Algorithm 3 *Phase 2, Compute Messages for Tracking:* Given M samples $\chi_{s,\tau}^{(i)} = \{x_{s,\tau}^{(i)}, \dot{x}_{s,\tau}^{(i)}\}$ from marginal $\hat{p}_{s,\tau}^{n-1}(\chi_{s,\tau})$, compute messages $m_{st,\tau}^{n}(x_{t,\tau})$ for neighbors $t \in \Gamma(s,\tau)$, $m_{s,\tau,\tau+1}(\chi_{s,\tau+1})$, and $m_{s,\tau,\tau-1}(\chi_{s,\tau-1})$

1. Draw $\theta_{st}^{(i)} \sim \mathbb{U}(0, 2\pi)$, $\quad \nu_{st}^{(i)} \sim \mathbb{N}(0, \sigma_\nu^2)$
2. Means: $x_{st,\tau}^{(i)} = x_{s,\tau}^{(i)} + (d_{st,\tau} + \nu_{st}^{(i)})[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})]$
3. Weights: $w_{st,\tau}^{(i)} = P_o(x_{st,\tau}^{(i)}) / m_{ts,\tau}^{n-1}(x_{s,\tau}^{(i)})$
4. Variance: $\Sigma_{st,\tau} = \sigma_\nu^2 \xi_M I$
5. Means: $\chi_{s,\tau+1}^{(i)} \sim p(\chi_{s,\tau+1} \mid \chi_{s,\tau}^{(i)})$
6. Weights: $w_{s,\tau+1}^{(i)} = 1 / m_{s,\tau+1,\tau}^{n-1}(\chi_{s,\tau}^{(i)})$
7. Variance: $\Sigma_{s,\tau+1} = \mathrm{ROT}(\{\chi_{s,\tau+1}^{(i)}, w_{s,\tau+1}^{(i)}\})$
8. Means: $\chi_{s,\tau-1}^{(i)} \sim \alpha p(\chi_{s,\tau}^{(i)} \mid \chi_{s,\tau-1})$
9. Weights: $w_{s,\tau-1}^{(i)} = 1 / m_{s,\tau-1,\tau}^{n-1}(\chi_{s,\tau}^{(i)})$
10. Variance: $\Sigma_{s,\tau-1} = \mathrm{ROT}(\{\chi_{s,\tau-1}^{(i)}, w_{s,\tau-1}^{(i)}\})$

time message. As described in Sec. IV-D, we first decompose these messages into the marginal robot position distribution, and the conditional distribution of velocity given position; both are Gaussian mixtures. Using a generalized NBP message-passing algorithm, we then integrate the information provided by temporal dynamics and distance measurements at time τ. The resulting improved position estimates are then transferred to velocity estimates via the previously computed conditional densities. As summarized in the pseudocode, several stages of importance sampling with resampling are included to convert sets of message samples into the message products required by NBP, and ensure that samples (and hence computational resources) are efficiently utilized.

A. Message Update Schedules

Our tracking scheduler applies the phase II algorithms, after first initializing location estimates for each robot using phase I. Because we assume somewhat informative dynamics models, the uncertainty in robot locations at the next timestep is sufficiently peaked to allow a serial message update schedule. Our tracking framework is sufficiently general to perform smoothing, and use information from future timesteps to update location estimates for the current timestep, which in SLAM is important for tasks such as loop closure [2]. While alternative schedules are subjects for future work, we focus here in a filtering schedule which only propagates messages forward in time. We perform one tracking iteration following localization in timestep 1, and then a constant number of iterations at subsequent times.

B. Computational Complexity

The most computationally demanding portion of our tracker requires $\mathcal{O}(kM^2|E|)$ operations per timestep, where $|E|$ is the number of observed inter-distance measurements. The parameters k and M correspond to the number of samples used to represent distributions as described in Sec. IV-C. Our experiments use k = 3 and M varies from 100-500.

Algorithm 4 *Phase 2, Compute Marginals for Tracking* Use incoming messages $m_{ts,\tau}^{n} = \{x_{ts,\tau}^{(i)}, w_{ts,\tau}^{(i)}, \Sigma_{ts,\tau}\}$, $m_{s,\tau-1,\tau}^{n} = \{\chi_{s,\tau-1}^{(i)}, w_{s,\tau-1}^{(i)}, \Sigma_{s,\tau-1}\}$, and $m_{s,\tau+1,\tau}^{n} = \{\chi_{s,\tau+1}^{(i)}, w_{s,\tau+1}^{(i)}, \Sigma_{s,\tau+1}\}$ to compute marginal $\hat{p}_{s,\tau}^{n}(\chi_{s,\tau})$

1. Let $m_{s,\tau-1,\tau}^{n}(\chi_{s,\tau}) = m_{s,\tau-1}^{n}(x_{s,\tau}) m_{s,\tau-1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau})$
2. Let $m_{s,\tau+1,\tau}^{n}(\chi_{s,\tau}) = m_{s,\tau+1}^{n}(x_{s,\tau}) m_{s,\tau+1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau})$
3. Draw $\frac{kM}{3}$ samples $\{x_{s,\tau}^{(i)}\}$ from $m_{s,\tau-1}^{n}(x_{s,\tau})$
4. Draw $\frac{kM}{3}$ samples $\{x_{s,\tau}^{(i)}\}$ from $m_{s,\tau+1}^{n}(x_{s,\tau})$
5. **for** Neighbor $t \in \Gamma(s,\tau)$ **do**
6. Draw $\frac{kM}{3|\Gamma(s,\tau)|}$ samples $\{x_{s,\tau}^{(i)}\}$ from each $m_{ts,\tau}^{n}(x_{s,\tau})$
7. **end for**
8. Weight each of the kM samples by $w_{s,\tau}^{(i)} = \frac{m_{s,\tau-1}^{n}(x_{s,\tau}^{(i)}) \cdot m_{s,\tau+1}^{n}(x_{s,\tau}^{(i)}) \cdot \prod_{t \in \Gamma(s,\tau)} m_{ts}^{n}(x_{s,\tau}^{(i)})}{m_{s,\tau-1}^{n}(x_{s,\tau}^{(i)}) + m_{s,\tau+1}^{n}(x_{s,\tau}^{(i)}) + \sum_{t \in \Gamma(s,\tau)} m_{ts}^{n}(x_{s,\tau}^{(i)})}$
9. Resample with replacement from these kM samples, producing M equally weighted samples $\{x_{s,\tau}^{(i)}\}$.
10. Draw $\frac{kM}{2}$ samples $\dot{x}_{s,\tau}^{(i)} \sim m_{s,\tau-1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau}^{(i)})$, producing samples $\chi_{s,\tau}^{(i)} = \{x_{s,\tau}^{(i)}, \dot{x}_{s,\tau}^{(i)}\}$ from the joint marginal
11. Draw $\frac{kM}{2}$ samples $\dot{x}_{s,\tau}^{(i)} \sim m_{s,\tau+1}^{n}(\dot{x}_{s,\tau} \mid x_{s,\tau}^{(i)})$, producing samples $\chi_{s,\tau}^{(i)} = \{x_{s,\tau}^{(i)}, \dot{x}_{s,\tau}^{(i)}\}$ from the joint marginal
12. Weight each of the kM joint samples by $w_{s,\tau}^{(i)} = \frac{m_{s,\tau-1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)}) \cdot m_{s,\tau+1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)})}{m_{s,\tau-1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)}) + m_{s,\tau+1}^{n}(\dot{x}_{s,\tau}^{(i)} | x_{s,\tau}^{(i)})}$
13. Resample with replacement from these kM samples, producing M equally weighted samples $\{\chi_{s,\tau}^{(i)}\}$.

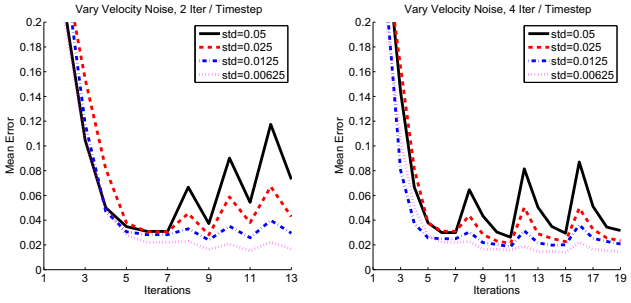

Fig. 4. Tracking error with varying levels of dynamics noise, using either two NBP iterations per timestep (left) or four iterations per timestep (right). The "sawtooth" shape of the error plot is because the each first iteration has the most error and reduces subsequent iterations. More iterations per timestep reduces errors, especially for noisier dynamics models. Our NBP algorithm makes use of multi-hop inter-distance readings, integrated over multiple iterations, to compensate for transition model errors. For a 100x100 meter room, a standard deviation of 0.025 corresponds to 2.5 meters.

Most time is spent either sampling from Gaussian mixtures, or evaluating Gaussian mixtures at a particular location. Using a naive approach, both operations require $\mathcal{O}(M)$ operations for an M-component mixture. Binary search algorithms can draw samples in $\mathcal{O}(\log(M))$ time; our simulator implements this technique. We can improve the speed of Gaussian mixture evaluation by constraining estimated covariance matrices to be diagonal. Multiscale, KD-tree representations can also be used to accelerate evaluation and sampling for large M [32].