Nonparametric Belief Propagation for Distributed Tracking of Robot Networks with Noisy Inter-Distance Measurements

Jeremy Schiff, Erik B. Sudderth, Ken Goldberg
{jschiff, sudderth}@eecs.berkeley.edu, goldberg@berkeley.edu

***Abstract*— We consider the problem of tracking multiple moving robots using noisy sensing of inter-robot and inter-beacon distances. Sensing is local: there are three fixed beacons at known locations, so distance and position estimates propagate across multiple robots. We show that the technique of Nonparametric Belief Propagation (NBP), a graph-based generalization of particle filtering, can address this problem and model multi-modal and ring-shaped uncertainty distributions. NBP provides the basis for distributed algorithms in which messages are exchanged between local neighbors. Generalizing previous approaches to localization in static sensor networks, we improve efficiency and accuracy by using a dynamics model for temporal tracking. We compare the NBP dynamic tracking algorithm with SMCL+R, a sequential Monte Carlo algorithm [1]. Whereas NBP currently requires more computation, it converges in more cases and provides estimates that are 3 to 4 times more accurate. NBP also facilitates probabilistic models of sensor accuracy and network connectivity.**

I. Introduction

Emerging advances in sensor networks: sensors, processors, and wireless communications are yielding improvements in sensor and transmission robustness, smaller sizes, cheaper devices, and lower power usage. Collaborative self-localization and tracking using wireless sensors has many applications such as tracking pallets in warehouses, vehicles on roadways, or firefighters is burning buildings. In this paper we consider the problem of tracking multiple moving robots using noisy sensing of inter-robot and inter-beacon distances. Sensing is local: there are three fixed beacons at known locations, so distance and position estimates propagate across multiple robots.

Consider a graph where nodes correspond to beacons or mobile robots, and edges link pairs of nodes for which distance measurements are available. Robots may be multiple hops away from beacons. The inter-distance tracking problem is illustrated in Fig. 1. Inter-distance tracking is also closely related to simultaneous localization and mapping (SLAM) problems [2], in which each robot is treated as a uniquely identifiable, but mobile, landmark.

We formalize the inter-distance tracking problem using a probabilistic graphical model, which integrates prior estimates about beacon locations, sensor models, and robot dynamics. In contrast with previous robot tracking methods, we then apply a variant of the belief propagation (BP) [3] algorithm, called nonparametric belief propagation (NBP) [4], to infer globally consistent estimates of robot location from noisy, local distance measurements using this graphical model. NBP approximates posterior distributions of unobserved variables by sets of representative samples. It is a generalization of particle filters [2] to domains with richer, non-temporal structure. NBP does not require linear or Gaussian models [5] and can model multi-modal and ring-shaped distributions produced by distance sensors.

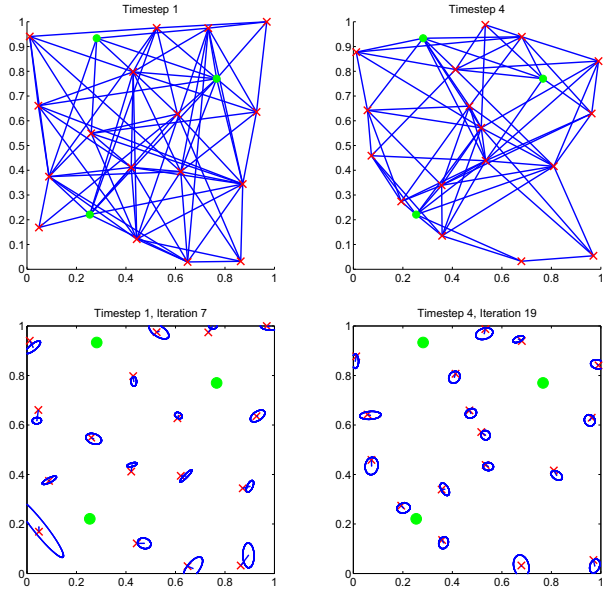

Fig. 1. Tracking seventeen robots at first and fourth timestep. The top figures depict inter-distance sensor connectivity: edges link pairs of robots that share a distance reading. Three green discs denote fixed reference beacons. True robot locations are indicated with red crosses. The bottom figures overlay true robot locations with point estimates of robot location generated by the NBP algorithm, shown as blue dots, at the center of blue ellipses, indicating twice the standard deviation. Note that although robots move substantially between timesteps and few robots are in direct contact with beacons, our inference shows good performance.

The algorithm runs in two phases. Sec. V describes phase I, where we localize robots at the first timestep, using a similar formulation to the static localization algorithm of Ihler et al. [6]. Sec. VI describes phase II, where a dynamics model is used to combine inter-distance measurements over time.

Our tracking algorithm uses dynamics models to improve accuracy, while reducing computation and commu-

J. Schiff and E. Sudderth are with the Dept. of EECS and K. Goldberg is with the Depts. of EECS, IEOR and the School of Information, University of California, Berkeley, CA, 94720, USA.

This work was funded in part by NSF CISE Award: Collaborative Observatories for Natural Environments (Goldberg 0535218, Song 0534848), and by NSF Science and Technology Center: TRUST, Team for Research in Ubiquitous Secure Technologies, with additional support from Cisco, HP, IBM, Intel, Microsoft, Symmantec, Telecom Italia and United Technologies.

nication requirements by up to 3 times when compared to a localization-only approach for a 20 robot network. In our experiments, we demonstrate cases where our tracking algorithm correctly determines the location of the robot, while processing each timestep independently fails to do so. Other approaches, for example those based on directly weighting temporal samples by inter-distance likelihoods, failed to localize the robots. Our approach, based on a novel decomposition of the temporal dynamics model, allows us to integrate multi-hop inter-distance readings in a single timestep and track the robots. We also show how NBP message updates should be scheduled to achieve robust, reliable convergence.

In the Experiments section, we compare our algorithm to a Monte-Carlo approach developed by Dil et al. [1]. We find that our algorithm is slower but 3 to 4 times more accurate, and more robust to noisy inter-distance sensor readings.

II. Related Work

Many robotics problems are related to our collaborative inter-distance tracking problem, including simultaneous localization and mapping (SLAM), simultaneous localization and tracking (SLAT), control of robotic swarms, and robot/sensor network localization. We distinguish between localization approaches, which use sensor readings from a single timestep (perhaps in series over multiple timesteps), and tracking approaches, which integrate location estimates using models of robot dynamics.

In classic SLAM problems, a single mobile robot produces a map of the environment while simultaneously determining its location. Several recent approaches allow SLAM problems to be efficiently solved with as many as 10^8 static features where distributions are modeled as multivariate Gaussians. Many of these methods are based on inference in an appropriate graphical model; for an overview, see [2]. SLAM has also been extended to domains with multiple robots, often under the assumption that the initial locations of the robots are known [7], [8]. In these approaches, robots share and localize using a global map, rather than through distributed observations of other robots as in our approach. Thrun and Liu [9] investigate merging multiple maps when initial locations of robots are unknown, and the landmarks are not uniquely identifiable. Howard [10] explores map merging when robots do not know their initial locations. In contrast, our approach uses a series of inter-robot sensor values.

The SLAT problem is a variant of SLAM in which static robots, which are typically nodes in a sensor network, use distance and bearing readings from a moving robot to localize themselves. The approach of Taylor et al. [11] avoids representing ring-shaped or multi-modal posterior distributions by assuming sensors roughly know their initial locations, and batch processing sets of distance measurements. Funiak et al. [12] propose a more complex method focused on SLAT problems in camera networks.

In distributed control, methods have been proposed for controlling robot swarms to maintain a specified formation [13], or using several mobile robots to localize a target [14]. These approaches assume sensors observe true locations plus Gaussian noise, resulting in posterior distributions which (unlike those arising in inter-distance tracking) are well approximated as Gaussian. However, our approach shares the goal of developing effective distributed algorithms for multiple robots.

A great deal of research focuses on distance-based localization in sensor networks; for a summary, see [15]. These localization techniques can be applied at each timestep to determine the location of each robot, but ignore dynamics. To address static localization problems, researchers have applied techniques such as multidimensional scaling (MDS), in both centralized [16] and distributed [17] implementations. Many localization methods produce global solutions by incrementally stiching local sensor maps together [17], for instance by recursively localizing new sensors with respect to previously localized sensors [18]. Other approaches solve more global inference problems, computing location estimates via iterative least squares [19], [20]. Such methods can be very effective when bearing estimates are available, as in the bundle adjustment methods widely used in camera networks [21]. However, with the greater posterior uncertainty produced by inter-distance measurements, the approximate marginal distributions computed by NBP are often more robust and effective [6].

In some multi-robot tracking applications, a sufficient number of beacon nodes (at least three) can be directly observed by all robots at all times [22], [23]. Other approaches assume sensors measure both orientation and distance, allowing for simplifying Gaussian assumptions [24]. Previous approaches to the inter-distance tracking problem include the work of Howard et al. [25], which formulates inter-distance tracking as numerical optimization; Park et al. [26], which solves a trilateration problem using quadratic approximations of the motion of robots relative to one another; and Dil et al. [1], which uses assumptions about maximum transmission radii to apply particle filters.

In contrast with previous localization methods for networks of mobile robots, our distributed tracking algorithm is based on applying the nonparametric BP algorithm to a probabilistic graphical model. Schiff et al. [27] previously demonstrated the feasibility of running BP in a real network of MICA2 motes for static sensor fusion.

III. Problem Formulation

A. Input, Assumptions, and Output

Our models supports tracking applications where n robots move around a space for T timesteps. The location of robot s at time τ is denoted by $x_{s,\tau} \in \mathbb{R}^2$. We currently model the space as a 1x1 unit square, but our approach can be easily relaxed to support other geometries. Each robot is equipped with sensors for inter-distance measurement. We denote the (noisy) distance estimate between robots s and t at time τ as $d_{st,\tau}$. As distance estimates are based on signals emitted by nearby robots, the likelihood a sensor will detect a nearby sensor diminishes with distance. We refer to the subset of

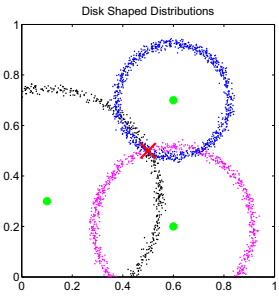

Fig. 2. Samples from the distributions of each of three beacons estimations of a robot's location, resulting in three ring-shaped distributions

robots which receive distance readings from robot s at time τ as the neighbors of s, and denote them by $\Gamma(s,\tau)$.

Our objective is to determine position estimates $\hat{x}_{s,\tau}$ which minimize the mean squared error of each robot's location estimates, averaged over all time steps:

$$L(x,\hat{x}) = \frac{1}{nT}\sum_{\tau=1}^{T}\sum_{s=1}^{n}||\hat{x}_{s,\tau} - x_{s,\tau}||^2 \qquad (1)$$

The NBP algorithm provides non-parametric distributions at each time step that can be used to quantify confidence based on variance. A multi-modal distribution can be resolved in a variety of ways, as discussed in the Future Work section.

B. Modeling System Properties

The state of each robot s at time τ is represented by its position and velocity $\chi_{s,\tau} = (x_{s,\tau}, \dot{x}_{s,\tau})$. We model errors in the estimated distance between robots s and t as follows:

$$d_{st,\tau} = ||x_{s,\tau} - x_{t,\tau}|| + \nu_{st,\tau} \qquad \nu_{st,\tau} \sim \mathbb{N}(0,\sigma_\nu^2) \qquad (2)$$

To unambiguously localize a robot, inter-distance readings from at least three other localized robots or beacons are required. Therefore, for localization at the first timestep, the network must contain at least three beacons and all robots must have at least three inter-distance readings. We demonstrate three ring-shaped distributions from exact beacons, used to localize a single robot in Fig. 2.

Following [20], we model the decay in a robot's probability P_o of measuring the distance to another robot as follows:

$$P_o(x_{s,\tau}, x_{t,\tau}) = \exp\left\{-\frac{1}{2}||x_{s,\tau} - x_{t,\tau}||^2/R^2\right\} \qquad (3)$$

Here, R is a parameter specifying the range of the transmitter used for distance estimation. More elaborate models could be used to capture environmental factors or multi-path effects.

Each robot moves over time according to a dynamics model $P(\chi_{s,\tau+1} \mid \chi_{s,\tau})$ defined as follows:

$$x_{s,\tau+1} = x_{s,\tau} + \dot{x}_{s,\tau+1} \qquad (4)$$

$$\dot{x}_{s,\tau+1} = \dot{x}_{s,\tau} + \omega_{s,\tau} \qquad \omega_{s,\tau} \sim \mathbb{N}(0,\sigma_\omega^2) \qquad (5)$$

In practice, these velocities are often unobserved variables used to explain the typically smooth character of real robotic motion. Alternatively, sensors such as accelerometers could provide more precise dynamics information.

IV. Background

A. Undirected Graphical Models

We model the relationships among the random variables in our tracking problem using an undirected graphical model, or pairwise Markov random field (MRF) [28]. This model is specified via an undirected graph G, with vertex set $V = \{1,\ldots,n\}$ and edge set E. Each node $s \in V$ is associated with a random variable x_s. For notational simplicity, we focus here and in Sec. V on static localization, so that node variables are robot positions x_s. For the tracking problems considered in Sec. VI, they become temporal states $\chi_{s,\tau}$.

The graph G specifies a factorization of the joint distribution of these random variables into a product of local, non-negative compatibility functions:

$$p(x) = \frac{1}{Z}\prod_{s\in V}\psi_s(x_s)\prod_{(s,t)\in E}\psi_{st}(x_s,x_t) \qquad (6)$$

Here, Z is a normalization constant, and the compatibility functions encode the information provided by dynamics models and observed inter-distances.

B. Belief Propagation

Belief propagation (BP) [3] is a general inference algorithm for graphical models. Let $\Gamma(s) = \{t \in V \mid (s,t) \in E\}$ denote the set of neighbors of node s. In BP, each node s iteratively solve the global inference problem by integrating information via local computation, and then transmitting a summary message to each neighbor $t \in \Gamma(s)$. In general, this message $m_{st}(x_t)$ is a function encoding sufficient statistics needed for node t to perform the next round of computation. For tree-structured graphs, BP is a dynamic programming algorithm which exactly computes posterior marginal distributions $p_s(x_s)$ for all nodes $s \in V$. However, the same update equations are widely applied to graphs with cycles, in which case they provide (often good) approximate marginal estimates $\hat{p}_s(x_s)$ [3].

The BP algorithm begins by initializing all messages $m_{st}(x_t)$ to constant vectors, and then updates the messages along each edge according to the following recursion:

$$m_{st}(x_t) \leftarrow \int_{x_s}\psi_s(x_s)\psi_{st}(x_s,x_t)\prod_{u\in\Gamma(s)\setminus t}m_{us}(x_s)\,dx_s \qquad (7)$$

This sum-product algorithm is then iterated until the set of messages converges to some fixed point. The order in which the messages are updated is a design parameter, and various schedules (parallel, sequential, etc.) exist. Upon convergence, the messages can be used to compute approximations to the marginal distributions at each node:

$$\hat{p}_s(x_s) \propto \psi_s(x_s)\prod_{t\in\Gamma(s)}m_{ts}(x_s) \qquad (8)$$

C. Nonparametric BP: Distributions as Samples

For inter-distance tracking and localization problems, it is computationally infeasible to discretize the state space as required by the standard sum-product algorithm. Thus, we employ nonparametric belief propagation (NBP) [4],

which directly approximates the marginal distributions of continuous robotic states via a collection of M sample points. Messages are represented via a weighted mixture of M Gaussian distributions:

$$m_{st}(x_t) = \sum_{i=1}^{M} w_{st}^{(i)} \mathbb{N}(x_t \mid x_{st}^{(i)}, \Sigma_{st}) \tag{9}$$

Mixture components are centered on samples $x_{st}^{(i)}$ from the underlying, continuous message function, with weights $w_{st}^{(i)}$ set via importance sampling principles as detailed below. By choosing a number of samples M, we can tradeoff accuracy versus computational cost.

A variety of methods are available for choosing the covariance Σ_{st} used to smooth message samples [29]. In many cases, we use the computationally efficient "rule of thumb" estimate $\Sigma_{st} = \mathrm{ROT}(\{x_{st}^{(i)}, w_{st}^{(i)}\})$, which is proportional to the weighted covariance of the observed samples:

$$\Sigma_{st} = M^{\frac{-2}{\delta+4}} \sum_{i=1}^{M} w_{st}^{(i)} (x_{st}^{(i)} - \bar{x}_{st})(x_{st}^{(i)} - \bar{x}_{st})^T \tag{10}$$

Here, $\bar{x}_{st} = \sum_i w_{st}^{(i)} x_{st}^{(i)}$ is the weighted sample mean, and $\delta = \dim(x_t)$. However, for ring-shaped messages which are far from unimodal, the rule of thumb estimator performs poorly, significantly oversmoothing the estimated density. In such cases, we set $\Sigma_{st} = \sigma_\nu^2 \xi_M I$, where ξ_M is a constant calibrated offline to the number of samples M.

D. Decomposing Gaussian Mixtures

In the tracking algorithm developed in Sec. VI, it is necessary to decompose a Gaussian mixture $p(\chi) = p(x, \dot{x})$ into marginal and conditional distributions $p(x)p(\dot{x} \mid x)$. Because the conditional distribution of a subset of jointly Gaussian variables is Gaussian, conditional distributions of Gaussian mixtures are mixtures of lower-dimensional Gaussians. We now derive formulas for the marginal and conditional portions of each mixture component. As the same formulas apply to all components, we drop the $\cdot^{(i)}$ notation.

Consider a Gaussian mixture component with weight w_χ, and mean vector and covariance matrix

$$\mu_\chi = \begin{bmatrix} \mu_x \\ \mu_{\dot{x}} \end{bmatrix}, \qquad \Sigma_\chi = \begin{bmatrix} \Sigma_{x,x} & \Sigma_{x,\dot{x}} \\ \Sigma_{\dot{x},x} & \Sigma_{\dot{x},\dot{x}} \end{bmatrix}. \tag{11}$$

The marginal $p(x)$ has weight w_χ, mean μ_x, and covariance matrix Σ_x. The conditional density $p(\dot{x} \mid x = a)$ then has the following mean, covariance, and weight:

$$\mu_{\dot{x}|x} = \mu_{\dot{x}} + \Sigma_{\dot{x},x}\Sigma_{x,x}^{-1}(a - \mu_x) \tag{12}$$

$$\Sigma_{\dot{x}|x} = \Sigma_{\dot{x},\dot{x}} - \Sigma_{\dot{x},x}\Sigma_{x,x}^{-1}\Sigma_{x,\dot{x}} \tag{13}$$

$$w_{\dot{x}|x} \propto w_\chi \left(\frac{|\Sigma_{\dot{x}|x}|}{|\Sigma_\chi|}\right)^{\frac{1}{2}} \exp\left\{-\frac{1}{2}\sigma_0^T \Sigma_\chi^{-1} \sigma_0\right\} \tag{14}$$

The transformations from a single joint multivariate Gaussian to a single conditional multivariate Gaussian, transforming the mean and standard deviation, are determined by standard formulas[30]. The weight $w_{\dot{x}|x}$ depends on a centered observation vector:

$$\sigma_0 = \begin{bmatrix} a - \mu_x \\ \mu_{\dot{x}|x} - \mu_{\dot{x}} \end{bmatrix} \tag{15}$$

We use this decomposition to factorize dynamics messages sent between the same robot at subsequent timesteps. By first integrating distance readings via the smaller position-only marginals, and then drawing appropriately reweighted velocity samples, we can integrate multi-hop inter-distance information over time.

V. Phase I: Localization Algorithm

The goal of the inter-distance localization sub-problem is to infer the location of each node using only information from a single timestep (the first). Our formulation closely follows the algorithm proposed by Ihler et al. [6]. Since we consider a single timestep, we drop the τ notation and velocity state variables, and denote the position of robot s by x_s. The NBP algorithm alternates between sending messages to neighboring nodes, and computing marginals at nodes which received messages. We refer to each pair of steps, in which all messages are updated once, as an iteration.

To compute new outgoing messages, we use the marginal estimate $\hat{p}_s^{n-1}(x_s)$ from the previous iteration, which is represented by a set of samples. The inter-distance sensor model of Eq. (2) implies that message samples are most easily generated in polar coordinates, with random orientation and radius perturbed from the noisy inter-distance reading:

$$\theta_{st}^{(i)} \sim \mathbb{U}(0, 2\pi) \qquad \nu_{st}^{(i)} \sim \mathbb{N}\left(0, \sigma_\nu^2\right) \tag{16}$$

$$x_{st}^{(i)} = x_s^{(i)} + (d_{st} + \nu_{st}^{(i)})[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})] \tag{17}$$

When the BP algorithm sends a message from s to t, the incoming message product (as in Eq. (7)) avoids double-counting by excluding the message sent from t to s. To account for this in our NBP implementation, we use an importance sampling framework [4], [6]. Each sample $x_{st}^{(i)}$ is reweighted by $1/m_{ts}^{n-1}(x_s^{(i)})$, where $m_{ts}^{n-1}(x_s)$ is the preceding iteration's message (a Gaussian mixture). Also accounting for the effects of distance on the probability of receiving a measurement, the overall sample weight is

$$w_{st}^{(i)} = P_o(x_{st}^{(i)}) / m_{ts}^{n-1}(x_s^{(i)}). \tag{18}$$

More generally, given a proposal density $q(x)$ from which we can draw samples $x^{(i)}$, and a target density $p(x)$ we would like to approximate, importance sampling methods [31] assign a weight $w^{(i)} \propto p(x^{(i)})/q(x^{(i)})$ to each sample $x^{(i)}$.

Importance sampling methods are also used to approximate the marginal update of Eq. (8). Recall that incoming messages $m_{ts}(x_s)$ are mixtures of M Gaussians. With $d = |\Gamma(s)|$ neighbors, the exact message product of Eq. (8) will produce an intractable mixture with M^d components. To construct a proposal distribution, we instead take an equal number of samples $x_s^{(i)} \sim m_{ts}(x_s)$ from each incoming message. Combining these d groups of samples into a single

set, they are then assigned importance weights

$$w_s^{(i)} = \frac{\prod_{t\in\Gamma(s)} m_{ts}(x_s^{(i)})}{\sum_{t\in\Gamma(s)} m_{ts}(x_s^{(i)})}. \tag{19}$$

To reduce importance sampling variance, Alg. 2 introduces a parameter $k > 1$. First, kM samples are drawn from the messages, and then M samples are drawn with replacement from the weights assigned to the kM samples.

Algorithm 1 *Phase 1, Compute Messages for Localization:* Given M samples $\{x_s^{(i)}\}$ representing marginal $\hat{p}_s^{n-1}(x_s)$, compute outgoing messages $m_{st}^n(x_t)$ for neighbors $t \in \Gamma(s)$

1. Draw $\theta_{st}^{(i)} \sim \mathbb{U}(0, 2\pi)$, $\quad \nu_{st}^{(i)} \sim \mathbb{N}(0, \sigma_\nu^2)$
2. Means: $x_{st}^{(i)} = x_s^{(i)} + (d_{st} + \nu_{st}^{(i)})[\sin(\theta_{st}^{(i)});\ \cos(\theta_{st}^{(i)})]$
3. Weights: $w_{st}^{(i)} = P_o(x_{st}^{(i)}) / m_{ts}^{n-1}(x_s^{(i)})$
4. Variance: $\Sigma_{st} = \sigma_\nu^2 \xi_M I$

Algorithm 2 *Phase 1, Compute Marginals for Localization:* Use incoming messages $m_{ts}^n = \{x_{ts}^{(i)}, w_{ts}^{(i)}, \Sigma_{ts}\}$ from neighbors $t \in \Gamma(s)$ to compute marginal $\hat{p}_s^n(x_s)$

1. **for** Neighbor $u \in \Gamma(s)$ **do**
2. Draw $\frac{kM}{|\Gamma(s)|}$ samples $\{x_s^{(i)}\}$ from each message m_{ts}^n
3. Weight each sample by $w_s^{(i)} = \prod_{u\in\Gamma(s)} m_{us}^n(x_s^{(i)}) / \sum_{u\in\Gamma(s)} m_{us}^n(x_s^{(i)})$
4. **end for**
5. Resample with replacement from these kM samples, producing M equal-weight samples.

A. Message Update Schedules

The convergence behavior of NBP depends on the order in which messages are transmitted between robotic nodes. In initial experiments, we found that a naive serial schedule, which iterates through each node one by one, performed poorly even with dozens of iterations and thousands of samples per message. Because sample-based density approximations cannot effectively populate large areas, biases introduced by poor initial message approximations can overwhelm informative messages sent by other robots. To resolve this, we employ an alternative schedule in which a node only transmits an outgoing message if it has received an incoming message from at least three neighbors. In situations where no node has three incoming messages, the threshold drops to two messages, and then finally to one.

VI. Phase II: Tracking Algorithm

Phase I provides estimates of each robot's location at a single timestep. We then use phase II to incorporate the dynamics of the robot, allowing us to determine robot locations at subsequent timesteps more quickly, and improve accuracy by resolving ambiguities. Fig. 3 provides an example with one robot that illustrates how the dynamics model can compensate for ambiguities.

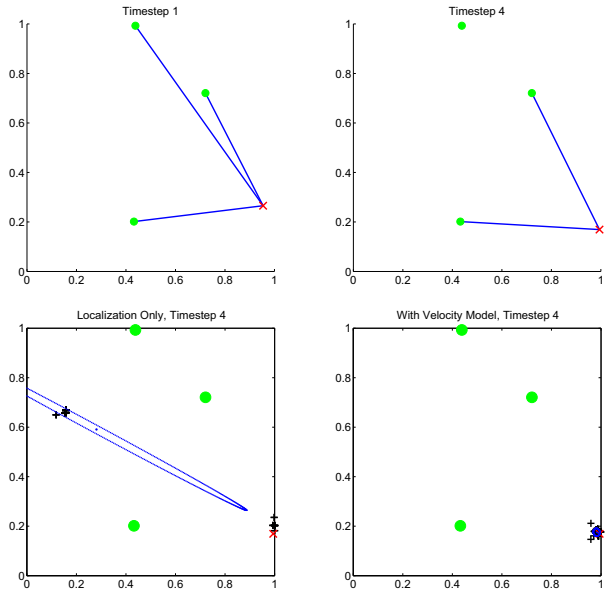

Fig. 3. Illustration of dynamics model resolving bimodal position estimate for a single robot with three beacons. Using the same notation as in Fig. 1, the upper figures show true locations of beacons and a robot in lower right of each plot. Samples from the robot marginal are indicated with black +s. Bottom figures show estimates of robot position for the 4th timestep, using a localization-only approach on the left, where there is substantial error, and using our tracking approach using dynamics on right, where error is negligible.

Our tracking algorithm for computing marginals in phase II integrates spatial messages from neighbors at the same timestep, as in the localization formulation, but also relates temporal messages for the same robot from the previous timestep $\tau - 1$, and the next timestep $\tau + 1$, when available. Given a sample of the node's current state $\chi_{s,\tau}^{(i)}$, the message sent to the next timestep is simulated from the dynamics model $\chi_{s,\tau+1}^{(i)} \sim p(\chi_{s,\tau+1} \mid \chi_{s,\tau}^{(i)})$. Because $\psi(x_{s,\tau+1}, x_{s,\tau}) = p(x_{s,\tau+1} \mid x_{s,\tau})$, when we send messages to a node at the previous timestep we fix $x_{s,\tau+1}^{(i)}$ and sample $x_{s,\tau}^{(i)} \sim \alpha p(x_{s,\tau+1}^{(i)} \mid x_{s,\tau})$, where α is a normalization constant. Concretely, the update becomes

$$\begin{aligned} x_{s,\tau}^{(i)} &= x_{s,\tau+1}^{(i)} - \dot{x}_{s,\tau+1}^{(i)} \\ \dot{x}_{s,\tau}^{(i)} &= \dot{x}_{s,\tau+1}^{(i)} - \omega_{s,\tau} \qquad \omega_{s,\tau} \sim \mathbb{N}(0, \sigma_\omega^2) \end{aligned} \tag{20}$$

We initially tested a simpler inference algorithm that propagates samples from the previous timestep according to the dynamics model, and then performs weighted resampling to account for new inter-distance readings. However, after more than four timesteps, this approach degraded to near-random estimates, even with thousands of samples M for each message. We hypothesize that this issue arises because the simpler formulation effectively only allows information to travel a single hop per time-step.

The more sophisticated approach of Algs. 3-4 overcomes these limitations by communicating multi-hop inter-distance information in a single timestep. Let $m_{s,\tau,\tau+1}(\chi_{s,\tau+1})$ denote the forward-time message sent by robotic node s at timestep τ, and $m_{s,\tau,\tau-1}(\chi_{s,\tau-1})$ the corresponding reverse-

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 $\{\chi_{s,\tau}^{(i)}\}$.

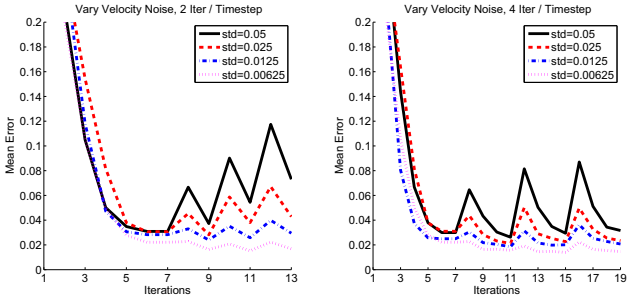

Fig. 4. Tracking error with varying levels of dynamics noise, using either two NBP iterations per timestep (left) or four iterations per timestep (right). The "sawtooth" shape of the error plot is because the each first iteration has the most error and reduces subsequent iterations. More iterations per 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].

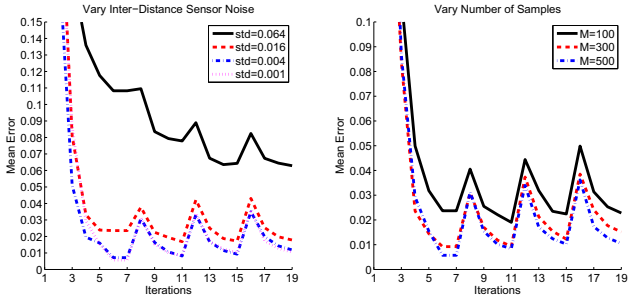

Fig. 5. NBP tracking performance in various conditions. In the left plot, we use $M = 500$ samples, and vary the amount of inter-distance noise. In the right plot, we use very accurate inter-distance sensors ($\sigma_\nu = 10^{-3}$), and vary the number of samples used to represent each NBP message. For a 100x100 meter room, a standard deviation of 0.004 corresponds to 0.4 meters.

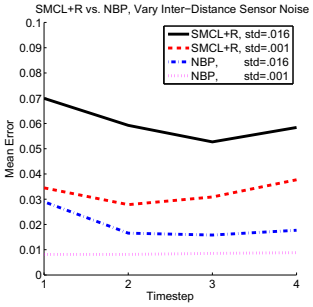

Fig. 6. A quantitative comparison of the range-based SMCL (SMCL+R) formulation [1] to our NBP tracking algorithm. As our approach is more computationally intensive, we used $M = 5000$ samples for SMCL+R, and $M = 500$ samples for NBP. We present results with two inter-distance sensor noise levels. In both cases, NBP is 3-4 times more accurate.

VII. Experiments

We implemented a Java simulator to explore our algorithm's performance and ran experiments on the single core of a 2.2 GHz Pentium Core Duo. While we simulate all robots on a single machine, as our algorithms are distributed, we interpret the runtime per robot. Our experiments using $M = 100$ samples took approximately 0.65 seconds per robot per timestep. With $M = 500$ samples, this grows to 16.0 seconds per robot per timestep. As described in Sec VI-B, there are remaining efficiency improvements we could employ to enable real-time operation on mobile robots.

For each experiment, we use $n = 20$ robots and 3 stationary beacons, the minimum required for the tracking problem to be well posed. We generate the initial locations at $t = 0$ by randomly placing each robot in a 1x1 unit area. We then simulate each robot's movements according to the dynamics model $p(\chi_{s,\tau+1} \mid \chi_{s,\tau})$, producing locations at times $\tau = 2, \ldots, T$. Finally, we determine if robot s receives a distance reading from robot t according to $P_o(x_{s,\tau}, x_{t,\tau})$, and if so sample a noisy observation $d^{(i)}_{st,\tau} \sim p(d_{st,\tau} \mid x_{s,\tau}, x_{t,\tau})$.

As described in Sec. VI-A, we first apply phase I to localize robots at time $\tau = 1$, and then use phase II to track motion over time. Weighted marginal samples produce estimates $\hat{x}_{s,\tau}$, which we use to determine our estimation error. We run each experiment 10 times, and plot the mean of our error metric across these trials. In Fig. 1, we illustrate the connectivity from timesteps 1 and 4 for one trial, and the resulting discrepancy between the simulated (ground-truth) and inferred robot locations.

In Fig. 4, we plot mean error over each iteration and vary the amount of noise in the dynamics model, allowing either two or four NBP iterations per timestep. The "sawtooth" shape of the mean error plot is due to a new timestep every two or four iterations, depending on the experiment. The error is worst at the first iteration of the new timestep, and then lessens as more iterations, providing inter-distance information from further hops, refining the estimate. With only two iterations per timestep, there is more error and variability than with four iterations per timestep. Extra iterations incorporate spatial information from robots further away, and compensate for unpredictable dynamics models. However, when the dynamics model is very accurate, we need fewer iterations per timestep to converge to accurate estimates. While localization of the first timestep takes six iterations, for more precise dynamics models, we only need two iterations of our tracking algorithm, resulting in a 3x savings in computation and communication.

In Fig. 5, we vary the amount of noise in the inter-distance sensors, and test how estimation accuracy varies when many samples ($M = 500$) are available. NBP's estimates degrade gracefully as noise increases, with errors increasing approximately proportional to sensor noise. The right half of Fig. 5 uses inter-distance sensors with low noise levels ($\sigma_\nu = 10^{-3}$), and examines how errors vary with the number of samples M. As desired, using additional samples leads to more accurate estimates.

VIII. Comparison with Range-Based SMCL

We compare our NBP tracker to the range-based, sequential Monte Carlo localization (SMCL+R) tracking formulation of Dil et al. [1]. In the SMCL+R approach, distances are approximated by the shortest path distance on the connectivity graph from each beacon to each mobile robot. Hop distances are computed from each beacon to each robot, and all beacon locations are communicated to each robot. To localize robots at each timestep, they use a polar model similar to ours. Samples are drawn at uniformly random angles around each beacon, but using a deterministic distance as determined by the shortest path to the robot of interest. A subset of these samples are then rejected, if they are either too far from the previous iteration's estimate based on a maximum velocity model and constraints from the inter-distance sensors. More specifically, if a robot was a single-hop away, then samples should be within the maximum range, and if they were i hops away, the samples must be within i times the maximum range. Samples are drawn at each timestep for each robot, until M samples have been accepted. For further implementation details, see [1].

As shown in Fig. 6, the NBP tracker is 3-4 times more accurate than SMCL+R. However, the SMCL+R algorithm

is faster, with a complexity of $\mathcal{O}(nb^2M)$ per timestep, if there are n robots and b beacons. We suspect our superior accuracy is in part due to our explicit modeling of sensor errors, rather than just using observed distances.

In some experiments, due to the hop-based distance approximation and rejection of samples based on maximum transmission range, SMCL+R did not converge as it was impossible to draw acceptable samples. As the approach assumes a unit-disk connectivity model [15], this was exacerbated by probabilistic connectivity and inter-distance measurement models. For example, with a probabilistic connectivity model P_o, if a robot is within the maximum transmission range from a beacon, but is two hops away, the correct samples for the robot will always be rejected. Thus, the experiment in Fig. 6 used a unit-disk model for both approaches. By modeling probabilistic connectivity and inter-distance measurements, our approach is more robust as it does not experience to these problems.

IX. Conclusion and Future Work

We present a new algorithm for collaborative robotic self-localization and tracking using NBP. We compare the NBP dynamic tracking algorithm with SMCL+R, a sequential Monte Carlo algorithm [1]. Whereas NBP currently requires more computation, it converges in more cases and provides estimates that are 3 to 4 times more accurate. We are now working to reduce the computation time of the NBP algorithm.

The graphical modeling framework underlying our approach is very rich and we anticipate several extensions and generalizations. We hope to explore how orientation estimates can improve localization accuracy by modifying the proposal distribution of Eq. (16). Similarly, knowledge of environment geometry could be used to prune trajectory estimates that pass through obstacles. We are also working on "active" tracking where multi-modal estimates of robot position can be resolved by directing robots to move in directions that will reduce uncertainty.

References

[1] B. Dil, S. Dulman, and P. Havinga, "Range-based localization in mobile sensor networks," *Wireless Sensor Networks*, pp. 164–179, 2006.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, 1st ed. MIT Press, 2005.

[3] J. Pearl, *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[4] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2003.

[5] R. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the American Society of Mechanical Engineers, Journal of Basic Engineering*, pp. 35–46, March 1960.

[6] A. Ihler, J. F. III, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 4, pp. 809–819, April 2005.

[7] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2000.

[8] S. Thrun and Y. Liu, "Simultaneous localization and mapping with sparse extended information filters," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, July-August 2004.

[9] ——, "Multi-robot slam with sparse extended information filers," in *Proceedings of International Symposium of Robotics Research (ISRR)*. Springer, 2003.

[10] A. Howard, "Multi-robot simultaneous localization and mapping using particle filters," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[11] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *Proceedings of information processing in sensor networks (IPSN)*, 2006, pp. 27–33.

[12] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proceedings of Information processing in sensor networks (IPSN)*, 2006, pp. 34–42.

[13] P. Yang, R. Freeman, and K. Lynch, "Multi-agent coordination by decentralized estimation and control," *IEEE Transactions on Automatic Control*, vol. 53, no. 11, pp. 2480–2496, December 2008.

[14] P. Yang, R. A. Freeman, and K. M. Lynch, "Distributed cooperative active sensing using consensus filters," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 405–410, April 2007.

[15] K. Whitehouse, "Understanding the prediction gap in multi-hop localization," Ph.D. dissertation, University of California at Berkeley, 2006.

[16] Y. Shang, W. Ruml, Y. Zhang, and M. P. Fromherz, "Localization from mere connectivity," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing*, 2003, pp. 201–212.

[17] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *Proceedings of INFOCOM*, 2004, pp. 2652–2661.

[18] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *ACM Workshop on Wireless Sensor Networks and Applications*, 2003, pp. 112–121.

[19] A. Das, J. Spletzer, V. Kumar, and C. Taylor, "Ad hoc networks for localization and control," in *Proceedings of Conference on Decision and Control*, 2002.

[20] O. L. Moses and R. Patterson, "Self-calibration of sensor networks," in *Proceedings of SPIE on Unattended Ground Sensor Technologies and Applications IV*, vol. 4743, Orlando, FL, April 2002.

[21] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[22] B. Sohn, J. Lee, H. Chae, and W. Yu, "Localization system for mobile robot using wireless communication with ir landmark," in *Proceedings of Robot Communication and Coordination (ROBOCOMM)*, 2007, p. Article No. 6.

[23] C. Rohrig and S. Spieker, "Tracking of transport vehicles for warehouse management using a wireless sensor network," in *Proceedings of Intelligent Robots and Systems (IROS)*, September 2008, pp. 3260–3265.

[24] S. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, Oct 2002.

[25] A. Howard, M. J. Mataric', and G. S. Sukhatme, "Team localization: A maximum likelihood approach," University of Southern California, Tech. Rep., 2002.

[26] J. geun Park, E. D. Demaine, and S. Teller, "Moving-baseline localization," in *Proceedings of Information Processing in Sensor Networks (IPSN)*, 2008, pp. 15–26.

[27] J. Schiff, D. Antonelli, A. G. Dimakis, D. Chu, and M. J. Wainwright, "Robust message-passing for statistical inference in sensor networks," in *Proceedings of Information Processing in Sensor Networks (IPSN)*. New York, NY, USA: ACM, 2007, pp. 109–118.

[28] M. I. Jordan, "Graphical models," *Statistical Science*, vol. 19, no. 1, pp. 140–155, 2004.

[29] B. Silverman, "Density estimation for statistics and data analysis," *New York: Chapman and Hall*, 1986.

[30] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. John Wiley & Sons, 2001.

[31] A. Doucet, S. Godsill, and C. Andrieu, "On sequential monte carlo sampling methods for bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, September 2000.

[32] A. Ihler, E. Sudderth, W. Freeman, and A. Willsky, "Efficient multiscale sampling from products of gaussian mixtures," in *Proceedings of Neural Information Processing Systems (NIPS)*, Dec. 2003.