Sampling Strategies for Particle Filtering SLAM

Kristopher R. Beevers
Department of Computer Science
Rensselaer Polytechnic Institute
beevek@cs.rpi.edu

Abstract

We describe several new sampling strategies for Rao-Blackwellized particle filtering SLAM. Two of the strategies, called *fixed-lag roughening* and the *block proposal distribution*, attempt to 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 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.

In addition, we examine the effectiveness of two new resampling techniques, *residual resampling* and *generalized resampling*, as applied to RBPF SLAM. These drop-in-place techniques are simple to use and (in the case of residual resampling) computationally cheaper than the standard random resampling approach. However, our results show that they offer no real improvement in performance over random resampling in SLAM.

This is an extended version of a paper (Beevers and Huang, 2007) previously submitted for publication.

1 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. (2006) 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 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 (in the basic particle filter, at every time step), 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.

Some previous work has addressed the issue of particle diversity. The improved proposal distribution of FastSLAM 2 (Montemerlo, 2003) seeks to sample "better" poses so that particle weights remain relatively uniform. The adaptive resampling technique employed by Grisetti et al. (2005) performs resampling only when the weights become skewed, rather than at every time step. (We further discuss these and other approaches in Section 2.)

In this paper we describe two new sampling strategies for particle filtering SLAM, inspired in part by the tracking literature (Gilks and Berzuini, 2001; Doucet et al., 2006), 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. We derive conditional distributions for a Gibbs sampler for the roughening case, and employ a well-known algorithm to simulate from joint distributions in state-space models for the block proposal. In addition, we show how to apply the techniques in practice using the standard Gaussian approximations to the motion and measurement models of the robot.

Other work in the statistics literature has pinpointed several alternatives to the usual random resampling strategy employed in particle filtering techniques (Liu, 2001). For purposes of comparison, we have implemented two of the approaches, residual resampling and generalized resampling, in the context of RBPF SLAM. Residual resampling deterministically selects a number of copies of each particle proportional to the weights. A benefit of this approach is that it requires access to many fewer random numbers than the standard approach. Generalized resampling draws particles with probability proportional to a *function* of the weights. The function can be tailored to balance the need for particle diversity with the need

for sample effectiveness. Unfortunately our results show that neither technique yields much improvement over random resampling in terms of filter performance.

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 3 and the block proposal distribution in Section 4. The two resampling strategies are discussed in Section 5. Finally, Section 6 presents the results of experiments comparing our techniques to previous approaches.

2 Particle filtering SLAM

The simultaneous localization and mapping (SLAM) problem is for a robot to concurrently estimate both 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 geometric objects (e.g., points or lines) and treat SLAM as a state estimation problem, with the goal of recovering the map $\mathbf{x}^m = [\mathbf{x}_1^m \ \ldots \ \mathbf{x}_n^m]^T$ and the robot's time-dependent trajectory $\mathbf{x}_{1:t}^r$ through the environment, where each intermediate pose $\mathbf{x}_t^r$ typically consists of the robot's position and orientation with respect to the world frame. 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 distribution:

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

(While correspondences $\mathbf{n}_{1:t}$ could be included in the inference process they are typically estimated in more ad hoc fashion, e.g., by taking the maximum likelihood correspondences.)

A common SLAM assumption is that landmarks are randomly and independently distributed in the environment. Under this assumption, correlation between landmark variables $\{\mathbf{x}_i^m\}$ in the SLAM estimation arises only through uncertainty in the robot's trajectory.[1] This motivates a "Rao-Blackwellized particle filtering" (RBPF) approach to estimating the posterior (1). The posterior is first factored according to the stated independence to obtain:

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

The posterior over trajectories is estimated nonparametrically using N samples ("particles"). Conditioned on the sampled values, landmark variables 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), commonly known as bootstrap filtering or particle filtering. (Particle filtering is a general method for sampling from high-dimensional, complicated distributions by building samples sequentially.) The basic idea is to employ for each particle $\phi_t^i = \left\{\mathbf{x}_{1:t-1}^{r,i}, \mathbf{x}^{m,i}\right\}$ a motion

[1] In (Beevers and Huang, 2006) we develop a particle filtering SLAM algorithm that does not rely on this common assumption, but in this work we'll stick with the usual model.

model $p(\mathbf{x}_t^r|\mathbf{x}_{t-1}^{r,i}, \mathbf{u}_t)$ as a *proposal distribution* to project the robot state forward by sampling, i.e.:

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

Once every particle is projected forward, the "prediction" step is complete and the posterior:

$$p(\mathbf{x}_{1:t}^r|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1}) \tag{4}$$

is represented by the particles.

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

$$\omega_t^i = \omega_{t-1}^i p(\mathbf{z}_t|\mathbf{x}_t^{r,i}, \mathbf{n}_t, \mathbf{x}_{\mathbf{n}_t}^{m,i}) \tag{5}$$

The weighted samples are (asymptotically) from the desired posterior (1).

Finally, a *resampling* step draws N times from the particle set, with replacement, according to the weights $\{\omega_t^i\}$. The weights are then reset uniformly. This typically improves the representation of the trajectory posterior because over time, most of the weight is concentrated in just a few particles. Resampling assigns more particles to areas of high likelihood which then leads to better estimation of future portions of the trajectory.

For more details about general particle filtering, see, e.g., (Liu, 2001). A detailed description of particle filtering in the context of SLAM can be found in (Montemerlo, 2003).

2.1 Consistency

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. (2006) 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.

The primary objective of this work is to develop particle filtering SLAM algorithms with consistency properties that are significantly more desirable than those of the standard approaches, without sacrificing the computational benefits of particle filtering techniques.

2.2 Related work

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

2.2.1 Improved proposal distribution

For robots with very accurate sensors such as scanning laser rangefinders, the measurement likelihood in (5) 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., to sample robot poses according to:

$$p(\mathbf{x}_t^r|\mathbf{x}_{1:t-1}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{6}$$

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 (Montemerlo, 2003), where it is often called "FastSLAM 2," and in occupancy grid scan-matching SLAM (Grisetti et al., 2005).

2.2.2 Effective sample size

The basic particle filtering algorithm resamples particles according to their weights at every iteration, i.e., for every SLAM update step. It can be shown that if the weights of particles are approximately the same, resampling only *decreases* the efficiency of the sampled representation (Liu, 2001). The *effective sample size* is a useful metric to determine whether resampling is necessary (Liu and Chen, 1995). It can be approximated as:

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\omega_t^i\right)^2} \tag{7}$$

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. (2005).

2.2.3 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. (2005) 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.

3 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 (Gilks and Berzuini, 2001; Doucet et al., 2006), and has been mentioned (but not pursued) in the context of SLAM by Bailey (2002).

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 $\phi_t^i, i = 1 \ldots N$, we sample:

$$\mathbf{x}_{t-L+1:t}^{r,i} \sim q(\mathbf{x}_{t-L+1:t}^{r}) \tag{8}$$

where q is an MCMC kernel with invariant distribution

$$p(\mathbf{x}_{t-L+1:t}^{r} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{9}$$

After the move, the particles are still approximately 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.

3.1 Fixed-lag Gibbs sampler for SLAM

An effective approach for sampling from the joint MCMC kernel $q(\mathbf{x}^r_{t-L+1:t})$ is to employ *Gibbs sampling*, which samples each component of $\mathbf{x}^r_{t-L+1:t}$ in turn from its conditional distribution given the values of other components. Specifically, we sample each component in turn according to the following scheme, for every particle ϕ^i_t:

$$\begin{aligned}
\mathbf{x}^{r,i}_{t-L+1} &\sim p(\mathbf{x}^r_{t-L+1}|\mathbf{x}^{r,i}_{1:t-L,t-L+2:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \\
&\cdots \\
\mathbf{x}^{r,i}_{k} &\sim p(\mathbf{x}^r_{k}|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \\
&\cdots \\
\mathbf{x}^{r,i}_{t} &\sim p(\mathbf{x}^r_{t}|\mathbf{x}^{r,i}_{1:t-1}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \qquad (10)
\end{aligned}$$

The last distribution (10) is equivalent to the usual (improved) proposal distribution. The other intermediate distributions are of the form:

$$\mathbf{x}^{r,i}_{k} \sim p(\mathbf{x}^r_{k}|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \qquad (11)$$

at a particular lag time k. We will concentrate on manipulating (11) into a form from which we can easily sample. We can first rewrite this distribution using Bayes' rule as follows:

$$\begin{aligned}
&p(\mathbf{x}^r_{k}|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \\
&\quad = \frac{p(\mathbf{z}_k|\mathbf{x}^{r,i}_{1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) p(\mathbf{x}^r_{k}|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})}{p(\mathbf{z}_k|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})} \qquad (12)
\end{aligned}$$

The denominator can be subsumed into a normalization constant η:

$$= \eta p(\mathbf{z}_k|\mathbf{x}^{r,i}_{1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) p(\mathbf{x}^r_{k}|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) \qquad (13)$$

Applying the Markov assumption and factoring, we obtain:

$$= \eta p(\mathbf{z}_k|\mathbf{x}^{r,i}_{1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) \underbrace{p(\mathbf{x}^r_{k}|\mathbf{x}^{r,i}_{k-1}, \mathbf{u}_{k})}_{\text{forward model}} \underbrace{p(\mathbf{x}^r_{k}|\mathbf{x}^{r,i}_{k+1}, \mathbf{u}_{k+1})}_{\text{backward model}} \qquad (14)$$

We now have three terms. The forward model is the distribution of robot poses given the previous pose and the current control input; the backward model is the distribution of poses given the next pose and control input. Finally, marginalizing the first term yields:

$$p(\mathbf{z}_k|\mathbf{x}^{r,i}_{1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) = \int \underbrace{p(\mathbf{z}_k|\mathbf{x}^{r,i}_k, \mathbf{n}_k, \mathbf{x}^{m,i}_{\mathbf{n}_k})}_{\text{measurement likelihood}} \underbrace{p(\mathbf{x}^{m,i}_{\mathbf{n}_k}|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})}_{\text{landmark distribution}} d\mathbf{x}^{m,i}_{\mathbf{n}_k} \quad (15)$$

Substituting this back into (14) we obtain:

$$p(\mathbf{x}^r_k|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) = \eta \int p(\mathbf{z}_k|\mathbf{x}^{r,i}_k, \mathbf{n}_k, \mathbf{x}^m_{\mathbf{n}_k}) p(\mathbf{x}^m_{\mathbf{n}_k}|\mathbf{x}^{r,i}_{1:k-1,k+1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})\, d\mathbf{x}^m_{\mathbf{n}_k} \; p(\mathbf{x}^r_k|\mathbf{x}^{r,i}_{k-1}, \mathbf{u}_k) p(\mathbf{x}^r_k|\mathbf{x}^{r,i}_{k+1}, \mathbf{u}_{k+1}) \quad (16)$$

3.2 Practical implementation

In practice we approximate the measurement likelihood, the landmark distributions, and the forward and backward models by Gaussians. Replacing the terms of (16) accordingly, we obtain the convolution of two Gaussians multiplied by two more Gaussians, i.e.:

$$\int \mathcal{N}\Big(g(\mathbf{x}^{m,i}_{\mathbf{n}_k}, \mathbf{x}^{r,i}_k), \mathbf{R}_k\Big)\, \mathcal{N}\Big(\mathbf{x}^{m,i}_{\mathbf{n}_k}, \mathbf{P}^{m,i}_{\mathbf{n}_k}\Big)\, d\mathbf{x}^{m,i}_{\mathbf{n}_k} \times \mathcal{N}\Big(h(\mathbf{x}^{r,i}_{k-1}, \mathbf{u}_k), \mathbf{V}_k\Big)\, \mathcal{N}\Big(s(\mathbf{x}^{r,i}_{k+1}, \mathbf{u}_{k+1}), \mathbf{V}_{k+1}\Big) \quad (17)$$

where the functions g, h, and s represent the measurement model, forward motion model, and backward motion model, respectively.

Typically, the measurement model must be linearized, e.g., by taking the first-order Taylor expansion:

$$g(\mathbf{x}^m_{\mathbf{n}_k}, \mathbf{x}^r_k) \approx g(\mathbf{x}^{m,i}_{\mathbf{n}_k}, h(\mathbf{x}^{r,i}_{k-1}, \mathbf{u}_k)) + \mathbf{H}_m(\mathbf{x}^m_{\mathbf{n}_k} - \mathbf{x}^{m,i}_{\mathbf{n}_k}) + \mathbf{H}_r(\mathbf{x}^r_k - \mathbf{x}^{r,i}_k) \quad (18)$$

where $\mathbf{H}_m$ and $\mathbf{H}_r$ are the derivatives of g with respect to the observed landmark and the robot's pose, respectively, evaluated at the expected values.

Note that aside from the backward motion model term, the distribution (17) is exactly the approximation of the improved proposal described by Montemerlo (2003), who has shown that the resulting distribution is Gaussian with mean $\tilde{\mu}^{r,i}_k$ and covariance $\tilde{\mathbf{P}}^{r,i}_k$ as follows:

$$\tilde{\mu}^{r,i}_k = \tilde{\mathbf{P}}^{r,i}_k \mathbf{H}^T_r \mathbf{S}^{-1}_k (\mathbf{z}_k - g(\mathbf{x}^{m,i}_{\mathbf{n}_k}, h(\mathbf{x}^{r,i}_{k-1}, \mathbf{u}_k))) + h(\mathbf{x}^{r,i}_{k-1}, \mathbf{u}_k) \quad (19)$$

$$\tilde{\mathbf{P}}^{r,i}_k = \Big(\mathbf{H}^T_r \mathbf{S}^{-1}_k \mathbf{H}_r + \mathbf{H}^T_u \mathbf{V}^{-1}_k \mathbf{H}_u\Big)^{-1} \quad (20)$$

where:

$$\mathbf{S}_k = \mathbf{R}_k + \mathbf{H}_m \mathbf{P}^{m,i}_{\mathbf{n}_k} \mathbf{H}^T_m \quad (21)$$

It remains to incorporate the backward motion model, which is straightforward since we can just merge the backward model distribution with the distribution (19-20), i.e.:

$$\mu_k^{r,i} = \tilde{\mu}_k^{r,i} + \tilde{\mathbf{P}}_k^{r,i}(\tilde{\mathbf{P}}_k^{r,i} + \mathbf{H}_u \mathbf{V}_{k+1} \mathbf{H}_u^T)^{-1}(s(\mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1}) - \tilde{\mu}_k^{r,i}) \quad (22)$$

$$\mathbf{P}_k^{r,i} = \left((\tilde{\mathbf{P}}_k^{r,i})^{-1} + \mathbf{H}_u^T \mathbf{V}_{k+1}^{-1} \mathbf{H}_u\right)^{-1} \quad (23)$$

The fixed-lag roughening algorithm computes the distribution $\mathcal{N}\left(\mu_k^{r,i}, \mathbf{P}_k^{r,i}\right)$ for each time step $k = t - L + 1 \ldots t$ for every particle ϕ_t^i to draw the new poses $\mathbf{x}_{t-L+1:t}^{r,i}$, and then updates the maps conditioned on the new trajectories.

3.3 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 (16) should incorporate all available information *except* the measurement $\mathbf{z}_k$ from the time step being moved. Thus, it is necessary to store "incremental" versions of the map over the lag time so that the intermediate map distributions can be computed. A simple strategy which we use in our implementation is to 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 kth time step.

To avoid storing multiple complete copies of the map of each particle, the binary tree data structure of log N FastSLAM (Montemerlo, 2003) can be used to store only the differences between maps from each time step.[2]

3.4 Discussion

Note that (16) is nearly identical to the result of similar manipulations of the improved proposal distribution (6) as described by Montemerlo (2003). The primary difference is the inclusion of 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.

4 Block proposal distribution

An alternative approach for exploiting "future" information in drawing trajectory samples over a fixed lag time L is to draw new samples for the last L poses directly from the joint "L-optimal block proposal distribution," i.e.:

$$p(\mathbf{x}_{t-L+1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}_{t-L}^{r,i}) \quad (24)$$

The basic idea is to sample from (24) at each time step, replacing the most recent L poses of each particle with the newly sampled ones. The result is a particle filter

[2] We note that while the tree described in (Montemerlo, 2003) stores landmarks only at the leaf nodes and thus requires $O(n \log n)$ memory for n landmarks, it is possible to store landmarks at every node in the tree, which requires $O(n)$ memory.

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. (2006) in the general particle filtering context. One can think of the standard improved proposal distribution (6) 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 (24). Our approach relies on the factorization due to Chib (1996):

$$\begin{aligned} p(\mathbf{x}^r_{t-L+1:t}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) &= p(\mathbf{x}^r_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) \times \\ p(\mathbf{x}^r_{t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_t) &\times \cdots \times p(\mathbf{x}^r_k|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{k+1:t}) \times \cdots \times \\ & p(\mathbf{x}^r_{t-L+1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{t-L+2:t}) \quad (25) \end{aligned}$$

Here, the typical term is:

$$p(\mathbf{x}^r_k|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{k+1:t}) \quad (26)$$

and application of Bayes' rule and the Markov assumption leads to (Chib, 1996):

$$\begin{aligned} & p(\mathbf{x}^r_k|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{k+1:t}) \\ & \quad \propto p(\mathbf{x}^r_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\ p(\mathbf{x}^r_{k+1:t}, \mathbf{z}_{k+1:t}|\mathbf{x}^{r,i}_k, \mathbf{x}^{r,i}_{t-L}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k}, \mathbf{n}_{1:t}) \quad (27) \\ & \quad = p(\mathbf{x}^r_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\ p(\mathbf{x}^r_{k+1}|\mathbf{x}^{r,i}_k, \mathbf{u}_{k+1}) \\ & \qquad \times p(\mathbf{x}^r_{k+2:t}, \mathbf{z}_{k+1:t}|\mathbf{x}^{r,i}_k, \mathbf{x}^{r,i}_{k+1}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) \quad (28) \\ & \quad \propto p(\mathbf{x}^r_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\ p(\mathbf{x}^r_{k+1}|\mathbf{x}^{r,i}_k, \mathbf{u}_{k+1}) \quad (29) \end{aligned}$$

where the final step follows because $p(\mathbf{x}^r_{k+2:t}, \mathbf{z}_{k+1:t}|\mathbf{x}^{r,i}_k, \mathbf{x}^{r,i}_{k+1}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L})$ is independent of $\mathbf{x}^r_k$.

The idea is to first *filter forward* over the robot's trajectory by computing the distributions $\{p(\mathbf{x}^r_k|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\}$ using alternating prediction and update steps (e.g., with an EKF), and then *sample backward*, first drawing:

$$\mathbf{x}^{r,i}_t \sim p(\mathbf{x}^r_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) \quad (30)$$

and then sampling the poses from the preceding time steps in reverse order using the distributions that arise from substituting the sampled values into (29). This process is repeated for every particle, and the corresponding maps are updated conditioned on the sampled trajectories.

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

$$\omega^i_t = \omega^i_{t-1} \frac{\text{target distribution}}{\text{proposal distribution}} \quad (31)$$

The optimal weight update is given by:

$$\omega_t^i = \omega_{t-1}^i \frac{p(\mathbf{x}_{1:t-L}^r|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}{p(\mathbf{x}_{1:t-L}^r|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})} \tag{32}$$

$$= \omega_{t-1}^i \frac{p(\mathbf{z}_t|\mathbf{x}_{1:t-L}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t})\ p(\mathbf{x}_{1:t-L}^r|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})}{p(\mathbf{x}_{1:t-L}^r|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})\ p(\mathbf{z}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t})} \tag{33}$$

$$\propto \omega_{t-1}^i\, p(\mathbf{z}_t|\mathbf{x}_{1:t-L}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) \tag{34}$$

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

4.1 Practical implementation

As with fixed-lag roughening, we implement the necessary models as Gaussians in practice. The algorithm for sampling from (24) proceeds in two steps: forward filtering and backward sampling.

4.1.1 Forward filtering

The forward filtering step estimates the intermediate pose distributions conditioned upon past and present data, but not on future data. Note that the distributions must be computed separately for each particle since they are also conditioned on the "starting pose" $\mathbf{x}_{t-L}^{r,i}$. We implement the step using an extended Kalman filter (EKF) with the usual alternation between predictions and updates, for $k = t - L + 1 \ldots t$. We first compute the prediction:

$$\tilde{\mu}_k^{r,i} = h(\tilde{\mu}_{k-1}, \mathbf{u}_k) \tag{35}$$

$$\tilde{\mathbf{P}}_k^{r,i} = \tilde{\mathbf{P}}_{k-1}^{r,i} + \mathbf{H}_u \mathbf{V}_k \mathbf{H}_u^T \tag{36}$$

and then apply the measurement(s) to improve the model using (19-20). The resulting distribution is used recursively to compute the distribution for the next time step. The process is initialized with $\tilde{\mathbf{P}}_{t-L}^{r,i} = \mathbf{0}$ since the pose $\mathbf{x}_{t-L}^{r,i}$ is already sampled.

Note that during the forward filtering step, 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. If the number of observed landmarks in a time step is less than a constant m (rather than a function of the size of the map), then the total cost of forward filtering is $O(NLm^3)$, i.e., asymptotically constant time to draw N samples at each time step.

An alternative approach 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, i.e.:

$$\mathbf{x}_{\mathbf{n}_k}^{m,i} = \mathbf{x}_{\mathbf{n}_k}^{m,i} + (\mathbf{P}_{\mathbf{n}_k}^{m,i} \mathbf{H}_m^T \mathbf{S}^{-1})(\mathbf{z}_k - g(\mathbf{x}_{\mathbf{n}_k}^{m,i}, \tilde{\mu}_k)) \tag{37}$$

$$\mathbf{P}_{\mathbf{n}_k}^{m,i} = \left((\mathbf{P}_{\mathbf{n}_k}^{m,i})^{-1} + \mathbf{H}_m^T \mathbf{R}_k^{-1} \mathbf{H}_m + \mathbf{H}_r^T (\tilde{\mathbf{P}}_k^{r,i})^{-1} \mathbf{H}_r\right)^{-1} \tag{38}$$

4.1.2 Backward sampling

Once the forward filtering step is complete, we draw samples for each intermediate pose $\mathbf{x}_k^r$, starting with $k = t$ and decrementing until $k = t - L + 1$. The first sample is drawn directly from the (approximately) optimal forward-filtered distribution:

$$\mathbf{x}_t^{r,i} \sim \mathcal{N}\left(\tilde{\mu}_t^{r,i}, \tilde{\mathbf{P}}_t^{r,i}\right) \tag{39}$$

The remaining samples are conditioned on the poses drawn for the succeeding time steps by applying the same backward model used in fixed-lag roughening, i.e., we apply the steps (22-23) with $\tilde{\mu}_k^{r,i}$ and $\tilde{\mathbf{P}}_k^{r,i}$ as computed by during forward filtering, and draw samples from the resulting Gaussians.

4.1.3 Reweighting

After drawing the samples $\{\mathbf{x}_{t-L+1:t}^{r,i}\}$, the particles must be reweighted to approximate the desired posterior. From (34), it can easily be seen that the appropriate weight update is:

$$\omega_t^i = \omega_{t-1}^i \times |2\pi \mathbf{L}_t^i|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{z}_t - g(\mathbf{x}_{\mathbf{n}_t}^{m,i}, \tilde{\mu}_t^{r,i}))^T (\mathbf{L}_t^i)^{-1} (\mathbf{z}_t - g(\mathbf{x}_{\mathbf{n}_t}^{m,i}, \tilde{\mu}_t^{r,i}))\right) \tag{40}$$

with:

$$\mathbf{L}_t^i = \mathbf{H}_r \tilde{\mathbf{P}}_t^{r,i} \mathbf{H}_r^T + \mathbf{H}_m \mathbf{P}_{\mathbf{n}_t}^{m,i} \mathbf{H}_m^T + \mathbf{R}_t \tag{41}$$

where $\mathbf{x}_{\mathbf{n}_t}^{m,i}$ and $\mathbf{P}_{\mathbf{n}_t}^{m,i}$ are as computed during the forward filtering step.

4.2 Discussion

At first it may appear that the samples drawn using the block proposal distribution 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 will generally be better. This is because poses over the lag time are drawn *directly* from the joint distribution that incorporates future information. On the other hand, in fixed-lag roughening, poses are originally drawn using past and present information only, and then are gradually moved as future information becomes available. Only through the application of many MCMC moves at each time step will the samples obtained by fixed-lag roughening be as good as those from the block proposal.

5 Resampling strategies

In addition to fixed-lag roughening and the block proposal distribution described above, we have examined the use of alternative resampling methods in RBPF SLAM. To our knowledge, all published RBPF SLAM algorithms employ the random resampling approach, which resamples particles with probability proportional to their importance weights. We briefly describe two alternative techniques from the statistical literature (Liu, 2001), termed *residual resampling* and *generalized resampling*, and apply them to particle filtering SLAM. Our results indicate that the performance of these strategies in the context of SLAM is not appreciably better than random resampling. However, they offer flexibility and, in the case of residual resampling, may be computationally beneficial.

5.1 Residual resampling

Residual sampling is a mostly-deterministic approach that enforces the number of copies of a particle retained during resampling to be (approximately) proportional to the weight of the sample. (Note that this is the expected result of random resampling.) The technique is shown in Algorithm 1.

Algorithm 1 Residual resampling

1: Let $\tilde{\omega}_t^j = \omega_t^j / \sum_{i=1}^N \omega_t^i$
2: Retain $k_j = \lfloor N\tilde{\omega}_t^j \rfloor$ copies of ϕ_t^j
3: Let $N_r = N - k_1 - \ldots - k_N$
4: Obtain N_r i.i.d. draws (with replacement) from $\{\phi_t^1, \ldots, \phi_t^N\}$ w.p. proportional to $N\tilde{\omega}_t^j - k_j$
5: $\forall j,\ \omega_t^j = 1/N$

Since the number of deterministically selected copies of all particles, $\sum_{j=1}^N k_j$, may be less than N, random resampling is performed according to the residuals $N\tilde{\omega}_t^j - k_j$ in Step 4 to prevent bias.

According to Liu (2001), residual resampling can be shown to "dominate" random resampling in that it yields more accurate PDF approximations and is comparable or better in terms of computation. Perhaps the primary benefit is that residual resampling gives comparable performance to random resampling while consuming many fewer random numbers, which may be costly to generate, particularly in embedded scenarios.

5.2 Generalized resampling

The idea of generalized resampling is to resample according to alternative probabilities $\{a_t^i\}$ instead of the usual importance weights $\{\omega_t^i\}$. The intuition behind this approach, which is depicted in Algorithm 2, is that $\{a_t^i\}$ can be used to "modify" the weights of particles, balancing focus (giving more presence to particles with high weights) with particle diversity.

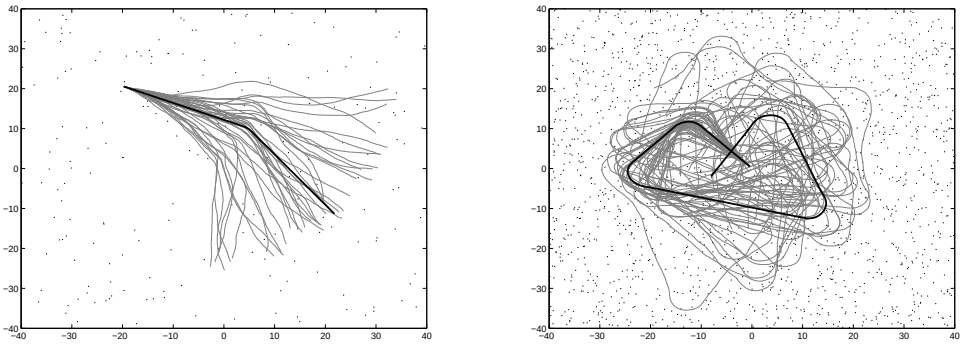
Algorithm 2 Generalized resampling

1: **for** $j = 1 \ldots N$ **do**
2: Draw k from $\{1, \ldots, N\}$ according to a_t^i, $i = 1 \ldots N$
3: Let $\tilde{\phi}_t^j = \phi_t^k$
4: Let $\tilde{\omega}_t^j = \omega_t^k / a_t^k$
5: **return** the $\tilde{\phi}_t^i$s and $\tilde{\omega}_t^i$s

Liu (2001) suggests assigning generalized weights according to:

$$a_t^i = \left(\omega_t^i\right)^\alpha \tag{42}$$

with $0 < \alpha \leq 1$. By picking $\alpha < 1$, the weight of seemingly poor particles is slightly amplified, giving them a "second chance." (Note that the a_t^is should be monotone in ω_t^i since we generally want to discard bad samples and duplicate good ones.) The weights are reset nonuniformly after resampling to prevent bias.



(a) Sparse environment (b) Dense environment

Figure 1: Simulated environments used to test RBPF SLAM algorithms. The environments consist of point landmarks placed uniformly at random. The solid dark lines represent the ground truth trajectories of the robot, which were kept the same for all simulations. The lighter gray lines depict several typical uncorrected odometry estimates of the robot's trajectory.

6 Results

Our experiments compared the standard FastSLAM 2 algorithm, the fixed-lag roughening (FLR) algorithm from Section 3, the block proposal (BP) distribution from Section 4, and FastSLAM 2 with residual (RES) and generalized (GEN) resampling as described in Section 5. For the roughening and block proposal approaches we tested the algorithms with several values for the lag time L, and generalized resampling was tested with several values of the parameter α. 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.

6.1 NEES comparison

We begin by comparing the normalized estimation error squared (NEES) (Bar-Shalom et al., 2001; Bailey et al., 2006) 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:

$$(\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)(\hat{\mathbf{P}}_t^r)^{-1}(\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)^T \tag{43}$$

The recent paper by Bailey et al. (2006) gives more details about using NEES to measure RBPF SLAM consistency.

We computed the NEES at each time step using the current particle set. Figures 2-3 and 4-5 show the resulting errors from each of the algorithms in the sparse and dense examples, respectively. In the sparse environment, NEES grows steadily for FS2, FLR, RES, GEN, 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 4.2). 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%. RES and GEN fail to improve on random resampling and in fact — aside from GEN(0.5) — do noticeably worse than FS2 in the sparse environment.

For the dense case the results are mostly 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. RES and GEN do better in the dense case, with performance comparable to that of FS2. In the dense environment, increasing α (placing more emphasis on representation accuracy than on particle diversity) leads to improved performance, a clue that generalized resampling is offering little benefit in this case aside from the expected slight reduction in degeneracies.

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 (Bailey et al., 2006).) While the estimation error using fixed-lag roughening and the 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.

6.2 $\hat{N}_{\text{eff}}$ comparison

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

Figures 6-7 show $\hat{N}_{\text{eff}}$ 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 $\hat{N}_{\text{eff}}$ 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

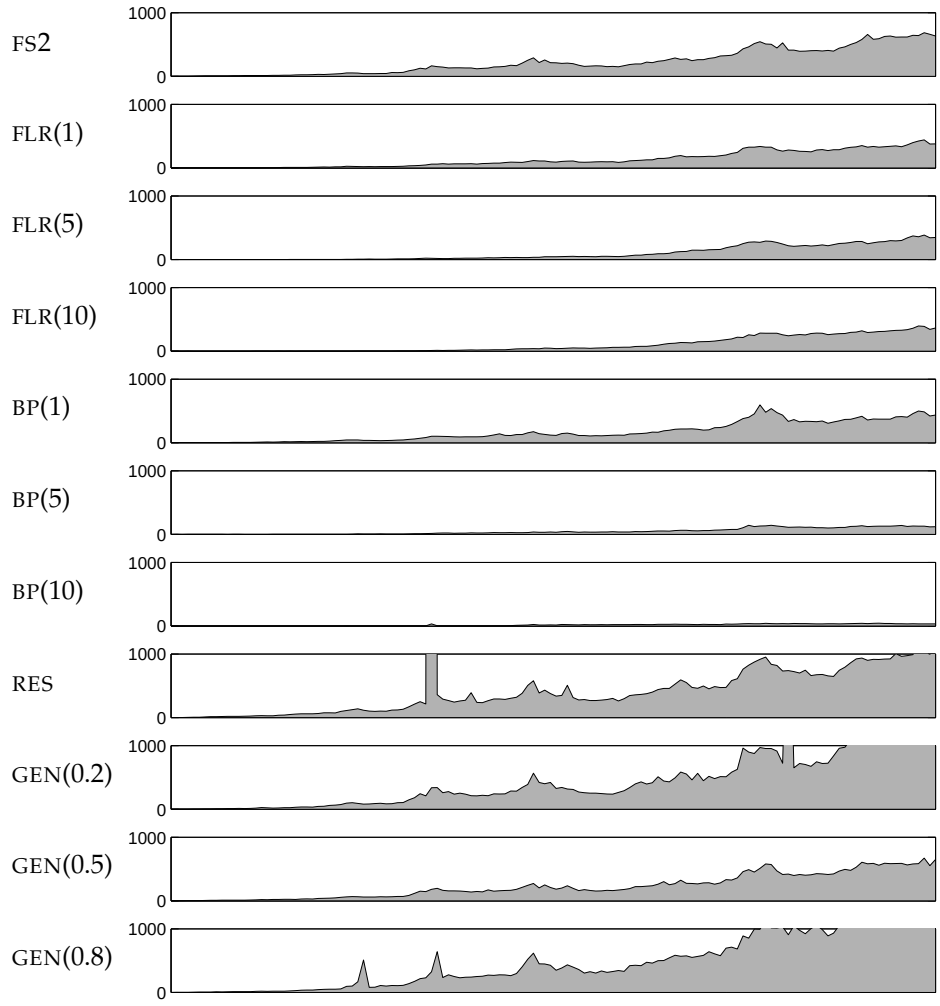


Figure 2: NEES for the sparse environment, versus the simulation time on the x-axis. We use the following abbreviations: FS2 for FastSLAM 2 with random resampling, FLR(L) for fixed-lag roughening with lag time L, BP(L) for the block proposal with lag time L, RES for FastSLAM 2 with residual resampling, and GEN(α) for generalized resampling with parameter α.

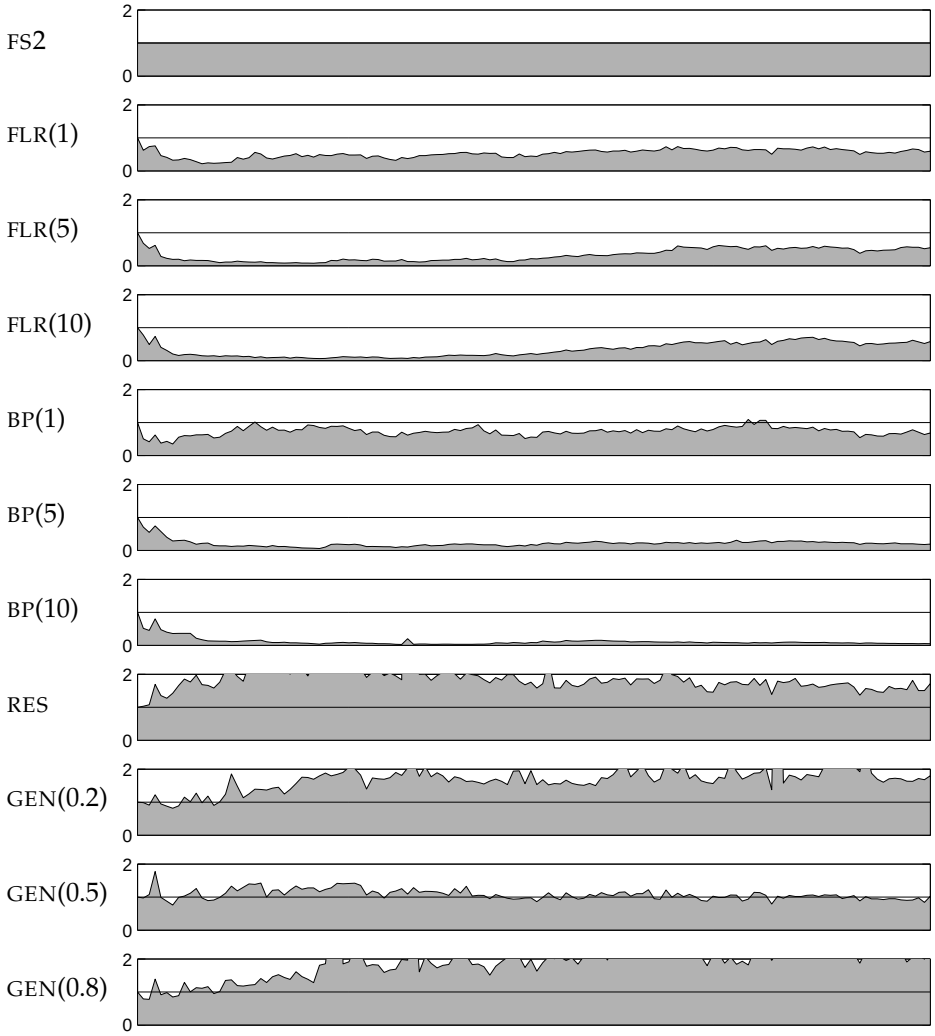


Figure 3: Ratio of NEES to that of FastSLAM 2 for the sparse environment.

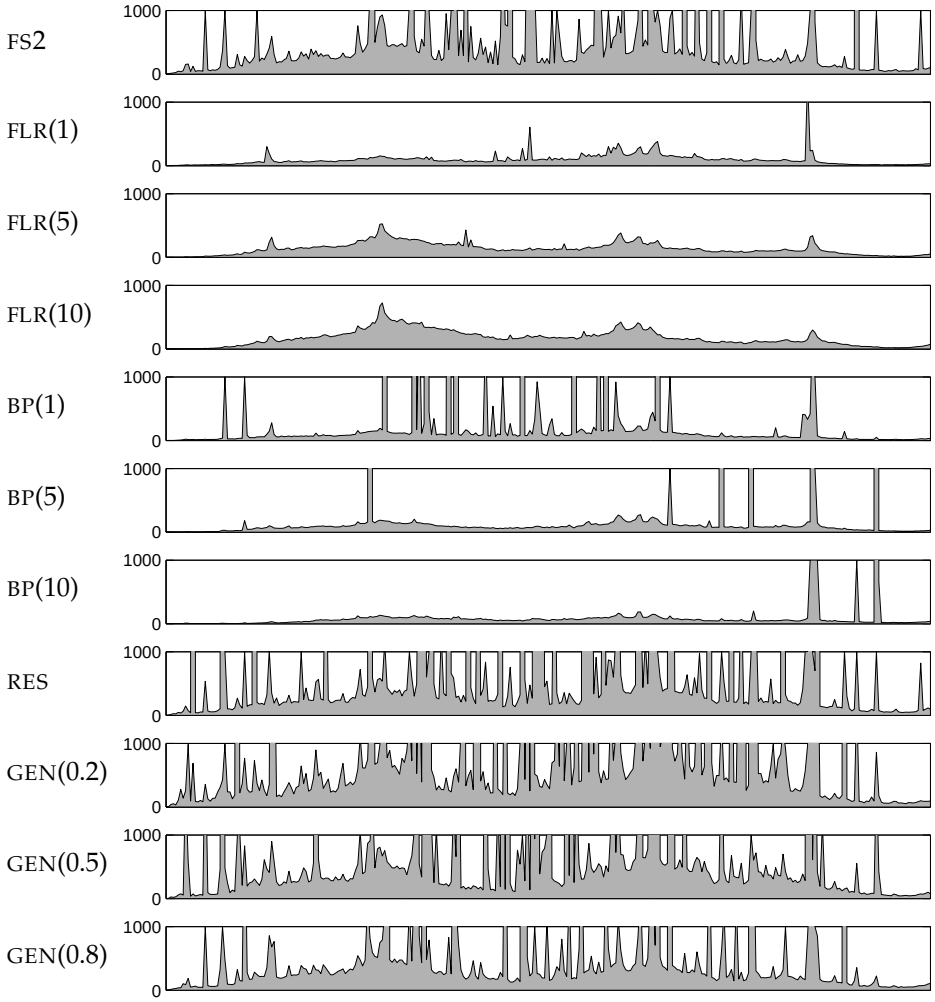


Figure 4: NEES for the dense environment.

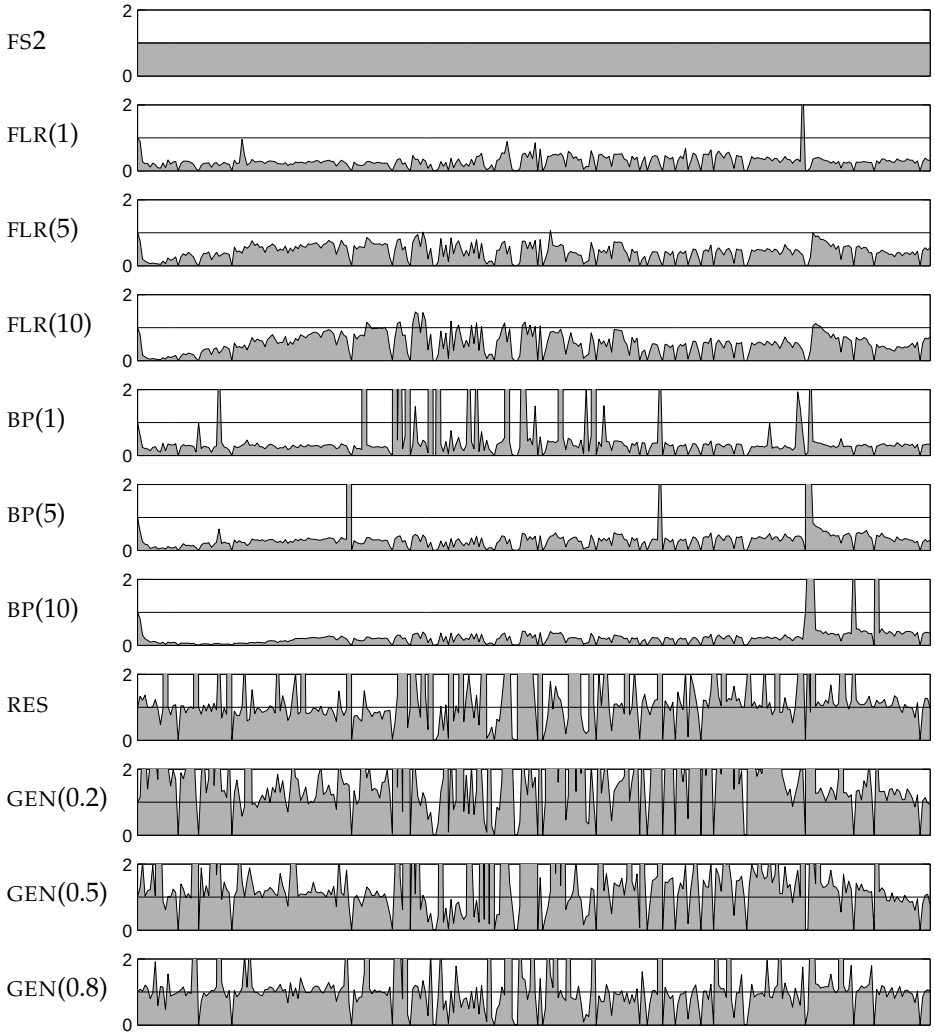


Figure 5: Ratio of NEES to that of FastSLAM 2 for the dense environment.

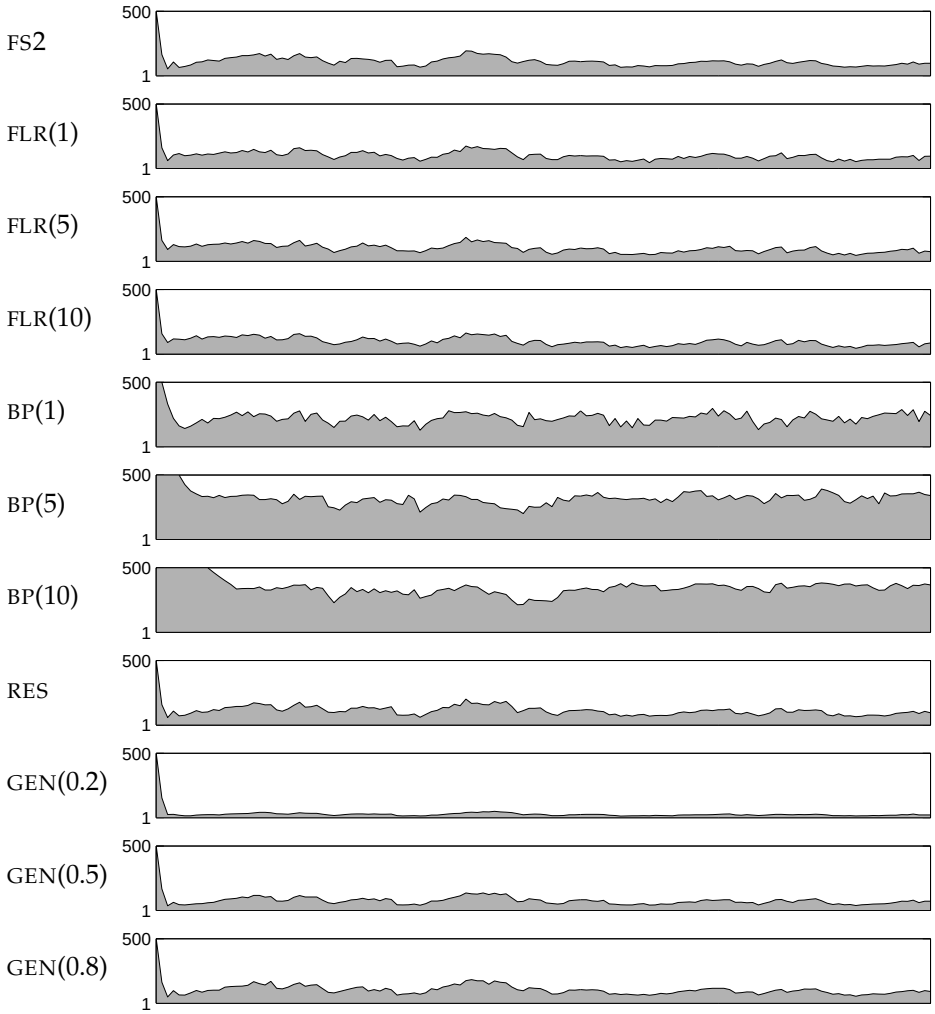


Figure 6: $\hat{N}_{\text{eff}}$ for the sparse environment, versus the simulation time on the x-axis.

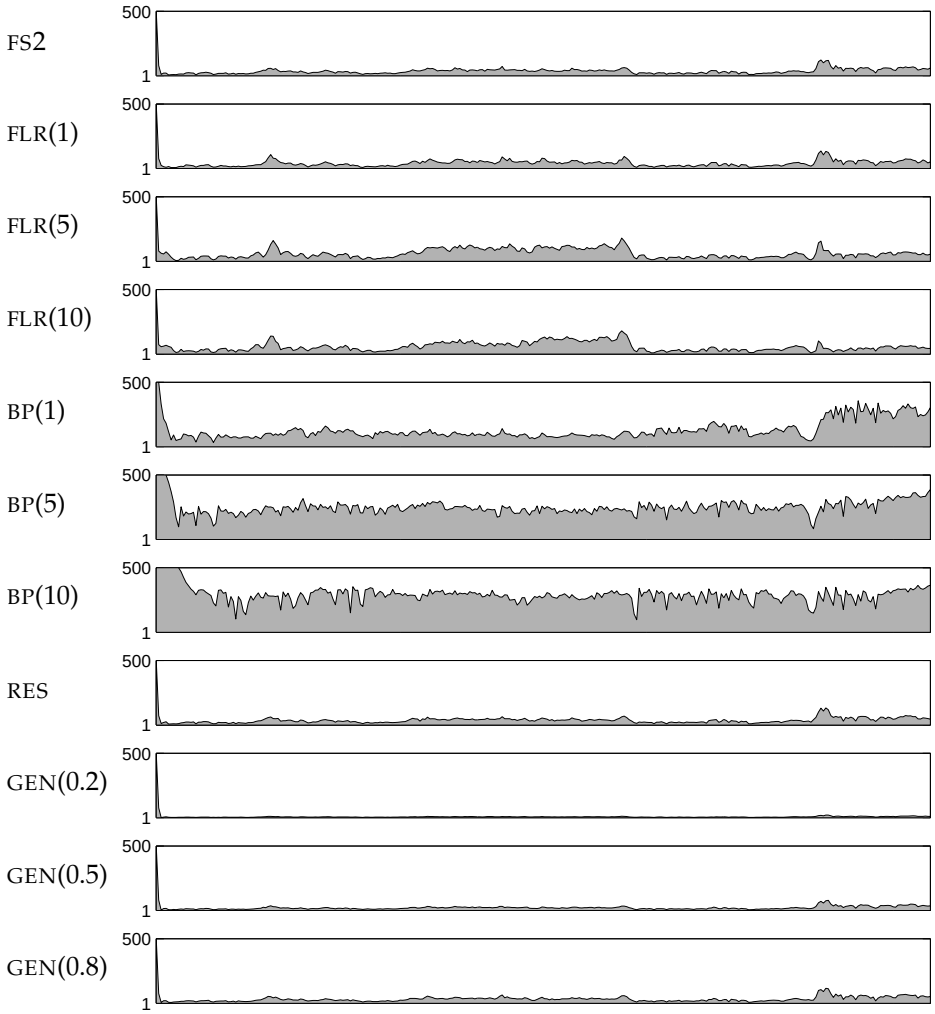


Figure 7: $\hat{N}_{\text{eff}}$ for the dense environment.

the 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. Residual resampling gives slightly better performance than FS2 (a result of its deterministic approach). GEN yields worse $\hat{N}_{\text{eff}}$ than FS2 in all cases, with the ratios approximately proportional to the parameter α. Again, this is the expected result since generalized resampling trades off representation accuracy (which is measured by $\hat{N}_{\text{eff}}$) for particle diversity.

6.3 Particle diversity

Finally, we examine particle diversity for each of the different filters. Figures 8 and 10 show the variance of the pose histories of all the particles, computed at the end of SLAM. Figures 9 and 11 show 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. RES gives slightly worse performance than FS2. GEN very slightly improves on FS2, with about 1% more unique samples of each pose on average for $\alpha = 0.2$ and $\alpha = 0.5$.

7 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.

Our results show that the new algorithms lead to substantial improvements in SLAM estimation. Fixed-lag roughening and the block proposal yield samples which exhibit less statistical estimation error than those of FastSLAM 2. Furthermore, the samples drawn by the block proposal distribution tend to have much more uniform importance weights (and thus higher "effective sample sizes"), leading to less need for resampling and consequently, improved particle diversity.

We have also examined the utility of two alternative resampling algorithms. Residual resampling deterministically selects a number of copies of each particle proportional to the weights, consuming many fewer random numbers than the usual random resampling. Generalized resampling selects samples according to some function of their importance weights, giving flexibility in balancing particle diversity with representation accuracy. While our results have shown that neither technique significantly improves upon the estimation performance of basic random resampling, the approaches expand the resampling toolkit; in particular, residual resampling may be useful in computationally restricted scenarios.

Our experiments so far have been in simulation for purposes of comparison. We are currently working to implement fixed-lag roughening and the block proposal distribution for line features, which are more suitable for use in real-world indoor mapping.

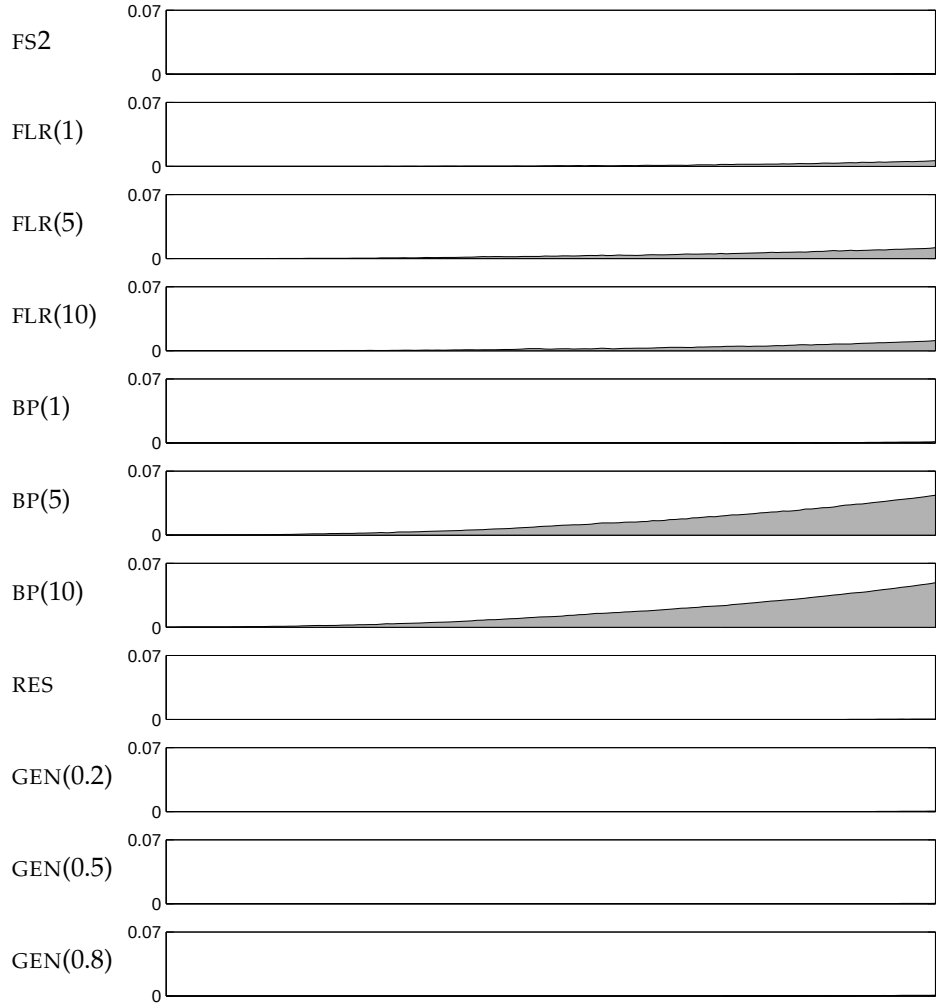


Figure 8: Sample variance of the final trajectories for the sparse environment. The plots shown here are computed using the trajectory samples at the end of SLAM, i.e., $\{\mathbf{x}_{1:t}^{r,i} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}\}$.

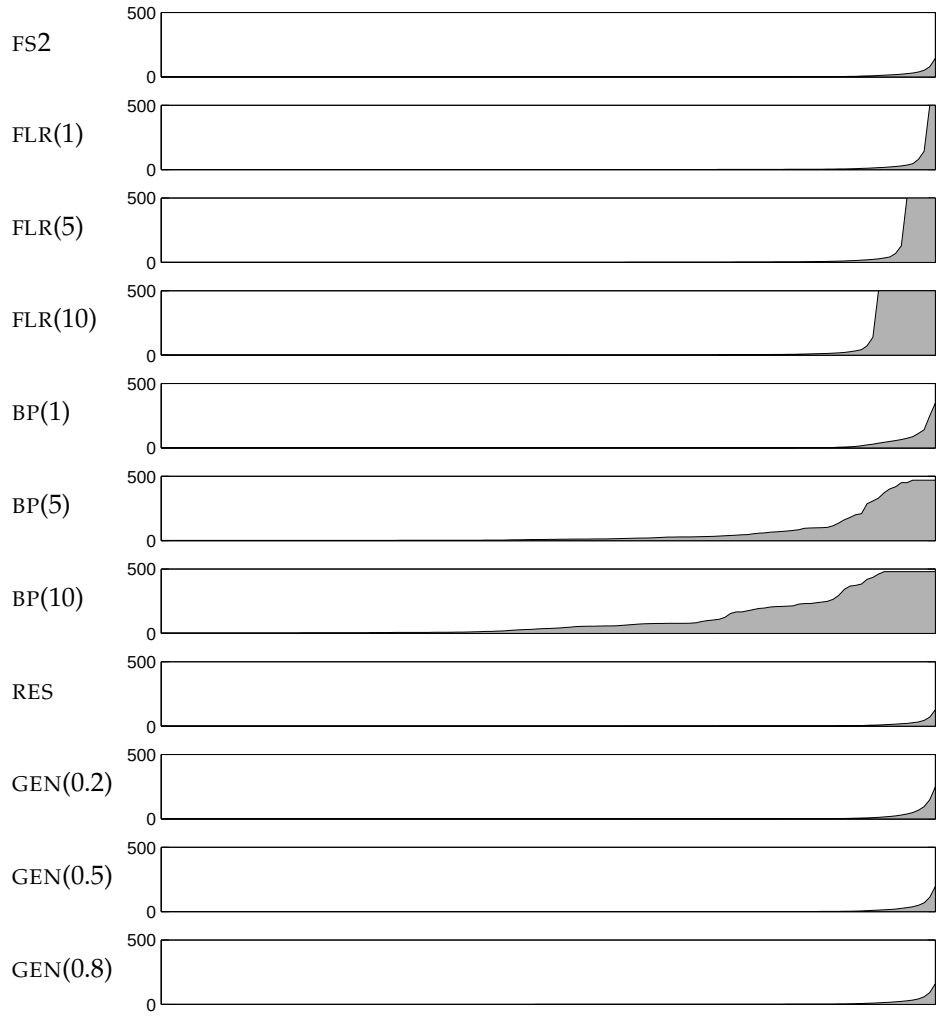


Figure 9: Number of unique samples representing each pose in the trajectory at the end of SLAM, for the sparse environment.

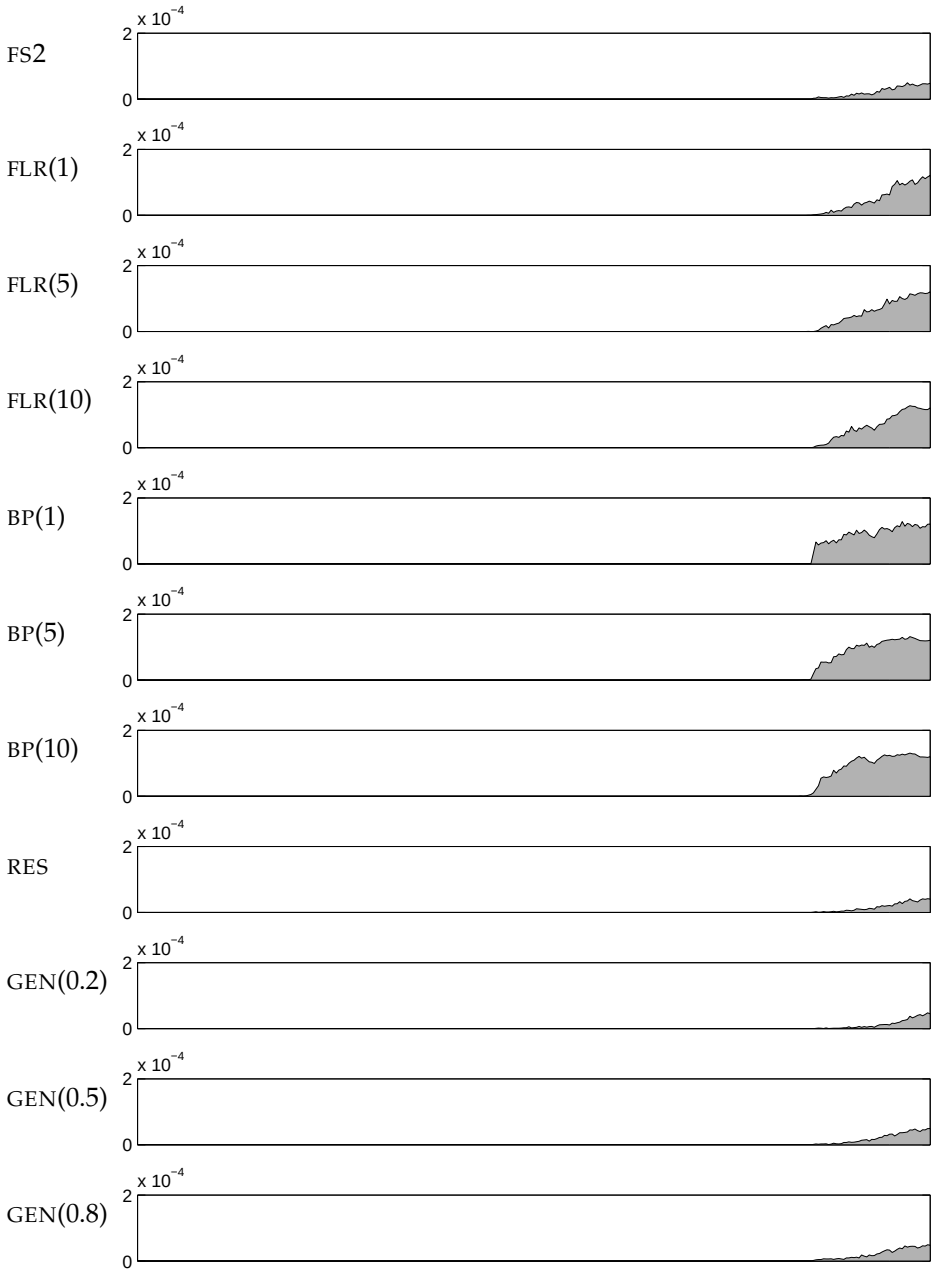


Figure 10: Sample variance of the final trajectories for the dense environment.

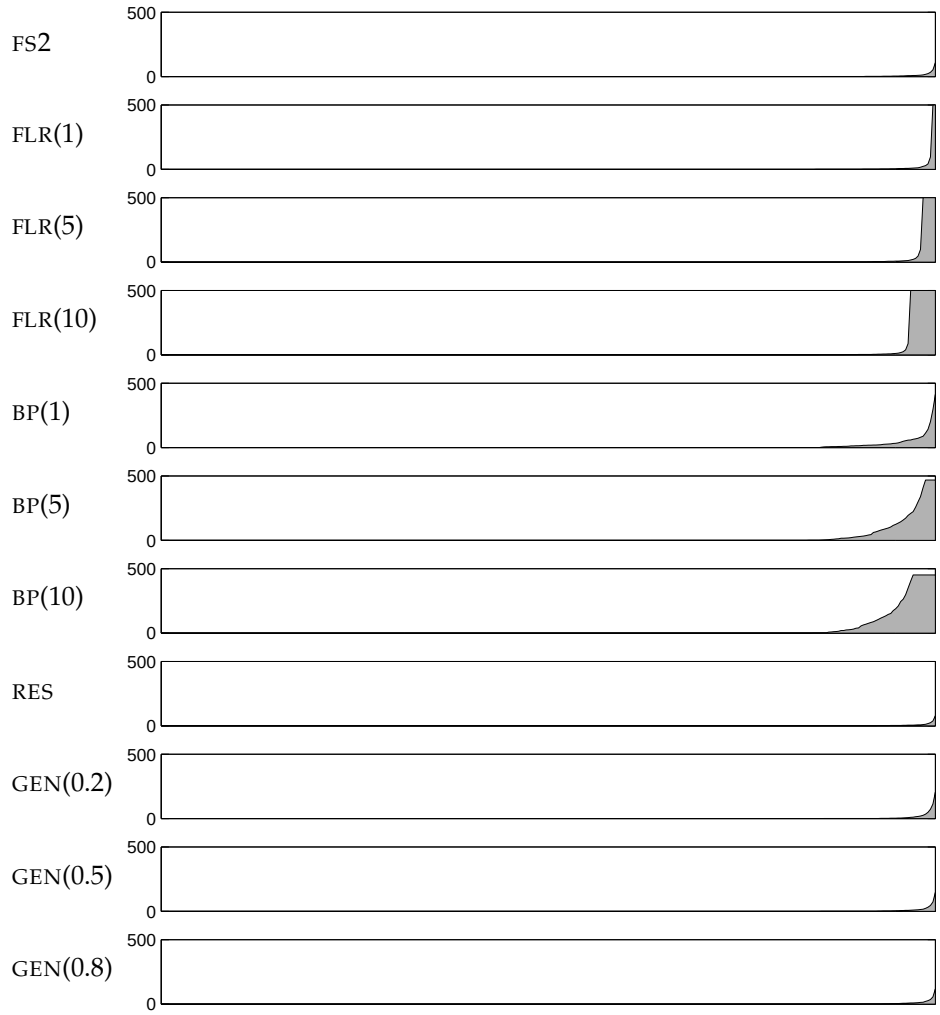


Figure 11: Number of unique samples representing each pose in the trajectory at the end of SLAM, for the dense environment.

8 Acknowledgement

The author would like to thank Wes Huang for various discussions on this topic, and for his help in editing several drafts.

References

T. Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, Australian Center for Field Robotics, University of Sydney, August 2002.

T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *IEEE Intl. Conf. on Robotics and Automation*, pages 424–427, 2006.

Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation*. Wiley, New York, 2001.

K.R. Beevers and W.H. Huang. Inferring and enforcing relative constraints in SLAM. In *Workshop on the Algorithmic Foundations of Robotics (WAFR 2006)*, New York, July 2006.

K.R. Beevers and W.H. Huang. Fixed-lag sampling strategies for particle filtering SLAM. 2007 IEEE International Conference on Robotics and Automation (ICRA 2007), submitted, April 2007.

S. Chib. Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, 75(1):79–97, 1996.

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.

W.R. 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.

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.

J.S. Liu. *Monte Carlo strategies in scientific computing*. Springer, New York, 2001.

J.S. Liu and R. Chen. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576, June 1995.

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.

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.