

Fixed-lag Sampling Strategies for Particle Filtering SLAM

Kristopher R. Beevers and Wesley H. Huang

Abstract—We describe two new sampling strategies for Rao-Blackwellized particle filtering SLAM. The strategies, called *fixed-lag roughening* and the *block proposal distribution*, both exploit “future” information, when it becomes available, to improve the filter’s estimation for previous time steps. Fixed-lag roughening perturbs trajectory samples over a fixed lag time according to a Markov Chain Monte Carlo kernel. The block proposal distribution directly samples poses over a fixed lag from their fully joint distribution conditioned on all the available data. Our experimental results indicate that the proposed strategies, especially the block proposal, yield significant improvements in filter consistency and a reduction in particle degeneracies compared to standard sampling techniques such as the improved proposal distribution of FastSLAM 2.

I. INTRODUCTION

Simultaneous localization and mapping (SLAM) algorithms based on particle filters have gained exceptional popularity in the last few years due to their relative simplicity, computational properties, and experimental successes. However, recent work such as that by Bailey *et al.* [2] and others has shown that particle filtering SLAM algorithms are susceptible to substantial estimation inconsistencies because they generally significantly underestimate their own error. In large part this is due to degeneracies in the particle filter sampling process.

A particle filter for SLAM represents the posterior distribution of the robot’s trajectory using a set of samples, or “particles.” Conditioned on each particle is a map, estimated using a series of small extended Kalman filters for each landmark. At each time step, particles are extended according to a motion model and maps are updated based on sensor observations. The particles are weighted according to the likelihood of the observations given the sampled poses and previous observations. Finally, particles are resampled (with replacement) according to their weights in order to give more presence to highly-weighted trajectories.

Particle degeneracies occur when the weights of particles in the filter are highly skewed. In this case, the resampling step selects many copies of a few highly weighted particles. Since resampling is repeated often, the sampled poses representing past portions of the robot’s trajectory tend to become degenerate (i.e., all or

mostly all identical) so that they insufficiently encode uncertainty in the estimation. These degeneracies can have significant consequences if the robot revisits the poorly estimated region, such as when closing a loop.

In this paper we describe two new sampling strategies for particle filtering SLAM, inspired in part by the tracking literature [7], [6], which improve the consistency of the filter’s estimation and the diversity of the trajectory samples. The first approach, termed *fixed-lag roughening*, incorporates a Markov Chain Monte Carlo (MCMC) move step, perturbing pose samples over a fixed lag time according to an MCMC kernel to combat particle degeneracy. The second technique employs a *block proposal distribution* which directly samples poses over a fixed lag time from their fully joint distribution conditioned on all of the available data. The main idea behind both methods is to exploit “future” information to improve the estimates of past portions of the trajectory, in the sense that the information becomes available only *after* initial estimation of the poses.

The new sampling techniques lead to significant reductions in estimation error over previous approaches. For example, in our experiments, estimation error using fixed-lag roughening was as little as 30% that of FastSLAM 2 on average, and error using the block proposal was as little as 12%, both at the expense of a constant factor increase in computation time. Furthermore, trajectory samples from the block proposal exhibit better diversity than those from FastSLAM 2, with the filter maintaining multiple samples over most of the pose history for reasonably long trajectories.

The primary complication of both approaches is in designing and drawing from the sampling distributions — i.e., the MCMC kernel for fixed-lag roughening, and the fully joint pose distribution for the block proposal. This paper describes the main highlights, but some of the details of the distribution derivations are provided in a companion technical report [4] to conserve space.

In the next section we formally introduce particle filtering SLAM and the consistency issue, and describe previous work on improving consistency and particle diversity. We introduce fixed-lag roughening in Section III and the block proposal distribution in Section IV. Section V presents the results of experiments comparing our techniques to previous approaches.

II. PARTICLE FILTERING SLAM

The simultaneous localization and mapping (SLAM) problem is for a robot to concurrently estimate both

K. Beevers is with the Department of Computer Science, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180, U.S.A., beevek@cs.rpi.edu

W. Huang is with Applied Perception, Inc., 220 Executive Drive, Suite 400, Cranberry Township, PA 16066, U.S.A., wes@appliedperception.com

a map of the environment and the robot’s pose with respect to the map. We consider the map to consist of a set of parameterized landmarks and treat SLAM as a state estimation problem, with the goal of recovering the map $\mathbf{x}^m = [\mathbf{x}_1^m \dots \mathbf{x}_n^m]^T$ and the robot’s time-dependent trajectory $\mathbf{x}_{1:t}^r$ through the environment. At each pose the robot executes a control command (i.e., motion) according to the control input \mathbf{u}_t . It then acquires an observation (or set of observations) \mathbf{z}_t from its sensors and computes correspondence variables \mathbf{n}_t mapping observations to landmarks in the map.

The goal of SLAM is therefore to estimate the PDF:

$$p(\mathbf{x}_{1:t}^r, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \quad (1)$$

In Rao-Blackwellized particle filtering (RBPF), the posterior (1) is factored under certain independence assumptions [11] to obtain:

$$\underbrace{p(\mathbf{x}_{1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}_{\text{trajectory posterior}} \prod_{i=1}^n \underbrace{p(\mathbf{x}_i^m | \mathbf{x}_{1:t}^r, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}_{\text{landmark } i \text{ posterior}} \quad (2)$$

The posterior over trajectories is estimated nonparametrically using N samples (“particles”). Conditioned on the sampled values, landmarks are independent, and each is estimated separately, typically by a small constant-size extended Kalman filter (EKF).

The estimation of the trajectory posterior by samples is done using sequential importance sampling with resampling (SISR), or “particle filtering”. The basic idea is to employ for each particle $\phi_t^i = \{\mathbf{x}_{1:t-1}^{r,i}, \mathbf{x}^{m,i}\}$ a motion model $p(\mathbf{x}_t^{r,i} | \mathbf{x}_{t-1}^{r,i}, \mathbf{u}_t)$ as a *proposal distribution* to project the robot state forward by sampling, i.e.:

$$\mathbf{x}_t^{r,i} \sim p(\mathbf{x}_t^r | \mathbf{x}_{t-1}^{r,i}, \mathbf{u}_t) \quad (3)$$

Once every particle is projected forward, the posterior $p(\mathbf{x}_{1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})$ is represented by the particles.

Next, the samples are weighted according to the sensor measurement likelihood, i.e.:

$$\omega_t^i = \omega_{t-1}^i p(\mathbf{z}_t | \mathbf{x}_t^{r,i}, \mathbf{n}_t, \mathbf{x}^{m,i}) \quad (4)$$

Finally, the particles are *resampled* according to the weights to obtain a representation of (1).

An important issue in particle filtering SLAM is the *consistency* of the SLAM filter. A filter is inconsistent if it significantly underestimates its own error, which can lead to divergence of the filter estimate from the truth. Bailey *et al.* [2] have shown experimentally that in general, current particle filtering SLAM algorithms are inconsistent. This is in large part due to degeneracies caused by frequent resampling such that most samples become identical for much of the robot’s trajectory.

A. Related work

Several researchers have addressed consistency in the context of RBPF SLAM; we describe two well-known approaches and a third recent development.

1) *Improved proposal distribution*: For robots with very accurate sensors such as scanning laser rangefinders, the measurement likelihood in (4) is highly peaked. Thus, the proposal distribution (3) samples many robot poses that are assigned low weights, so only a few samples survive the resampling step. This can quickly lead to particle degeneracies.

An alternative is to incorporate the current sensor measurement \mathbf{z}_t into the proposal distribution, i.e.:

$$\mathbf{x}_t^{r,i} \sim p(\mathbf{x}_t^r | \mathbf{x}_{1:t-1}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \quad (5)$$

Using this approach, many more samples are drawn for robot poses that match well with the current sensor measurement and particles are more evenly weighted, so more particles are likely to survive resampling. This “improved proposal distribution” has been used in both landmark-based SLAM [11] (FastSLAM 2) and in occupancy grid scan-matching SLAM [8].

2) *Effective sample size*: The basic particle filtering algorithm resamples particles according to their weights at every iteration. It can be shown that if the weights of particles are approximately the same, resampling only *decreases* the efficiency of the sampled representation [9]. The *effective sample size* is a useful metric to determine whether resampling is necessary [10]. It can be approximated as: $\hat{N}_{\text{eff}} = 1 / \sum_{i=1}^N (\omega_t^i)^2$. If the effective sample size is large, say, $\hat{N}_{\text{eff}} > N/2$, resampling is undesirable since the PDF over robot trajectories is well represented. This technique was first applied to SLAM by Grisetti *et al.* [8].

3) *Recovering diversity through stored state*: The preceding methods focus on preventing loss of particle diversity. Another approach is to attempt to “recover” diversity. Stachniss *et al.* [12] store the “state” of the particle filter upon detecting the robot’s entry into a loop. After repeatedly traversing the loop to improve the map (a process normally resulting in loss of diversity), the filter state is restored by splicing the loop trajectory estimate onto each saved particle, effectively restoring the diversity of the filter prior to loop closing.

III. FIXED-LAG ROUGHENING

Resampling leads to degeneracies because multiple copies of the same highly-weighted particles survive. In this section we describe a modification of the particle filter sampling process that incorporates “roughening” of the sampled trajectories over a fixed lag so that the posterior over trajectories is better estimated. A similar approach termed “resample-move” has been described in the statistical literature in the context of target tracking [7], [6], and has been mentioned (but not pursued) in the context of SLAM by Bailey [1].

The basic idea is to incorporate a post-SISR Markov Chain Monte Carlo (MCMC) step to “move” the trajectory of each particle over a fixed lag time L after the usual RBPF update is complete. Specifically, for

each particle $\phi_t^i, i = 1 \dots N$, we sample $\mathbf{x}_{t-L+1:t}^{r,i} \sim q(\mathbf{x}_{t-L+1:t}^r)$, where q is an MCMC kernel with invariant distribution $p(\mathbf{x}_{t-L+1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})$. After the move, the particles are still distributed according to the desired posterior but degeneracies over the lag time L have been averted. Furthermore, some “future” information is used in drawing new values for previously sampled poses. The samples are already approximately distributed according to the desired posterior before the move, so the usual burn-in time of MCMC samplers can be avoided. The MCMC move can be repeated to obtain better samples, although in our implementation we only perform a single move at each time step.

There are two main difficulties in implementing the approach. First, an appropriate kernel q and method for sampling from it must be devised. Second, care must be taken to avoid bias from counting the same measurement twice, leading to a need for a simple mechanism to manage incremental versions of the map.

A. Fixed-lag Gibbs sampler for SLAM

An effective approach for sampling from the joint MCMC kernel $q(\mathbf{x}_{t-L+1:t}^r)$ is to employ *Gibbs sampling*, which samples each component of $\mathbf{x}_{t-L+1:t}^r$ in turn from its conditional distribution given the values of other components. At a particular lag time k , the single-component distribution is of the form:

$$\mathbf{x}_k^{r,i} \sim p(\mathbf{x}_k^r | \mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \quad (6)$$

This distribution can be manipulated to obtain:

$$\begin{aligned} & p(\mathbf{x}_k^r | \mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \propto \\ & \int p(\mathbf{z}_k | \mathbf{x}_k^{r,i}, \mathbf{n}_k, \mathbf{x}_{\mathbf{n}_k}^m) p(\mathbf{x}_{\mathbf{n}_k}^m | \mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) d\mathbf{x}_{\mathbf{n}_k}^m \\ & \times p(\mathbf{x}_k^r | \mathbf{x}_{k-1}^{r,i}, \mathbf{u}_k) p(\mathbf{x}_k^r | \mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1}) \quad (7) \end{aligned}$$

The derivation is given in [4]. We employ the usual Gaussian approximations of the terms in (7) to sample each component of the trajectory in turn (again, see [4]). The process is repeated for every particle to obtain N samples $\{\mathbf{x}_{t-L+1:t}^{r,i}\}$, and then the maps of the particles are updated conditioned on the new trajectories.

B. Incremental map management

To avoid bias the intermediate map estimate $p(\mathbf{x}_{\mathbf{n}_k}^m | \mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})$ used in (7) should incorporate all available information *except* the measurement \mathbf{z}_k from the time step being moved. Thus, we store “incremental” versions of the map over the lag time so that the intermediate map distributions can be computed.¹ To avoid storing multiple complete copies of the map of each particle, the binary tree data structure of log N FastSLAM [11] can be used to store only the differences between maps from each time step.

¹In our implementation, we store the map from time $t-L$ and the measurements $\mathbf{z}_{t-L+1:t}$. The intermediate map distributions are computed on the fly by applying EKF updates to the map using the observations from all but the k th time step.

C. Discussion

Note that (7) is nearly identical to the result of similar manipulations of the improved proposal distribution (5) as described by Montemerlo [11]. The main difference is the “backward model” $p(\mathbf{x}_k^r | \mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1})$ since we are sampling a pose in the midst of the trajectory rather than simply the most recent pose.

Note also that we do not reweight the particles after performing the MCMC roughening step. This is because the particles before the move are asymptotically drawn from the same distribution as those after the move.

IV. BLOCK PROPOSAL DISTRIBUTION

An alternative approach is to draw new samples for the last L poses directly from the joint “optimal block proposal distribution,” i.e.:

$$p(\mathbf{x}_{t-L+1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i}) \quad (8)$$

The basic idea is to sample from (8) at each time step, replacing the most recent L poses of each particle with the newly sampled ones. The result is a particle filter that is “current” in that its samples are always distributed according to the desired posterior (1) and can be used for, e.g., planning, but which yields much better samples since future information is directly exploited by the joint proposal. Thus, degeneracies in the weights of particles are much less likely to occur. A related technique was recently described by Doucet *et al.* [6] in the general particle filtering context. One can think of the standard improved proposal distribution (5) as a “1-optimal” version of the block proposal.

The main difficulty in employing the block proposal is in drawing samples from the joint distribution (8). Our approach relies on the factorization due to Chib [5]:

$$\begin{aligned} & p(\mathbf{x}_{t-L+1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i}) = \\ & p(\mathbf{x}_t^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i}) \times \\ & p(\mathbf{x}_{t-1}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i}, \mathbf{x}_t^{r,i}) \times \dots \times \\ & p(\mathbf{x}_{t-L+1}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i}, \mathbf{x}_{t-L+2:t}^{r,i}) \quad (9) \end{aligned}$$

Here, the typical term is:

$$\begin{aligned} & p(\mathbf{x}_k^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i}, \mathbf{x}_{k+1:t}^{r,i}) \\ & \propto p(\mathbf{x}_k^r | \mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}_{t-L}^{r,i}) p(\mathbf{x}_{k+1}^r | \mathbf{x}_k^{r,i}, \mathbf{u}_{k+1}) \quad (10) \end{aligned}$$

(See [4] for details.) The idea is to first *filter forward* over the robot’s trajectory by computing the distributions $\{p(\mathbf{x}_k^r | \mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}_{t-L}^{r,i})\}$ using alternating prediction/update steps (e.g., with an EKF), and then *sample backward*, first drawing $\mathbf{x}_t^{r,i} \sim p(\mathbf{x}_t^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i})$, and then sampling the poses from the preceding time steps in reverse order using the distributions that arise from substituting the sampled values into (10). This process is repeated for every particle, and the maps are updated conditioned on the sampled trajectories.

Once new samples have been drawn for $\{\mathbf{x}_{t-L+1:t}^{r,i}\}$, the particles are reweighted with the usual technique:

$$\omega_t^i = \omega_{t-1}^i \frac{\text{target distribution}}{\text{proposal distribution}} \quad (11)$$

The optimal weight update can be shown [4] to be:

$$\omega_t^i = \omega_{t-1}^i p(\mathbf{z}_t | \mathbf{x}_{1:t-L}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) \quad (12)$$

i.e., the update is proportional to the likelihood of the current measurement using the forward-filtered pose and map distribution.

A. Practical implementation

As with fixed-lag roughening, we implement the necessary models as Gaussians in practice. Forward filtering employs an EKF to compute the intermediate distributions $\{p(\mathbf{x}_k^r | \mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}_{t-L}^{r,i})\}$, and must be performed separately for each particle ϕ^i , initialized with a zero-covariance distribution centered at $\mathbf{x}_{t-L}^{r,i}$.

During forward filtering, a temporary version of the map *must be updated* with the measurements from each time step to obtain the correct forward-filtered distributions. The ideal approach is to apply the EKF to the full state vector $[\mathbf{x}^r \ \mathbf{x}^m]^T$ over the lag time. An alternative which we have used (see [4]) is to assume the landmarks are independent and apply the usual RBPF updates to the landmarks during forward filtering, inflated by the uncertainty of the intermediate pose distributions computed by the EKF.

After forward filtering, samples are drawn for each intermediate pose \mathbf{x}_k^r , for $k = t \dots t-L+1$. The first sample is drawn directly from the (approximately) optimal forward-filtered distribution. The remaining samples are conditioned on the poses drawn for succeeding time steps by applying the same backward model as for fixed-lag roughening. Again, see [4] for details.

B. Discussion

At first it may appear that the samples from the block proposal are no different from those obtained with fixed-lag roughening. In fact, while the samples are asymptotically from the same distribution (the desired posterior), those obtained from the block proposal are generally better, because poses over the lag time are drawn *directly* from the joint distribution that incorporates future information. In fixed-lag roughening, poses are originally drawn using past and present information only, then are gradually moved as future information becomes available. Only by applying many MCMC moves at each time step would the samples obtained by fixed-lag roughening be as good as those from the block proposal.

V. RESULTS

Our experiments compared FastSLAM 2, the fixed-lag roughening (FLR) algorithm from Section III, and the block proposal (BP) distribution from Section IV.

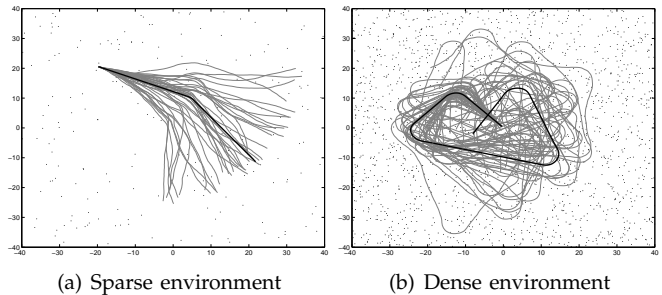


Fig. 1. Simulated environments used to test the algorithms, consisting of point landmarks placed uniformly at random. Solid dark lines are ground truth trajectories, kept the same for all simulations. Light gray lines depict typical uncorrected odometry estimates.

For the roughening and block proposal approaches we tested the algorithms with several values for the lag time L . All experiments used $N = 500$ particles and resampling was performed only when $\hat{N}_{\text{eff}} < N/2$.

Our experiments were in simulation since comparing the estimation error of the filters requires ground truth. We assumed known data associations to prevent poor correspondence-finding from influencing the comparison between filtering algorithms. Noise was introduced by perturbing odometry and range-bearing measurements. The observation model used $\sigma_r = 5$ cm and $\sigma_b = 0.3^\circ$ with a sensing radius of 10 m, and the motion model used $\sigma_x = 0.12d \cos \theta$, $\sigma_y = 0.12d \sin \theta$ and $\sigma_\theta = 0.12d + 0.24\phi$ for translation d and rotation ϕ .

Experiments were performed in a variety of simulated environments consisting of point features. We present results from two representative cases with randomly placed landmarks: a “sparse” map with a simple 27 sec. trajectory (no loops) and a “dense” map with a 63 sec. loop trajectory. The environments, ground truth trajectories, and typical raw odometry estimates are shown in Figure 1. All results were obtained by averaging 50 Monte Carlo trials of each simulation.

A. NEES comparison

We begin by comparing the normalized estimation error squared (NEES) [3], [2] of the trajectory estimates produced by each algorithm. The NEES is a useful measure of filter consistency since it estimates the *statistical* distance of the filter estimate from the ground truth, i.e., it takes into account the filter’s estimate of its own error. For a ground truth pose \mathbf{x}_t^r and an estimate $\hat{\mathbf{x}}_t^r$ with covariance $\hat{\mathbf{P}}_t^r$ (computed from the weighted particles assuming they are approximately Gaussian), the NEES is $(\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)^T (\hat{\mathbf{P}}_t^r)^{-1} (\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)$. The recent paper by Bailey *et al.* [2] gives more details about using NEES to measure RBPF SLAM consistency.

We computed the NEES at each time step using the current particle set. Figure 2 shows the resulting errors from each of the algorithms. (To conserve space, we present reduced versions of the plots here. Larger

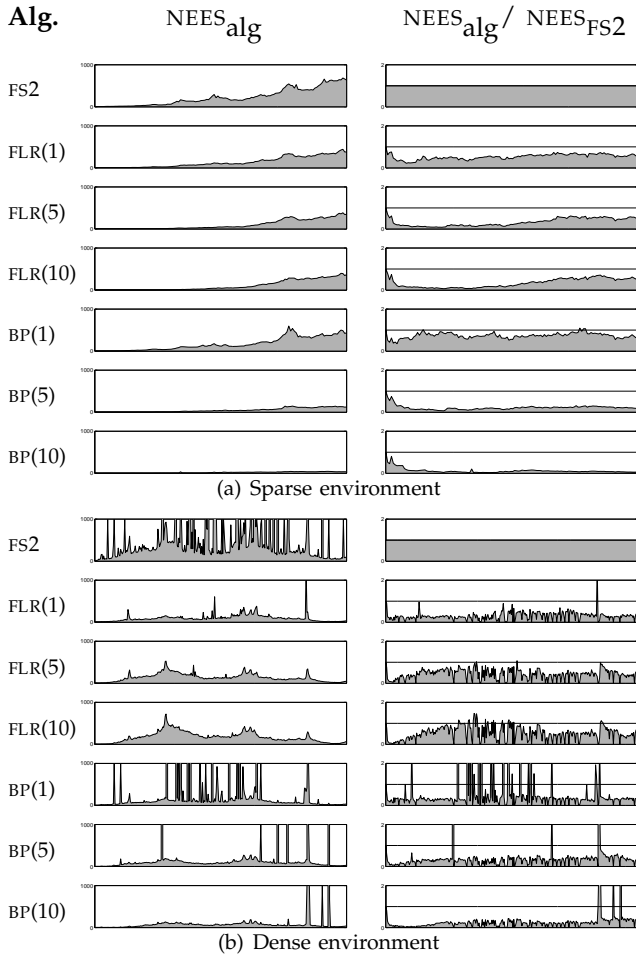


Fig. 2. NEES (left column) and the ratio of NEES to that of FastSLAM 2 (right column) for each of the algorithms, versus the simulation time on the x -axis. We use the abbreviations FS2, FLR(L), and BP(L) to indicate FastSLAM 2, fixed-lag roughening with lag L , and block proposal with lag L respectively.

versions can be seen in [4].) In the sparse environment, NEES grows steadily for FS2 and FLR, and for BP with small lag times. Increasing the lag time for FLR has relatively little effect on NEES because “future” information is exploited slowly (see Section IV-B). FLR’s NEES is approximately 33% that of FS2 on average for $L = 5$ and $L = 10$. On the other hand, increasing the lag time for BP dramatically reduces NEES. The NEES of BP(1) is roughly 73% that of FS2 on average; for BP(5), 22%; and for BP(10), 12%. For the dense case the results are similar. Note that FLR avoids degeneracies (manifested as spikes in the NEES plots) by moving particles after resampling. Interestingly, increasing L in a dense environment appears to slightly *increase* the NEES of FLR, a subject warranting further investigation.

Note that the range of the NEES plots is quite large — none of the filters is truly consistent. (A consistent filter over 50 Monte Carlo trials should have NEES less than 3.72 with 95% probability [2].) While the estimation error using fixed-lag roughening and the

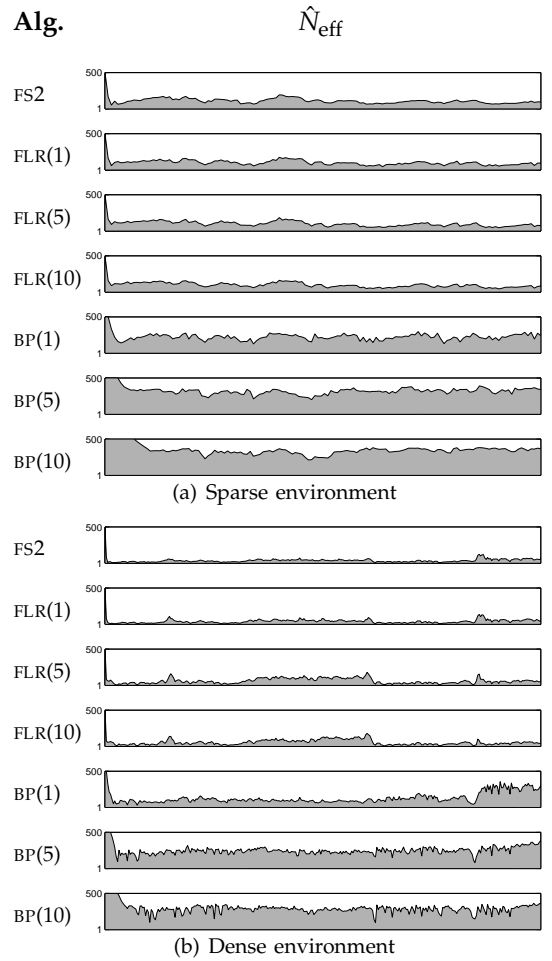


Fig. 3. \hat{N}_{eff} for each of the algorithms, versus the simulation time on the x -axis.

block proposal is significantly reduced, these strategies alone do not guarantee a consistent filter, at least with reasonably small lag times. In fact it is likely that guaranteeing consistent SLAM estimation (with high probability) while representing the trajectory posterior by samples requires drawing the full dimensionality of the samples from a distribution conditioned on all the measurements, e.g., with MCMC, since particle filtering is always susceptible to resampling degeneracies depending on the environment and trajectory.

B. \hat{N}_{eff} comparison

The effective sample size \hat{N}_{eff} is also a useful statistic in examining filter consistency. If \hat{N}_{eff} is high, the weights of particles are relatively unskewed, i.e., all particles are contributing to the estimate of the trajectory posterior. Furthermore, since \hat{N}_{eff} dictates when resampling occurs, high values of \hat{N}_{eff} indicate less chance for degeneracies in past portions of the trajectory estimate because resampling occurs infrequently.

Figure 3 shows \hat{N}_{eff} as computed at each time step in the simulations. In the sparse case, FLR exhibits no

significant improvement over FS2, but in the dense environment FLR(5) and FLR(10) have about 72% higher \hat{N}_{eff} than FS2 on average. Again, BP exhibits stronger results, with BP(10) more than 1000% better than FS2 in the dense case, and 340% better in the sparse case. This can be attributed to direct use of future information by the block proposal, which leads to better samples for essentially the same reason FastSLAM 2's samples are better than those of FastSLAM 1.

C. Particle diversity

Finally, we examine particle diversity for each of the filters. Figure 4 shows the variance of the pose histories of all the particles, computed at the end of SLAM, along with the number of unique particles representing each pose. For all of the algorithms, the end of the trajectory is better represented than earlier portions. FLR extends the representation over the lag time but the typical quick dropoff remains. BP avoids the loss of diversity in the sparse case, maintaining non-zero variance over most of the trajectory, as one would expect since little resampling occurs due to the high effective sample size. In a denser environment a significant amount of resampling still occurs, reducing the benefit somewhat.

VI. CONCLUSIONS

We have described two new sampling strategies for particle filtering SLAM. The first method, fixed-lag roughening, applies an MCMC move to the trajectory samples over a fixed lag at each time step. The second approach, the block proposal distribution, draws new samples for all poses in a fixed-lag portion of the trajectory from their joint distribution. Both techniques exploit “future” information to improve the estimation of past poses. We have given overviews of the technical details behind each approach; full details are in [4].

Our results show that the new algorithms lead to substantial improvements in SLAM estimation. Fixed-lag roughening and the block proposal yield samples with lower statistical estimation error than those of FastSLAM 2. Furthermore, samples drawn from the block proposal tend to have much more uniform importance weights, leading to less need for resampling and consequently, improved particle diversity.

REFERENCES

- [1] T. Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, Australian Center for Field Robotics, University of Sydney, August 2002.
- [2] T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *IEEE Intl. Conf. on Robotics and Automation*, pages 424–427, 2006.
- [3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation*. Wiley, New York, 2001.
- [4] K. Beevers. Sampling strategies for particle filtering SLAM. Technical Report 06-11, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 2006.
- [5] S. Chib. Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, 75(1):79–97, 1996.

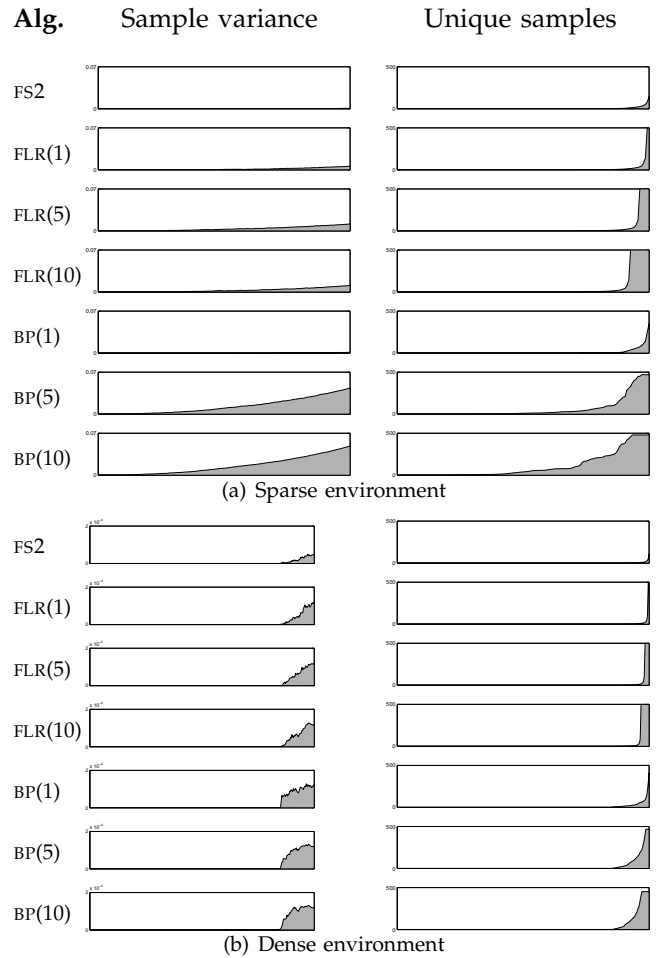


Fig. 4. Comparison of particle diversity. The plots shown here are computed using the trajectory samples at the end of SLAM, i.e., $\{\mathbf{x}_{1:t}^i, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}\}$. On the left is the sample variance for each pose in the trajectory, computed as $\text{trace}(\hat{\mathbf{P}}_t^i)$. On the right is the number of unique samples representing each pose.

- [6] A. Doucet, M. Briers, and S. Sénécal. Efficient block sampling strategies for sequential Monte Carlo methods. *Journal of Computational and Graphical Statistics*, to appear, 2006.
- [7] W. Gilks and C. Berzuini. Following a moving target — Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society B*, 63(1):127–146, 2001.
- [8] G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 2443–2448, 2005.
- [9] J. Liu. *Monte Carlo strategies in scientific computing*. Springer, New York, 2001.
- [10] J. Liu and R. Chen. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576, June 1995.
- [11] M. Montemerlo. *FastSLAM: a factored solution to the simultaneous localization and mapping problem with unknown data association*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2003.
- [12] C. Stachniss, G. Grisetti, and W. Burgard. Recovering particle diversity in a Rao-Blackwellized particle filter for SLAM after actively closing loops. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 667–672, 2005.