states are subsumed in the relative transformation between these states. Now we just have to invert a submatrix of size 4, which is again $O(1)$ time. The only problem is that $O(t)$ reductions are required. Since for t states, there are $O(t)$ relative transformations to be estimated, the worst case complexity for this process becomes $O(t^2)$. But in practice, very few of the states are actually loop states, thereby making this process quite efficient (almost close to $O(t)$).

Now that we have the conditionals, all we need to do to resample from the complete posterior is to resample from these conditionals using Gibbs resampling.

This now brings us to the last question, when should we resample from the complete posterior? While there is no clear theoretical way to decide this, we can resample, every time the effective number of particles required to represent the posterior, N_{eff}, (given as $\frac{1}{\sum_i w_i^2}$) falls below a prespecified threshold or if a loop is detected. This threshold is map specific and determining it in practice is one of the avenues for future research.

Note, the function being approximated is the same as in traditional FastSLAM, but the procedure employed is different. FastSLAM approximates the posterior by first obtaining weights for the current step and then combining it with the past weights. This results in forgetfulness but is efficient enough to make the method usable in real time. On the other hand, the above method will maintain diversity at the cost of increased update complexity.

Algorithm 2 Method B

```
$\Omega \leftarrow \infty, \xi \leftarrow 0$
$ps \leftarrow initializeParticles(x_0)$
$ps \leftarrow PFactionUpdate(ps, u_t)$
$\mu, \Sigma \leftarrow obtRelative\mu And\Sigma(ps)$
$\Omega \leftarrow \Omega + \Sigma$
$\xi \leftarrow \xi + \mu$
$\Omega, \xi \leftarrow IFmeasurementUpdate(z_t)$
$N_{eff} \leftarrow \frac{1}{\sum_i w_i^2}$
if loop detected or $N_{eff} < threshold$ then
    $\Omega, \xi \leftarrow reduce(\Omega, \xi)$
    $conditionals \leftarrow obtainConditionals(\Omega, \xi)$
    $ps \leftarrow gibbsResampling(conditionals)$
end if
$ps \leftarrow PFmeasurementUpdate(ps, z_t)$
```

D. Comparisons

1) With FastSLAM: The important difference in the proposed resampling scheme and the traditional resampling scheme is that if a particle is weeded out by traditional resampling, the entire trajectory is lost and cannot be recovered again, whereas in the proposed scheme, it is the relative transformation that is resampled. Hence the entire trajectory is not lost. This makes the proposed method more resistant to sample impoverishment compared to traditional FastSLAM. To see why, let us compare the probability of a trajectory p surviving k steps is,

$$\begin{aligned} p_{Gibbs}^{survival}(x_{0:t}) &= p(x_{0:t}|z_{1:t}, u_{1:t}) \\ &= \prod_i p(z_i|x_i)p(x_i|x_{i-1}, u_i) \end{aligned} \tag{8}$$

Instead of,

$$p_{FS}^{survival} = \prod_i (p(z_i|x_i)p(x_i|x_{i-1}, u_i))^{t-i+1}$$

Thus it can be seen that in the proposed method, the samples are not exponentially forgotten. Instead, there is only a linear dependence, thereby reducing the tendency of the filter to forget past hypotheses. This improves the diversity in the set of maintained trajectories, enabling it to close bigger loops. We repeat, however, this comes at the price of increased time complexity in the sampling step.

2) With GraphSLAM: The Gibbs resampling step is a faster way of obtaining the complete path from the Information Matrix and Information Vector compared to the matrix inversion step used in GraphSLAM. To see why, consider, the complexity of each step in the proposed method.

1) reduction step $= O(N)$
2) obtaining the conditionals $= O(t)$
3) gibbs resampling $= O(nt)$
4) resampling $= O(n)$

Thus the overall complexity of the proposed method is $O(N + nt)$, where n is the number of particles being used, t is the length of the trajectory and N are the total number of landmarks. The complexity of GraphSLAM is given as,

1) reduction step $= O(N)$
2) path inference $= O(t^{2.4})$ (involves the matrix inversion)

Thus the overall complexity is $O(N + t^{2.4})$. For large trajectories, this becomes much larger that the $O(N + nt)$ complexity for the proposed method.

V. Conclusion

This paper presents two methods that combine the use of the Particle Filter and the Information Filter. The first uses the Particle Filter to build local submaps and combines them into a global map using the Information Filter. A mathematical derivation for the method was provided to justify it. The second method uses the Particle Filter to build the global map online and uses the Information Matrix to remember its past uncertainty. Whenever the particle diversity is too low, (for example, while closing loops), the particles are resampled using a Gibbs resampling strategy. The advantages of both methods with respect to traditional FastSLAM and GraphSLAM are described.

References

[1] Thrun et al, *Probabilistic Robotics*, MIT press
[2] Tim Bailey, *Mapping and Localization in extensive outdoor environments*, PHD thesis, University of Sydney, Australian Center for Field Robotics, 2002
[3] Alex Brooks, Tim Bailey, *HybridSLAM: Combining FastSLAM and EKF SLAM for reliable mapping*,
[4] Juan Nieto Tim Bailey and Eduard Nebot, *ScanSLAM: Combining EKF-SLAM and scan correlation*,
[5] Michael Bosse et al, *An Atlas framework for scalable mapping*, ICRA 2003
[6] Frese et al, *A Multilevel Relaxation Algorithm for Simultaneous Localization and Mapping*, IEEE Transaction On Robotics,VOL. 21, NO. 2,2005
[7] Tardos et al, *Robust mapping and localization in indoor environments using sonar data*, IJRR, 2002
[8] Thrun et al, *Simultaneous mapping and localization with sparse extended information filters*, Proceedings of the fifth Intl. Workshop on Algorithmic Foundations of Robotics, 2002
[9] Tim Bailey et al, *Consistency of the FastSLAM Algorithm*
[10] J.A. Castellanos, J.D. Tardos, and G. Schmidt, *Building a global map of the environment of a mobile robot: The importance of correlations.* In IEEE International Conference on Robotics and Automation, pages 10531059, 1997.
[11] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem With Unknown Data Association.* PhD thesis, Carnegie Mellon University, 2003.
[12] R. Smith, M. Self, and P. Cheeseman. *A stochastic map for uncertain spatial relationships.*, In International Symposium of Robotics Research, pages 467474, 1987.
[13] M.W.M.G. Dissanayake et al, *A solution to the simultaneous localization and map building (SLAM) problem.* IEEE Transactions on Robotics and Automation, 17(3):229 241, 2001.