Efficient Node Self-Localization in Large Ad-Hoc Wireless Networks Using Interlaced Particle Filters

Benjamin R. Hamilton, Xiaoli Ma,
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA 30332

Robert J. Baxley
Information and Communications Laboratory
Georgia Tech Research Institute
Atlanta, Georgia, USA 30318

***Abstract*—Large self-organizing networks of wireless devices have shown potential in many emerging applications. While knowing the physical location of devices in enhances and enables many such applications, the locations of some or most of the wireless devices may be unknown and the devices may be mobile. Several recent works have examined using distance or RSS measurements to estimate the position of such wireless devices.**

In this work we examine methods to efficiently estimate and track the positions of a subset of mobile wireless nodes in the network and consider how well performance of these algorithms scale with network size. We consider position estimate error and convergence rate of these methods as the size of the network increases. We use simulations to show that a new localization method using interlaced particle filters converges faster and has higher estimation accuracy than existing algorithms.

I. Introduction

Self-organizing ad hoc networks promise a revolution in communications with increased flexibility and reliability and reduced deployment and maintenance costs. These networks also show promise not only in civilian applications ranging from intrusion detection to environmental monitoring, but also have military applications such as monitoring friendly forces, providing surveilance of both opposing forces and terrain, and detection of chemical, biological or radiological hazards [1]. Most of these applications require the position of wireless nodes in the network to be known, but traditional techniques such as GPS may not be available due to cost or environmental effects ranging from obstructions to hostile jamming, so the ability of such networks to self-localize is essential.

Recent works have focused on developing localization techniques which use angle, distance, delay or received signal strength (RSS) measurements for localizing nodes in the network [2]–[7]. These works often assume a subset of nodes called "reference" nodes know their own position and use this information along with the measurements to estimate the location of the other, "floating," nodes in the network. Reference nodes are assumed to obtain their position estimates from some external source such as GPS. Since measurements other than RSS require specialized hardware, techniques using RSS measurements have been preferred.

There are several proposed methods that use distance or received signal strength (RSS) measurements to determine node location. Multilateration [5], [8] uses the observation update portion of a distributed extended Kalman filter (EKF) to estimate node locations. A method using Semidefinite Programming [6], [9] has been proposed. This method tries to be scalable by dividing the problem among clusters, but scales poorly when the number of reference nodes is small or the reference nodes are concentrated on the edges of the network. A method in [10] used distributed particle filters to perform localization.

In this work we consider the scalability of localization algorithms as the number of floating nodes become large. We introduce a new localization algorithm using interlaced particle filters, and, through simulation, compare its performance to methods using both distributed Kalman filters, as in [5], [8], and interlaced Kalman filters, as well as the distributed particle filter from [10]. We show that the presented interlaced particle filter has both a higher convergence rate and lower mean-squared error (MSE) than existing methods.

II. System Model

We consider a network consisting of N_r reference nodes and N_m floating nodes, either of which can be mobile. These nodes are distributed uniformly in a two dimensional region such that the density of floating nodes is D. All nodes are assumed to have a maximum transmission range R.

A. Movement Model

The nodes in the network move according to one of the many movement models presented in the literature [11]. While we only specifically address two movement models in this paper, the presented algorithms can be adapted to any desired mobility model. We consider both a simple AR position model which we will use to derive our localization algorithms, and a somewhat more realistic random velocity model which we use in simulations.

In the AR position model, node position is modeled as an autoregressive (AR) random process. The position is assumed to have a state equation such that $\boldsymbol{s}[t+1] = \boldsymbol{s}[t] + \boldsymbol{\psi}[t]$, where $\boldsymbol{s}[t]$ is the position at time t and $\boldsymbol{\psi}[t]$ is a zero mean Gaussian random noise vector with covariance $\sigma_\psi^2 \boldsymbol{I}$.

The random velocity model is similar to the random waypoint model [11]. Since the random waypoint model has been shown to suffer from speed decay, where the average speed of nodes approaches 0 as time progresses, we instead use a random velocity model. In the random velocity model, mobile

978-1-4673-0081-0/11/$26.00 ©2011 IEEE

nodes select a random direction ϕ (from $-\pi$ to π radians), speed $0 \leq |\boldsymbol{v}| \leq 1$, and travel interval $0 \leq \tau \leq 1$ from suitable uniform distributions. Nodes travel at the chosen velocity for the travel interval and then choose another random velocity. Nodes that hit the bounds of the network immediately choose a new random velocity.

B. Measurement Model

We assume that nodes are able to obtain a measure of the distance to each of their neighbors. We consider two measurement models: one where distance is measured, and another that uses the received signal strength.

1) Using distance measurements: In the first model, the node is assumed to simply record a noisy measurement of the distance: $\hat{d} = d + \gamma$, where $\hat{d}$ is the measurement, d is the actual distance, and γ is Gaussian distributed noise. The distance for a given link is calculated as the euclidean distance between the nodes involved: $d_{nm} = \sqrt{(s_n^x - s_m^x)^2 + (s_n^y - s_m^y)^2}$, where $\boldsymbol{s}_n = (s_n^x, s_n^y)$ is the position of node n.

2) Using RSS as a proxy for distance: Most common devices do not have specialized hardware capable of directly determining distance. Instead, the distance is usually calculated from the received signal strength (RSS).

The RSS measures the received signal magnitude. We model this received signal magnitude as a pure function of distance disturbed by additive Gaussian noise with zero mean and variance σ_γ^2. So the RSS can be calculated as

$$\hat{z} = d^{-\alpha} + \gamma, \tag{1}$$

where α is the distance attenuation exponent. Note that the distance measurement model described previously can be thought of as simply a special case of this model with $\alpha = -1$.

The RSS strength measurement can used for localization in one of two ways, either First the location can be estimated from the RSS directly or the RSS can be converted to an equivalent distance measurement. Using the RSS measurement directly has the advantage that disturbing noise is simply the additive noise γ. Since γ is zero mean and Gaussian, the least-squares (LS) estimate of the position will be unbiased. On the other hand, localization using RSS measurements may be unstable do to the highly nonlinear relationship between the position and RSS measurements. Converting the RSS measurements to distances, on the other hand, causes the noise to be non-Gaussian, non-zero mean and correlated with distance. This noise creates a bias in the distance, causing far away nodes to seem further away, but nearby nodes to appear nearer still. As a result, the LS estimate of the position based on these distances will be biased, but the bias will depend on the network topology. Despite this bias, algorithms using these distance estimates are preferable since they are more stable. For this reason, we will only use the second method in this work.

C. Location Estimation Models

In a centralized location estimation algorithm, a single Bayesian filter (such as a Kalman filter or particle filter) would use all the measurements between each of the nodes in the network to simultaneously update the position estimates of all of the floating nodes in each iteration. Consider a system with state $\boldsymbol{\chi}_t = [\boldsymbol{s}_0[t]^T \cdots \boldsymbol{s}_{N-1}[t]^T]^T$ at time t and measurements $\boldsymbol{\zeta}_t = [z_{nm}[t] | n, m \in \mathcal{L}[t]]$, where $\mathcal{L}[t]$ is the set of tuples n, m such that nodes n and m are neighbors at time t. This Bayesian update consists first updating the state probability distribution from time $t-1$ to t, and second of incorporating the likelihood of the current measurement:

$$\mathrm{P}\left\{\boldsymbol{\chi}_t | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\} = \mathrm{P}\left\{\boldsymbol{\chi}_t | \boldsymbol{\chi}_{t-1}\right\} \mathrm{P}\left\{\boldsymbol{\chi}_{t-1} | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\} \tag{2}$$

$$\mathrm{P}\left\{\boldsymbol{\chi}_t | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_t\right\} = \mathrm{P}\left\{\boldsymbol{\zeta}_t | \boldsymbol{\chi}_t\right\} \mathrm{P}\left\{\boldsymbol{\chi}_t | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\}. \tag{3}$$

This single filter is very flexible since it can take advantage of correlations between the node positions estimates, and can take into account how uncertainty in the position estimates can affect the measurement likelihood. This flexibility comes at a high cost, however, since it is not easily distributed and computational and memory requirements scale poorly with the size of the network. In this paper we instead consider two simplifications of this model which distribute the computation over each node in the network, allowing for a complexity that scales linearly with the number of nodes in the network. In both simplifications we will also assume that node movement is independent, so Eq. (2), simplifies to:

$$\mathrm{P}\left\{\boldsymbol{\chi}_t | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\} = \mathrm{P}\left\{\boldsymbol{\chi}_{t-1} | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\} \prod_m \mathrm{P}\left\{\boldsymbol{s}_m[t] | \boldsymbol{s}_m[t-1]\right\}. \tag{4}$$

In the distributed filter model, each node uses an independent recursive Bayesian filter to estimate its own position using only the position estimates and RSS measurements from its neighbors. This is equivalent to assuming that there is no correlation in the node position estimates and that the measurement likelihood does not depend on the neighbor's position uncertainty. More formally, each node n assumes

$$\mathrm{P}\left\{\boldsymbol{\chi}_t | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\} = \prod_m \mathrm{P}\left\{\boldsymbol{s}_m[t] | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\}$$

and $\mathrm{P}\left\{\boldsymbol{s}_m[t] | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\} = 1$ for $m \neq n$. The measurement update equation then becomes:

$$\mathrm{P}\left\{\boldsymbol{s}_n[t] | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_t\right\} = \mathrm{P}\left\{\boldsymbol{\zeta}_t | \boldsymbol{\chi}_t\right\} \mathrm{P}\left\{\boldsymbol{s}_n[t] | \boldsymbol{\zeta}_0 \cdots \boldsymbol{\zeta}_{t-1}\right\}. \tag{5}$$

This effectively treats each of the neighboring nodes as reference nodes. This generally will cause the filter to both underestimate the position uncertainty, and converge to the final solution more slowly. The advantage of this model is that only the node position estimates need to be exchanged between nodes. An extended Kalman filter based on this model has been proposed for localization [5], [8].

In the interlaced filter model, each node in the network estimates its own position based on both the position estimates of its neighbors and the uncertainty in these estimates. This simplification ignores the correlation in position estimates

between nodes in the network, but uses the neighbors' position uncertainty to determine the measurement likelihood. Formally, we assume

$$\mathrm{P}\left\{\boldsymbol{\chi}_t|\boldsymbol{\zeta}_0\cdots\boldsymbol{\zeta}_{t-1}\right\} = \prod_m \mathrm{P}\left\{\boldsymbol{s}_m[t]|\boldsymbol{\zeta}_0\cdots\boldsymbol{\zeta}_{t-1}\right\}.$$

The measurement update equation then becomes:

$$\mathrm{P}\left\{\boldsymbol{s}_n[t]|\boldsymbol{\zeta}_0\cdots\boldsymbol{\zeta}_t\right\} = \mathrm{P}\left\{\boldsymbol{\zeta}_t|\boldsymbol{\chi}_t\right\}\prod_m \mathrm{P}\left\{\boldsymbol{s}_m[t]|\boldsymbol{\zeta}_0\cdots\boldsymbol{\zeta}_{t-1}\right\}. \tag{6}$$

If the position estimate error can be assumed to be Gaussian, as is the case when the Kalman filter is employed, this uncertainty can be completely captured by its covariance. This sort of simplification has been described previously in the literature. The simplification of a single Kalman filter to a set of interlaced Kalman filters for general systems is described in greater detail in [12]. An extended Kalman filter based on this model was listed as a potential enhancement for the localization algorithm described in [13].

III. Localization with Interlaced Particle Filters

We propose a localization algorithm using interlaced particle filters. In this algorithm, each floating node uses a particle filter (PF) [14] to estimate its own location, and then all the nodes in the network broadcast both their most recent position estimate and the position estimate's covariance to all of their neighbors.

More explicitly, each floating node n independently forms an estimate of its own position $\boldsymbol{s}_n$. Each floating node has N_P particles which represent samples from its position distribution. Each particle has a position $\boldsymbol{s}_n^{(k)}$ and a weight $w_n^{(k)}$. The weight is proportional to the likelihood of that particle's position being correct. Floating nodes calculate an estimate of their locations $\boldsymbol{s}_n$ as the average of the locations of all of their particles, weighted by the particle weights:

$$\boldsymbol{s}_n = \sum_{k=0}^{N_P-1} \boldsymbol{s}_n^{(k)} w_n^{(k)}, \tag{7}$$

and the covariance of this estimate $\boldsymbol{C}_n$ as:

$$\boldsymbol{C}_n = \frac{\sum_{k=0}^{N_P-1} w_n^{(k)}(\boldsymbol{s}_n^{(k)} - \boldsymbol{s}_n)(\boldsymbol{s}_n^{(k)} - \boldsymbol{s}_n)^T}{1 - \sum_{k=0}^{N_P-1} {w_n^{(k)}}^2}. \tag{8}$$

At each measurement interval t, each floating node n receive beacons from each neighbor m containing their neighbor's current position estimate $\boldsymbol{s}_m$ and this estimate's covariance $\boldsymbol{C}_m$. The node also measures the RSS of the beacon as $\hat{z}_{\boldsymbol{s}_n,\boldsymbol{s}_m}$. The floating node first updates its particles according to the movement model. In the case of the AR position movement model described in Section II-A, the particles are each disturbed by a zero mean Gaussian random noise vector with covariance $\sigma_\psi^2\boldsymbol{I}$, and the weights are unchanged: $\boldsymbol{s}_n^{(k)}[t] = \boldsymbol{s}_n^{(k)}[t-1] + \boldsymbol{\psi}_n^{(k)}[t]$. The node then performs the measurement update.

For the measurement update, the node first calculates the expected measurement $\tilde{z}_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m}$ for each particle k. This expected measurement represents what the measurement would be if the node were actually located at the position of that particle. This expected measurement depends on the position of the neighboring node m, so some of the uncertainty in the neighboring node's position will propagate through to expected measurement. The measurement is a nonlinear function of position, so the actual effect of the position uncertainty cannot be found analytically. We use the Jacobian of the measurement function, as a first-order approximation to model the propagation of the position uncertainty to this expected measurement. This Jacobian is found to be:

$$\boldsymbol{j}_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m} = -\alpha(\boldsymbol{s}_n^{(k)} - \boldsymbol{s}_m)\tilde{z}^{\frac{\alpha+2}{\alpha}}_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m}. \tag{9}$$

Using this, we can approximate the variance in expected measurement due to the uncertainty in the position of neighboring node m as:

$$\sigma^2_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m} = \boldsymbol{j}^T_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m}\boldsymbol{C}_m\boldsymbol{j}_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m}. \tag{10}$$

The particle weights can then be updated as

$$w_n^{(k)}[t+1] = w_n^{(k)}[t]\prod_m q[n,k,m], \tag{11}$$

where $q[n,k,m]$ is the weight adjustment for particle k from the measurement from node m. This weight adjustment should be proportional to $\mathrm{P}\left\{z_{\boldsymbol{s}_n,\boldsymbol{s}_m} = \tilde{z}_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m}|\hat{z}_{\boldsymbol{s}_n,\boldsymbol{s}_m}\right\}$. Since we are assuming that both the measurement noise γ and the uncertainty in the expected measurement are Gaussian with the same mean, the weight adjustment $q[n,k,m]$ in Eq. (11) should be the Gaussian probability density function with variance $\sigma_\mu[n,k,m]^2 = \sigma_\gamma^2 + \sigma^2_{\boldsymbol{s}_n^{(k)},\boldsymbol{s}_m}$:

$$q[n,k,m] = \frac{1}{\sqrt{2\pi\sigma_\mu[n,k,m]^2}}\, e^{-\frac{\left(\hat{z}_{\boldsymbol{s}_n,\boldsymbol{s}_m} - \tilde{z}_{\boldsymbol{s}_n^{(k)},m}\right)^2}{2\sigma_\mu[n,k,m]^2}}. \tag{12}$$

After all of the measurements have been processed, it is necessary to renormalize the particle weights so that $\sum_{k=0}^{N_P-1} w_n^{(k)}[t+1] = 1$. Since particle filters require resampling periodically to prevent the weights of the majority of particles from degenerating to 0, each node will periodically resample the particles. During resampling the particles are converted to an approximate probability distribution and random values are then sampled from that distribution to serve as the new particles. We model the probability distribution as a sum of N_P Gaussians centered on each particle such that the distribution has the probability density function:

$$p_n(x) = \sum_{k=0}^{N_P-1} \frac{w_n^{(k)}}{\sqrt{2\pi\sigma_s^2}}\, e^{-\frac{1}{2\sigma_s^2}\left(x-\boldsymbol{s}_n^{(k)}\right)^2},$$

where σ_s^2 is a configurable spread parameter. After resampling, all the particles have weight $\frac{1}{N_P}$ and are random samples from this distribution (i.e. $\boldsymbol{s}_n^{(k)} \sim p_n(x)$). As in [14], resampling is triggered when the effective number of particles,

$$N_{\text{eff}} = \left(\sum_k {w_n^{(k)}}^2\right)^{-1},$$

falls below a set threshold, N_{thr}.

Initially, the particles are assumed to be uniformly distributed over the field with equal weight. This algorithm continues iteratively as often as necessary to achieve the desired accuracy and for as long as location estimates are required. A summary of this algorithm is shown in Algorithm 1.

Algorithm 1 Interlaced Particle Filter algorithm

```
At each node n:
Initialize particle positions and weights:
s_n^(k) ~ Uniform over network
w_n^(k) = 1/N_P
for each measurement period t do
  if Node n is a floating node then
    for each neighbor m do
      Receive beacon with s_m, ẑ_{s_n,s_m}, and C_m
      Form expected RSS measurement z̃_{s_n^(k),s_m} and expected variance σ_μ[n,k,m]^2
      Adjust weights using Eq. (11)
    end for
    Normalize weights
    Calculate effective number of particles N_eff.
    if N_eff < N_thr then
      Resample Particles:
      s_n^(k) ~ Σ_k N(s_n^(k), σ_s^2)
      w_n^(k) = 1/N_P
    end if
    Estimate position s_n = Σ_k w_n^(k) s_n^(k)
    Estimate position variance C_n with Eq. (8)
  else
    Obtain position estimate from external source
    Set position variance: C_n = 0
  end if
  Broadcast beacon with position estimate s_n and covariance C_n to neighboring nodes
end for
```

IV. Other Localization Algorithms

For comparison purposes we define several algorithms based on both the distributed and interlaced estimate models from Section II-C.

A. Distributed Extended Kalman Filter

We first define a localization algorithm which uses an extended Kalman filter at each node to localize that node relative to its neighbors. This can be seen to be a trivial extension of the Multilateration algorithm [5], [8], to allow it to deal with a network of mobile nodes.

The extended Kalman filter approximates the state and measurement distributions as Gaussian, and consists of two main parts: the state update and observation update. For the distributed localization problem, the state update merely consists of forming the prediction of the position at time t, $\boldsymbol{s}_n[t|t-1]$, from the position estimate at time $t-1$, $\boldsymbol{s}_n[t-1|t-1]$ and forming the covariance of the prediction, $\boldsymbol{C}_n[t|t-1]$ from the covariance of the estimate, $\boldsymbol{C}_n[t-1|t-1]$. At each time t, all the nodes in the network broadcast their predicted position $\boldsymbol{s}_n[t|t-1]$. Each node n measures the RSS of these broadcasts and uses these measurements in the observation update to form the position estimate $\boldsymbol{s}_n[t|t]$ and its covariance $\boldsymbol{C}_n[t|t]$.

In the case of the AR position movement model described in Section II-A, the state update consists of the following: the position is disturbed by a zero mean Gaussian random noise vector with covariance $\sigma_\psi^2 \boldsymbol{I}$. This leaves the prediction equal to the previous estimate, $\boldsymbol{s}_n[t|t-1] = \boldsymbol{s}_n[t-1|t-1]$, with its variance is adjusted as

$$\boldsymbol{C}_n[t|t-1] = \boldsymbol{C}_n[t-1|t-1] + \sigma_\psi^2 \boldsymbol{I}.$$

The observation update proceeds as in [5], [8]. Each node n uses the RSS measurements to form the measurement vector $\hat{\boldsymbol{z}}_n$, with elements $\hat{z}_{\boldsymbol{s}_n,\boldsymbol{s}_m}$, for each neighboring node m. From the measurement model in Section II-B, the covariance of the measurement vector is assumed to be $\boldsymbol{R}_n[t] = \gamma \boldsymbol{I}$. The node then uses the current position estimate to form the corresponding expected measurement vector $\tilde{\boldsymbol{z}}_n$, with elements $\tilde{z}_{\boldsymbol{s}_n,\boldsymbol{s}_m}$. The covariance of the expected measurement vector $\boldsymbol{S}_n[t]$ is assumed to be only due to the variance of the position prediction of this node. The Jacobian of the measurement function is used as a first-order approximation to model the propagation of this position uncertainty to this expected measurement, so it is found as:

$$\boldsymbol{S}_n[t] = \boldsymbol{J}_n[t]^T \boldsymbol{C}_n[t|t-1] \boldsymbol{J}_n[t], \tag{13}$$

where $\boldsymbol{J}_n[t]$ is a matrix of Jacobians of the measurement function for each neighboring node m, with columns $\dot{\boldsymbol{J}}_{\boldsymbol{s}_n[t|t-1],\boldsymbol{s}_m[t|t-1]}$, calculated as in Eq. (9). The Kalman gain is then found as

$$\boldsymbol{K}_n[t] = \boldsymbol{C}_n[t|t-1] \boldsymbol{J}_n[t]^H (\boldsymbol{R}_n[t] + \boldsymbol{S}[t])^{-1}, \tag{14}$$

and the position estimate and its covariance are found as:

$$\boldsymbol{s}_n[t|t] = \boldsymbol{s}_n[t|t-1] + \boldsymbol{K}_n[t](\hat{\boldsymbol{z}}_n - \tilde{\boldsymbol{z}}_n) \tag{15}$$

$$\boldsymbol{C}_n[t|t] = (\boldsymbol{I} - \boldsymbol{K}_n[t] \boldsymbol{J}_n[t])\, \boldsymbol{C}_n[t|t-1] \tag{16}$$

The filter is initialized based on the assumption that the nodes are uniformly distributed in the network. Ideally this would mean that all of the nodes would have an initial position estimate $\boldsymbol{s}_n[0|0]$ that was set to the center of the network. Unfortunately, the Jacobian is undefined when two nodes share the exact same position. For this reason, all of the nodes initial position estimate is disturbed by a small Gaussian offset so they do not all share the same initial position estimate.

B. Interlaced Extended Kalman Filter

We also define a localization algorithm which uses an interlaced extended Kalman filter [12]. The interlaced extended Kalman filter was initially proposed as a method to reduce the complexity of the extended Kalman filter by assuming portions of the state are uncorrelated and dividing the filter

into several smaller parallel filters which each act on a portion of the system state.

Since the motion of the individual nodes is independent, the only modification to the distributed EKF described in Section IV-A is the addition of an extra term in the calculation of the covariance of the expected measurement vector $\boldsymbol{S}_n[t]$ in Eq. (13):

$$\boldsymbol{S}_n[t] = \boldsymbol{J}_n[t]^T \boldsymbol{C}_n[t|t-1]\boldsymbol{J}_n[t] + \boldsymbol{Q}_n[t],$$

where $\boldsymbol{Q}$ is a diagonal matrix containing the effective variance in the observed measurement due to uncertainty in the position of this neighboring node. The elements in $\boldsymbol{Q}$ can be calculated as in Eq. (10), using the Jacobian from Eq. (9) with the current position estimates, $\boldsymbol{j}_{\boldsymbol{s}_n,\boldsymbol{s}_m}$. This sort of distributed localization algorithm was briefly mentioned as a potential enhancement in [13], but otherwise has not previously been described.

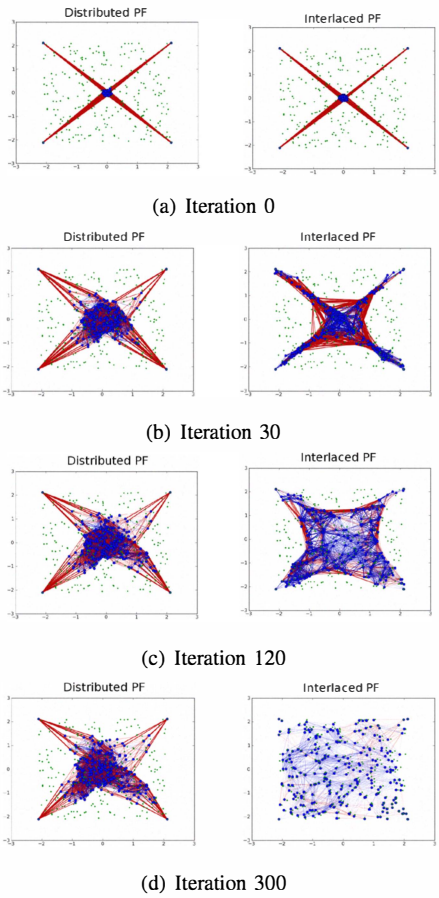
C. Distributed Particle Filter

We also define an algorithm for localization using a distributed particle filter. This filter is similar to that of the Interlaced Particle Filter described in Section III, only replacing the variance σ_μ^2 in Eq. (12) with $\sigma_\mu^2 = \sigma_\gamma^2$, since this variance is assumed not to depend on the uncertainty of the neighboring node's estimated position.

V. Performance Comparison

We compare the performance of the interlaced particle filter presented in Section III with the performance of the other algorithms defined in Section IV through simulation. We simulate a network of nodes in a square area with 4 reference nodes at each of the corners of the network area. Nodes were assumed to have a range of 1 unit, and the size of the network was adjusted such that the average node density was 14 nodes per unit area, or 14π neighbors per node. Nodes moved according to the random velocity model described in Section II-A, and simulated RSS or distance measurements were taken every 0.001 time units and used for localization. The simulation was run for 300 measurement intervals. The position MSE was measured at each interval, and then averaged over 50 simulations. The position MSE for both RSS and distance measurements were almost identical, so only the results for RSS measurements are shown.

Figure 1 shows an example comparing the distributed particle filter with the interlaced particle filter in a network of 250 nodes after 0, 30, 120 and 300 measurement and localization iterations. The reference nodes and floating nodes are represented by the large diamonds and circles, respectively. The current position estimate is shown by the smaller circles. The lines represent links between nodes, and the thickness of the line represents the magnitude of the difference between the true distance of this link and distance using the current position estimate. The two filters start with roughly the same initial estimates, as shown in Figure 1(a). In the distributed particle filter, measurements from all the neighbors are given equal weight, so the position estimates are "pushed" outward from the center in almost a random direction depending on the initial position estimate and into a local minimum. In contrast, the interlaced particle filter, nodes are almost unaffected by measurements to neighbors that have poor location estimates, so nodes directly connected to reference nodes are "pulled" toward them, forming a sort of 'X' shape. These effects are shown in the 30th measurement iteration, in Figure 1(b). In the distributed particle filter, the position estimates are slowly pulled to their proper position, but they have to go against the established local minimum, so progress is slow. In the interlaced particle filter, the position estimates for the nodes directly connected to reference nodes slowly improve, and begin to pull their neighbors, and their neighbor's neighbors towards the reference nodes. Eventually, nodes indirectly connected to reference nodes are pulled towards more than one reference node, causing them to be pulled towards a point between the reference nodes. This causes the position estimates to begin to fill out the 'X' shape. These effects are appear by the 120th iteration, shown in Figure 1(c). Eventually the interlaced particle filter converges to a good estimate of the nodes' position, while the distributed particle filter is still trying to escape from its local minimum, shown in Figure 1(d).



(a) Iteration 0

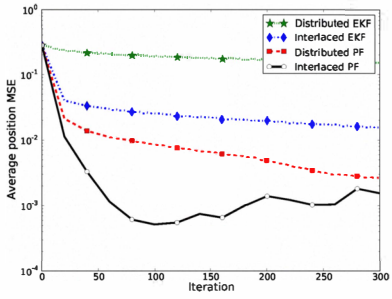
(b) Iteration 30

(c) Iteration 120

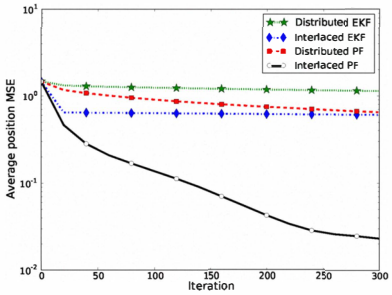
(d) Iteration 300

Fig. 1. Localization Example

The average position estimate MSE using RSS measurements for the first 300 localization iterations are shown in Figure 2. Figure 2(a) shows the position MSE for a network



(a) 50 nodes (RSS)



(b) 250 nodes (RSS)

Fig. 2. Position MSE convergence

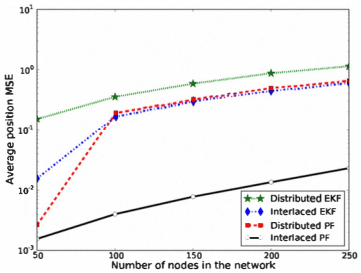


Fig. 3. Position MSE after 300 iterations (RSS)

of 50 nodes using RSS measurements. These figures show that the algorithms using particle filters have much lower position MSE than the algorithms based on Kalman filters. This effect is expected, since the particle filters are more accurately able to match the true probability distribution of node positions. Indeed, the distributed EKF offers only a negligible improvement over its initial guess. The distributed particle filter has an initial rapid improvement in MSE, but then only offers slow improvements over each iteration as it tries to escape its local minimum. The interlaced particle filter has a position estimate MSE that is strictly less than that provided by the other algorithms, and quickly converges to its minimum MSE.

In larger networks, these results change. Figure 2(b) shows the corresponding MSE for networks of 250 nodes. The distributed particle filter doesn't seem to converge. With such a large number of nodes, it gets stuck in a local minimum at a much higher MSE, and escapes it much more slowly. For this reason, the two interlaced filters converge much faster than the distributed methods in larger networks. The interlaced particle filter both converges much faster and has a lower steady-state MSE than the other methods.

If we only consider how the position estimate MSE scales with network size, the interlaced particle filter also outperforms the other methods. The position estimate MSE using RSS measurements after 300 iterations is shown in Figure 3. In this figure, the position MSE achieved by the interlaced particle filter increases much slower with network size than the other methods.

VI. Conclusion

In this paper we presented a new algorithm for localization in large multi-hop wireless networks using interlaced particle filters. We used simulation to compare this algorithm to similar algorithms and examined the convergence rate and position estimate MSE as the size of the network increases. These results show that our interlaced particle filter localization algorithm offers a substantially faster convergence rate and lower steady-state MSE compared to existing methods.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[2] N. Patwari, J. N. Ash, S. Kyperountas, A. O. I. Hero, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[3] G. Mao, B. Fidan, and B. D. Anderson, "Wireless Sensor Network Localization Techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[4] I. Guvenc and C.-C. Chong, "A Survey on TOA Based Wireless Localization and NLOS Mitigation Techniques," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 107–124, 2009.

[5] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proc. ACM WSNA*. New York, NY, USA: ACM, 2002, pp. 112–121.

[6] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," in *Proc. IEEE/ACM IPSN*, Apr. 2004, pp. 46–54.

[7] V. Seshadri, G. V. Zaruba, and M. Huber, "A Bayesian sampling approach to in-door localization of wireless devices using received signal strength indication," in *Proc. IEEE PerCom*, Mar. 8–12, 2005, pp. 75–84.

[8] A. Savvides, H. Park, and M. B. Srivastava, "The n-hop multilateration primitive for node localization problems," *Mobile Network Applications*, vol. 8, no. 4, pp. 443–451, 2003.

[9] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang, "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *IEEE Trans. Autom. Sci. Eng.*, vol. 3, no. 4, pp. 360–371, Oct. 2006.

[10] B. R. Hamilton, X. Ma, and R. J. Baxley, "SAL: Shadowing assisted localization," in *Proc. SSP*, Jun. 2011.

[11] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, Sep. 2002.

[12] L. Glielmo, R. Setola, , and F. Vasca, "An interlaced extended kalman filter," *IEEE Trans. Autom. Control*, vol. 44, no. 8, pp. 1546–1549, Aug. 1999.

[13] B. R. Hamilton, X. Ma, R. J. Baxley, and B. Walkenhorst, "Node localization and tracking using distance and acceleration measurements," in *Proc. CIP*, Jun. 2010.

[14] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, 2002.