Motion-aided network SLAM with range

Joseph Djugash and Sanjiv Singh

The International Journal of Robotics Research 2012 31: 604
DOI: 10.1177/0278364912441039

The International Journal of
Robotics Research
31(5) 604–625

DOI: 10.1177/0278364912441039
ijr.sagepub.com

Motion-aided network SLAM with range

Joseph Djugash and Sanjiv Singh

Abstract

A key problem in the deployment of sensor networks is that of determining the location of each sensor such that subsequent data gathered can be registered. We would also like the network to provide localization for mobile entities, allowing them to navigate and explore the environment. In this paper, we present a thorough evaluation of our algorithm for localizing and mapping the mobile and stationary nodes in sparsely connected sensor networks using range-only measurements and odometry from the mobile node. Our approach utilizes an extended Kalman filter (EKF) in polar space allowing us to model the non-linearity within the range-only measurements using Gaussian distributions. Utilizing the motion information from a mobile node, we show additional improvements to the static network localization solution. In addition to this centralized filtering technique, an asynchronous and decentralized approach is investigated and experimentally proven. This decentralized filtering technique distributes the computation across all nodes in the network, leveraging their numbers for improved efficiency. We demonstrate the effectiveness of our approach using simulated and real-world experiments in challenging environments with limited network connectivity. Our results reveal that our proposed method offers good accuracy in these challenging environments even when little to no prior information is available. Additionally, it is shown that by initializing the network map with a static network solution, the network mapping with a mobile node can be further improved.

Keywords

localization, mapping, SLAM, range only, ranging radio

1. Introduction

The growing trend of wireless communication and sensing technologies have emphasized the importance and applicability of sensor networks for a wide variety of application domains. Here we focus on the problem of estimating the node positions of a sensor network given only range data between nodes in the network. Previous work has shown that, given sufficient connectivity between the nodes in the network, it is possible to acquire an accurate estimate of the node positions (Djugash et al., 2006, 2008). However, existing strategies assume fairly simplistic scenarios, where the nodes operate within a large open area with little impedance to their measurements. In contrast, we explore more challenging and realistic scenarios, where the network is sparsely connected, has noisy measurements and has mobile nodes.

In many cases, the nodes have too few neighbors and their locations cannot be determined uniquely even in the ideal case with zero noise. In these cases, a mobile node with odometry can make a big difference in improving the network mapping accuracy. The significance of incorporating the odometry information from a mobile node is examined by comparing the 2D map reconstruction of an environment with a laser scanner given different position estimation methods.

The problem at hand is similar to the well-known SLAM (simultaneous localization and mapping) problem, where one or more mobile node(s) must localize its own position while simultaneously mapping the positions of other stationary nodes in the network/environment. As the size of the networks grow too large, the task of collecting all of the observations at a central location to perform the computation becomes impractical. Since observations are

Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA

Corresponding author:
Joseph Djugash, Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA.
Email: josephad@ri.cmu.edu

distributed across the network, nodes must coordinate to incorporate each other's observations and update their estimate in a decentralized and distributed manner. It is desirable that any such decentralized algorithm is also asynchronous, so that the nodes in the network can operate independently without the need to coordinate/synchronize their communications.

In this article, we build upon our previous work to develop a comprehensive method that solves the network SLAM problem and is robust to issues of noise, sparse connectivity and non-linear measurement distributions. In addition, we propose the use of an asynchronous belief propagation (BP) algorithm, more commonly known as 'loopy' BP. Unfortunately, the traditional loopy BP algorithm is only guaranteed to converge on network graphs without cycles. Thus, we present an extension that specifically addresses this limitation by first reducing any arbitrary graph into a minimum depth spanning tree which provides an approximate solution with guarantees on its convergence. This modification to the loopy BP algorithm is shown to provide good results comparable with the centralized solution on a variety of networks.

Experimental results on data that are particularly challenging to range-only estimation due to the amount of measurement noise, multi-path and outlier measurements are also presented. The experiments were conducted outdoors, with the nodes distributed around several buildings with no clear line of sight between many of the nodes. Simulated experiments are also presented to demonstrate the proposed method on much larger networks.

Our experiments address two distinct cases of network SLAM. First, the 'static' network mapping problem is tackled. Here, it is assumed that the nodes are sparsely connected and remain stationary. The particular challenge in this case is to accurately model the non-linear uncertainty distributions that arise while estimating a sparsely connected node's position in the presence of noise (such as multi-path or other outlier measurements). Next, we extend the static network mapping problem to deal with moving nodes. This general problem of localizing a network of both stationary and mobile nodes is called the *network SLAM* problem. The addition of a mobile node provides further constraints to overcome the ambiguities due to sparse connectivity between the nodes. However, the challenge here is to actively fold in the measurements and noisy odometry from the mobile node while maintaining an accurate estimate of the non-linear uncertainty distributions of all of the nodes' positions. We show through our experiments that the proposed method is robust, reliable and accurate even in challenging environments with many obstacles that restrict line of sight between the nodes in the environment. In addition, it is shown that when a mobile robot is introduced to a static network, and the static mapping solution used to initialize SLAM, a significant reduction in the node mapping errors is achieved.

2. Related work

In sensor networks the problem of mapping the locations of all of the nodes in the network is also known as self-calibration or self-localization. Most sensor networks are capable of measuring relative bearing, range or in some cases both range and bearing between nodes within the environment. Of particular interest to us are those that use range to localize the network. For instance, the RADAR system, developed by Bahl and Padmanabhan, utilizes signal strength of packets in the commonly available 802.11b wireless networks for localization of network devices (Bahl and Padmanabhan, 2000). However, signal strength measurements are often erratic and can be affected by slight changes in the environment. Alternately, fixed ultrasound emitters and embedded receivers have also been used to measure range between nodes in a network (Priyantha et al., 2000; Smith et al., 2004; Djugash et al., 2006). Eustice et al. (2011) also employ an acoustic modem-based communication and navigation system that in addition to computing pseudo-range measurements, using one-way travel time, also enables concurrent navigation of multiple underwater vehicles. Our system utilizes a new commercially available ranging radio system that uses radiofrequency (RF) signals to measure range between two nodes in the network (NanoTron, 2008).

Some of the early work in localizing a sensor network with range-only information relied on solving a least-squares optimization problem. Methods such as multi-dimensional scaling (MDS) provide a good solution if the network is fully connected (Borg and Groenen, 1997). For a less connected network with sufficient connections to provide 'rigidity' to the network, it is still possible to determine the map of the network. Moore et al. introduced the idea of the *robust quadrilaterals* as a way to avoid ambiguities in the solution (Moore et al., 2004). In practice, however, it is not easy to achieve a high degree of connectivity between nodes of the network.

Distributed and decentralized inference has received some attention in the literature. For example, particle filtering (PF) techniques have been applied to these settings: Zhao et al. (2003) use (mostly) independent PFs to track moving objects, and Rosencrantz et al. (2003) run PFs in parallel, sharing measurements as appropriate. However, Pearl's BP (Pearl, 1988) algorithm is one of the main methods for performing probabilistic inference. Pearl showed that the BP algorithm produces posterior probability distributions equivalent to centralized algorithms when run on networks without loops. One may try to run this algorithm on networks with loops, using multiple iterations of passing messages around the network. The resulting algorithm is called loopy BP. In loopy networks the beliefs are not guaranteed to converge, and if they do converge they might not converge to the correct posterior distribution. Nevertheless, empirical results have shown that in a large number of

cases, the algorithm converges to approximately correct posterior beliefs in a short amount of time (Pfeffer and Tai, 2005). Thus, loopy BP has emerged as one of the most competitive algorithms for approximate inference. Owing to this reason and the fact that loopy BP is an asynchronous algorithm, we adopt a modified variant of loopy BP algorithm to solve our decentralized network SLAM problem.

Ihler et al. (2004) extend the loopy BP framework to non-parametric belief representations (such as particle filters). While their approach lends itself to a distributed implementation and accommodates non-linear estimate distributions, it represents the state using samples, which could lead to divergence in the absence of sufficient samples. In addition, their approach assumes a negative information model to reduce the uncertainty and improve estimation, thereby reducing the number of samples necessary to achieve stability in their distributed implementation. In most real-world applications, however, the lack of a measurement could be due to a variety of reasons, making it difficult to accurately model negative information. In contrast, the proposed method scales well to large uncertainty distributions without the need to use negative information and produces good approximations of the centralized solution.

When implementing any distributed inference algorithm on a real system, it is crucial to note a few important criteria. Namely, it is necessary for any proposed algorithm to be decentralized and asynchronous. If a distributed inference problem is not decentralized, then there can exist a single point of failure (e.g. a 'master' node). In such a case, if a failure were to occur then the entire system would fail to function. Similarly, synchronicity introduces the requirement that every agent in the network coordinates their communication, which could add undesirable delays to the estimates reported by the nodes. Cao et al. (2006, 2008) among others have focused on this problem of achieving consensus between multiple agents in an asynchronous manner.

Another common approximate inference method is the Boyen–Koller algorithm (BK) (Boyen and Koller, 1998). In this approach, the state variables are factored into clusters. Instead of maintaining a joint distribution over all of the variables, distributions over the clusters are maintained. The joint distribution is approximated by the product of the cluster distributions. In essence, it entails performing BP on a modified graph called a junction tree. The basic premise is to eliminate cycles by clustering them into single nodes. Funiak et al. (2006) present one such approach of using the BK algorithm along with a junction tree decomposition to cluster the nodes.

Other researchers have also explored the distributed inference problem deriving motivation from other fields of research. Looking at the problem from the data fusion point of view, Makarenko and Durrant-Whyte have presented a Bayesian decentralized data fusion (BDDF) algorithm (Makarenko and Durrant-Whyte, 2004) that provides a convergent solution in a general Bayesian network. In the controls community, Yang et al. (2007) and others have looked at the use of a consensus filter to arrive at a consensus on a parameterized internal state, which can later be transformed to extract the filter state. Within the target tracking domain, Kamath et al. (2007) among others adopt the use of distributed algorithms designed to specifically assign roles for individual sensors that merge their local information in order to triangulate and estimate the position of targets within their sensor field of view (FOV).

While most research in sensor networks has focused on static nodes (network localization), work in SLAM has focused on the incorporation of motion from mobile robots into the estimation of static and mobile nodes. Olson et al. (2004) presented an extended Kalman filter (EKF)-based SLAM algorithm that reliably dealt with noisy measurements and required no prior information. Their method utilizes an initial pre-filtering step to approximately locate the landmarks/nodes thus making the linearization feasible. However, the performance of the pre-filter is highly dependent on its input data. Previously, we have presented a method based on an EKF that jointly estimates location of the static and mobile nodes (Djugash et al., 2006, 2008; Djugash and Singh, 2008). However, the experiments presented in these works were limited due to their assumption that the nodes are deployed within an open area with high connectivity between them and that there is little multi-path/outlier measurements. However, in most applications, it is rare that a sensor network can be deployed within an open area where line of sight between the nodes can be guaranteed or that the sensor measurements contain minimal erroneous measurements.

In this article, we extend our prior work to provide an asynchronous and decentralized algorithm that is designed to accurately model the non-linear and multi-modal distributions that naturally occur with range measurements. We demonstrate through several simulated and experimental results that the proposed approach is better suited to deal with sparsely connected and noisy networks deployed within an obstacle filled environment. The proposed decentralized implementation is shown to provide results close to the centralized solution with significantly reduced computation performed at each node in the network. Furthermore, we demonstrate through our experiments that incorporating a mobile node within an otherwise static network and initialing SLAM with the static mapping solution typically yields significant improvements to the network SLAM problem.

3. Technical approach

We model the network localization problem as a linear dynamical system. At each time step, t, the state of node i is represented by $X_{i,t} = [c_i^x, c_i^y, r_i, \theta_i]^{\mathrm{T}}$. Each node's estimate is represented in a polar coordinates (Djugash and Singh, 2008) (known as the Relative-Over Parameterization or ROP parameterization), where (c_i^x, c_i^y) are the center of

the polar coordinate frame and (r_i, θ_i) are the corresponding range and angle values. The use of this parameterization derives motivation from the polar coordinate system, where annuli, crescents and other ring-like shapes can be easily modeled. In addition, for each mobile node within the system, an additional term that represents the current heading of the node, ϕ_i, is also maintained within the state. The complete state vector at time t is represented as:

$$X_t = [X_{1,t}, \phi_1, \ldots, X_{M,t}, \phi_M, X_{M+1,t}, X_{M+2,t}, \ldots, X_{N,t}]^{\mathrm{T}}$$

where M is the number of mobile nodes and N is the total number of nodes. At each time step, we get some set of motion and range observations, u_t and z_t, respectively. The belief state at time t is defined as $p(X_t|z_{1:t}, u_{1:t})$. Our filtering algorithm iteratively computes the belief state at time $t+1$ using the previous belief state at time t. Specifically, in our implementation the belief state is represented by a mean vector μ_t and a covariance matrix Σ_t, and it is computed using an EKF.

When dealing with a mobile node, care needs to be taken to properly model the motion of the moving node. Whether odometry information is available or if a random walk model is assumed, it needs to be incorporated into the filter correctly (we refer the reader to our prior work for further details on improving the robot motion models (Djugash et al., 2009)).

3.1. Measurement model

When two nodes, i and j, are within a given range and sensor FOV to each other, a range observation is generated which is represented by $z_t^{i,j}$. This observation depends on the position of the two nodes i and j:

$$z_t^{i,j} = \hat{z}_t^{i,j}(X_{i,t}, X_{j,t}) + \delta$$

$$\begin{aligned} \hat{z}_t^{i,j} &= \sqrt{(m_{i,t}^x - m_{j,t}^x)^2 + (m_{i,t}^y - m_{j,t}^y)^2} \\ m_{k,t}^x &= c_{k,t}^x + r_{k,t} \cdot \cos(\theta_{k,t}) \\ m_{k,t}^y &= c_{k,t}^y + r_{k,t} \cdot \sin(\theta_{k,t}), \end{aligned} \tag{1}$$

where δ is zero-mean Gaussian noise and $(m_{k,t}^x, m_{k,t}^y)$ is the projection of the estimate for node k from the polar parameterization into Cartesian xy-space. The belief state is then conditioned on the observations of the current time step by computing

$$p(X_{t+1}|z_{1:t+1}, u_{1:t+1}) = \eta p(X_{t+1}|z_{1:t}, u_{1:t+1}) \cdot p(z_{t+1}|X_{t+1}), \tag{2}$$

$$p(z_{t+1}|X_{t+1}) = \prod_k p(z_{t+1}^k | X_{i \in g(z_{t+1}^k), t+1}), \tag{3}$$

where η is the normalization constant. The second term in the right-hand side of Equation (2) is the likelihood of the current observations. Equation (3) shows how this likelihood can be decomposed under the assumption that observations are independent given the locations of the nodes that made the observation. Note that each observation depends only upon the locations of the nodes in the set $g(z_{t+1}^k)$, which is the set of nodes that made the observation, and not the joint state vector. The range observations are augmented into the belief state by multiplying into the belief state a likelihood for each observation.

Upon the first observation of a particular node, the true distribution of the node is best represented as an annulus, see Figure 1(a). While an annulus is extremely non-Gaussian and difficult to model within the Cartesian xy-space, using the polar parameterization it is possible to approximate the annulus by an elongated Gaussian in polar coordinates ($r\theta$-space). This Gaussian approximation is given an arbitrary mean in θ (within the range $[0, 2\pi)$) with a large variance term, such that the probability along the θ dimension is near uniform, see Figure 1(b). Note that the Gaussian shown in the figure is an illustration and it does not represent any specific sigma contour of the Gaussian distribution. In practice, a Gaussian with variance 3.28 has been shown to provide a good approximate of the true uniform distribution in θ. Figure 1(c) shows the Gaussian ellipse (blue ellipse) overlaid on top of the true distribution (green shaded rectangle) in polar coordinates. By using this polar parameterization, a simple ellipse in polar coordinates transforms into an non-linear annulus when projected into the xy-space. It must also be noted that the elongated ellipse in the polar coordinate extends past the range of the true distribution. This extended tail of the Gaussian ellipse, when projected into the xy-space appears curled up within the estimated annulus, as can be seen in Figure 1(b).

3.2. Multi-hypothesis filter

Thus far, we have assumed a unimodal Gaussian model, capable of approximating the non-linearities within single range observations. We have also presented a probabilistic filtering method that is well suited for an EKF-based network localization system. While this approach deals with non-linearities of an annulus, it fails to adequately deal with the multi-modal distribution of the system (Figure 1(d)). Thus, whenever an annulus is split into separate modes, we simply duplicate the filter and adjust the mean of each filter to represent the two distinct intersection points. Then, by performing a measurement update using the new mean, we are able to appropriately update the covariance terms within the filter. The simple case of splitting a single annulus into two separate modes given a new range observation is depicted in Figure 1(d)–(f).

Given an annulus-like prior distribution, a new range observation that intersects the annulus at two distinct locations leads to a multi-modal distribution with two distinct modes (peaks/local maxima in the distribution). We refer to this as the *flip ambiguity* in range-only estimation tasks. This multi-modal distribution can be modeled using separate filters/hypotheses for each mode. To elaborate, whenever an annulus is split into separate modes, we simply

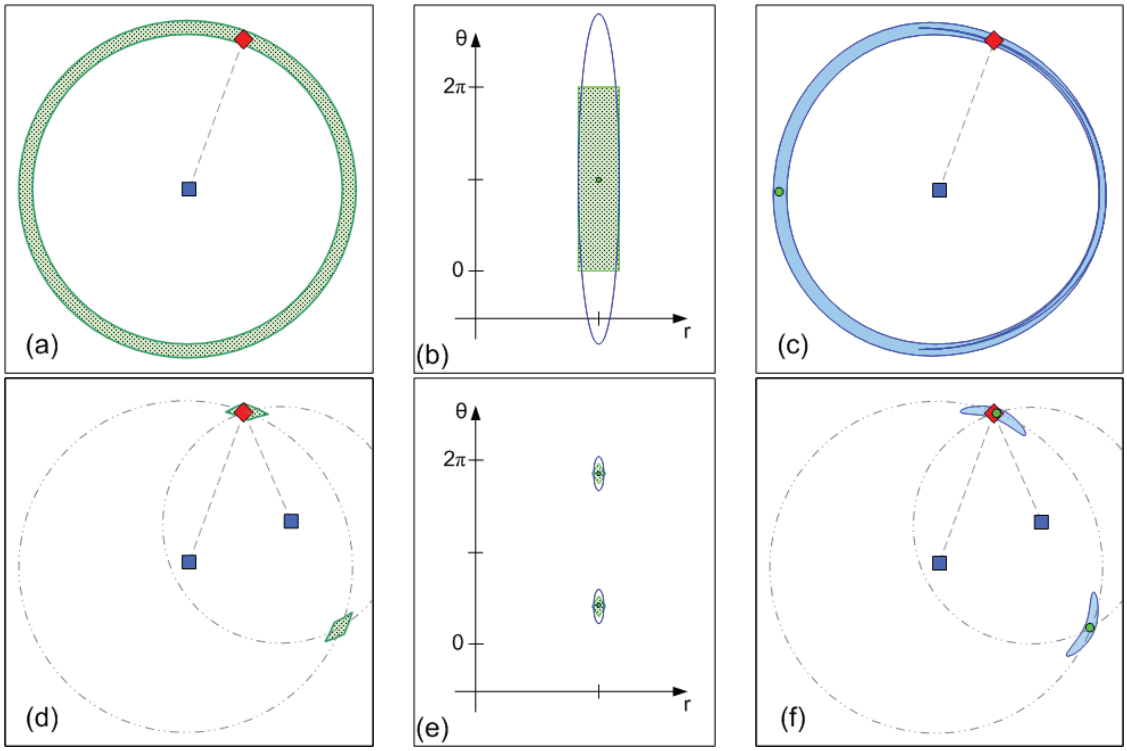

Fig. 1. The linearized uncertainty distribution using the ROP parameterization is shown along with its corresponding true and Cartesian projected uncertainty distributions for both the annulus and dual-modal estimate distributions. Blue squares represent *observing* nodes, whose location is known. Red diamonds represent the true location of the *observed* node/robot, whose position is being estimated and green circles represent the mean(s) for each mode of the estimated node. Green shaded regions (in (a), (b), (d), (e)) represent the true uncertainty distribution and blue ellipses (in (b), (c), (e), (f)) represent the estimated uncertainty distribution. The dashed gray lines and circles (in (c), (f)) represent the observed range measurements. (a) The true distribution (an annulus) of a single range observation. (b) The true distribution of an annulus (shaded rectangle), shown in polar coordinates, along with the unimodal Gaussian approximation (ellipse) of the true distribution. (c) The projection of the unimodal Gaussian ellipse from polar coordinates, shown in (b), to Cartesian *xy*-space. Note that the elongated Gaussian in polar coordinates, when projected into the *xy*-space maintains a tail (curled up within itself), which helps make the distribution uniform along θ (in the range $[0, 2\pi)$). (d) The true (dual mode) distribution of two unique range observations. (e) The true multi-modal distribution (shaded region) and its multi-modal Gaussian approximation (ellipse), shown in its native polar coordinates. (f) The projection of the multi-modal Gaussian approximation, shown in (e), into the *xy*-space.

duplicate the filter and adjust the mean of each hypothesis to represent the two distinct intersection points. It should be noted here that this duplication only occurs if the new measurement is 'novel', compared with the initial measurement that initialized the robot to an annulus-like distribution. Novelty of measurements, in this case, is directly correlated to the difference in the locations of the stationary nodes making the two range observations. Thus, if the distance between the node that made the new observation and the original observation is larger than a threshold, the new observation is used to generate the duplicate hypothesis. The threshold used here is set proportional to the sum of the measurement standard deviation and the standard deviation of the two observing nodes along the line connecting their mean. In practice, it was found that a threshold of 1.5 times the sum of the standard deviation provided good results. If the distance between the two nodes is less than this threshold, the measurement is used only to perform an EKF measurement update on the existing hypothesis.

Upon duplicating the filter and creating a second hypothesis, it is necessary to adjust the mean of both hypotheses. The new mean for each of the two hypotheses are calculated by triangulation, using the locations of the two stationary nodes that made the observation. Then, by performing a measurement update using the new mean, we are able to appropriately update the covariance terms within the filter. The simple case of splitting a single annulus into two separate modes given a new range observation is shown in Figure 1(d) and (e). Figure 1(f) shows the Gaussian ellipses (blue ellipses) for the dual modes overlaid on top of the true distribution (green shaded rectangles) in polar coordinates. The mean of the two modes can be determined easily using triangulation, given the location of the two observing nodes, as described by Faloutsos and Lin (1995). At the end of each update, we check the (normalized) likelihood of each hypothesis, given all of the measurements, and retain hypotheses with a (normalized) likelihood above a threshold (in practice using $1/(3 * NumofHypotheses)$

for the threshold provides good results). In addition, in our implementation, we remove any duplicate hypotheses. A hypothesis is considered duplicate, when it has a mean and covariance similar to another hypothesis. This can be checked using a distribution comparison metric such as the Kullback–Leibler (KL) distance. In addition, in the rare cases, where two hypotheses might have similar means but different covariances, the history of measurement residuals are used to prune the hypothesis with higher residuals in its recent history. Lastly, it should be noted here that in the localization case, the hypothesis count for the system will be never greater than two. Thus, implementing a multi-hypothesis filter is fairly straight forward and efficient. The complexity of adopting the multi-hypothesis filter in SLAM is discussed in the next section.

The ROP parameterization and multi-hypothesis filter proposed here are designed to accurately represent the non-linear distributions that are generated by range-only observations. However, it is important to remember that the distributions generated by the parameterization are still linearized versions of the true distributions. In other words, the proposed method, while capable of more accurately representing the non-linear distributions (such as an annulus or crescent), still uses a Gaussian distribution to approximate the true distribution. It is for this reason that when creating a second hypothesis, the mean of both hypotheses need to be adjusted. The adjustment, usually only in the θ_t^r parameter, highlights the point around which the linearization takes place. The exact adjustment to θ_t^r is derived directly from the two modes extracted from triangulation as described by Faloutsos and Lin (1995). Upon finding the position of two expected modes from triangulation, we simply compute the value for θ_t^r for each of the two hypothesis that will place its mean at the same location as the mode. In other words, we linearize the Gaussian around the two modes found through triangulation. Failing to properly adjust the means of the two hypotheses could cause the filter to take longer to converge or even diverge.

Figure 2 shows the general flow chart for the EKF measurement processing in our proposed ROP parameterization. Apart from special handling of a few initial measurements and the inclusion of a pruning step, the approach follows a similar flow to that of a standard EKF measurement update.

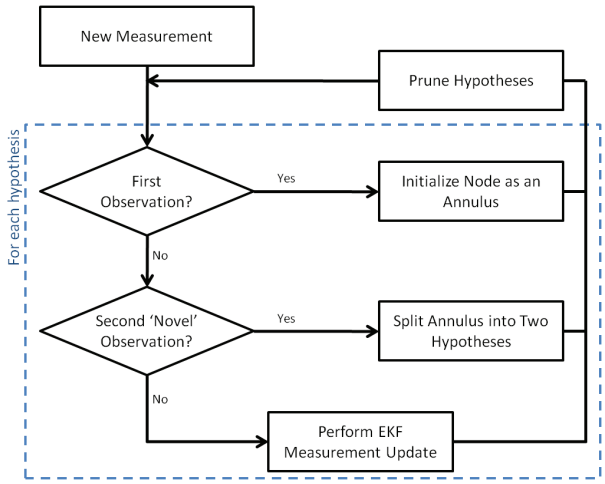

Fig. 2. Flow chart describing the procedure followed by the proposed ROP SLAM update step upon receiving a new measurement.

3.3. *Complexity of the multi-hypothesis filter*

Looking closer at the SLAM scenario, with specific focus on the creation of new hypotheses, we find that the number of hypotheses that can be generated by the filter is dependent on several factors. The most important of which is the number of nodes in the map. Based on what we observed with the localization problem, a single node in the map can generate a dual mode distribution as a result of the *flip ambiguity* discussed in Section 3.2. This in turn results in two hypotheses to be represented within the multi-hypothesis filter. As additional nodes are added to the map, each of those nodes would themselves generate a dual mode distribution, thus requiring two more hypothesis for each node. This implies that the multi-hypothesis filter needs to track *at least* $2N$ hypotheses, where N is the number of nodes in the map.

However, in the presence of inter-node measurements, it is easy to see that $2N$ is not the maximum number of hypotheses that need to be tracked by the filter. For example, let us first assume that the filter currently has an estimate of the robot's pose and one other node's estimate. This other node, which we will label 'node *A*', has a dual mode distribution. The ROP–EKF SLAM filter, presented above, stores this dual mode distribution with two hypotheses. At this point, if a new measurement to a previously unseen node *B* is observed by the robot, then this new node will be initialized to an annulus-like distribution within both the existing hypotheses (Figure 3(left)). Now, consider what happens next if node *A* observes a range measurement to node *B*. When the second range measurement to node *B* from node *A* is observed, the annulus-like distribution is split into a dual mode distribution (Figure 3(middle)). However, given that the filter already contains two hypotheses and each of those hypotheses have an annulus-like distribution for node *B*, each of those two hypotheses need to be split into two new hypotheses. In other words, the filter will now contain four hypotheses. Each hypothesis capturing one of the four combinations of permuting the two possible modes for each of the two nodes *A* and *B*. If a new node, 'node *C*', was observed by nodes *A* and *B*, a total of eight hypotheses are needed to properly represent all the likely modes of node *C*'s distribution (Figure 3(right)).

The example described above demonstrates the simple case, where adding a second node to the system grows the two hypothesis filter to a four hypothesis filter. It is easy to see the ramifications of this growth in the hypotheses.

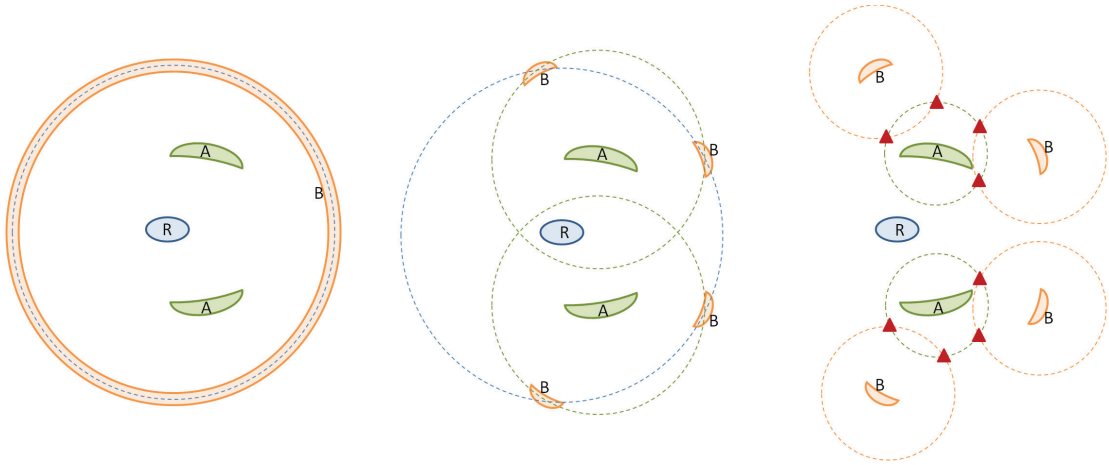

Fig. 3. (Left) Example illustration of node B's distribution upon processing the range measurement from the robot R. (Middle) Processing the range measurement between nodes A and B generates four unique modes/hypotheses for node B's position. (Right) Adding a new node 'C' that is observed by nodes A and B could double the number of existing hypotheses (the mean of the unique modes of node 'C' are shown by the red triangles).

Each new node added to the filter could possibly *double* the number of existing hypotheses. Formally, for each new node B added to the system, if there was Ω, number of modes/hypotheses, in the filter, then the observed node B can generate Ω additional hypotheses. Thus, for a N-node system (excluding the robot), the maximum number of hypotheses needed to capture all possible multi-modal distribution is 2^N. As might be expected, in an under-constrained system, where only sparse connectivity exists among nodes, the number of hypotheses needed to be represented by the filter will grow quickly. Furthermore, it should be noted that this bound is worse if we wish to merge two multi-hypothesis filters acquired by two different robots. In the worst case, given two robots (each with their own multi-hypothesis filters) with no commonly mapped nodes in their estimates, the maximum number of hypotheses required to fully capture the complete multi-modal distribution is $\Omega_1 \cdot \Omega_2$ (where Ω_1 and Ω_2 are the number of hypotheses maintained by each robot/filter).

It is easy to see that, in the worst case, this solution does not scale well to the addition of more nodes with sparse connectivity. Thus, intelligently deciding when to add new hypotheses and delete duplicate or unlikely ones could help limit the excessive growth of hypothesis count. In our implementation, at each iteration when the belief state is updated, we remove any duplicate/unlikely hypotheses with the pruning step described in Section 3.2. This pruning step is crucial to limit the number of hypothesis represented by the filter at any given time. In addition, it is also possible to delay the creation of new hypotheses by simply waiting until the hypothesis count of the filter is below some threshold (decided by the user to meet specific computational load) before a newly discovered node's estimate can be split, generating additional hypotheses. This approach is not ideal, because by delaying the 'splitting' of hypotheses, the 'current'-time estimate of the filter is less accurate.

Next, a decentralized approach that gathers the information from neighboring nodes in the network and merges them into each node's own belief using a BP algorithm is presented.

4. Decentralized estimation

Here we propose a simple scheme for formulating the estimation algorithm presented in the previous section in a decentralized manner. Let us assume that each node is able to share messages to its immediate neighbors (nodes that have connectivity to this node). In these messages, each node shares the part of its belief state that encodes information about its own estimate that is novel to each of its neighbors. Node i computes its belief at time t by merging its local observations (if any) with the messages from its neighbors, denoted Γ_i:

$$p(X_{i,t}|Z_t^i, m_{t-1}^i) = \alpha p(X_{i,t-1}|Z_t^i) \prod_{k\in\Gamma_i} m_{t-1}^{k,i} \tag{4}$$

where $X_{i,t}$ is the belief of node i at time t, Z_t^i is the set of measurements observed by node i at time t and α is the normalization constant (necessary to avoid the degenerate convergence to $\mathbf{0}$). Here m_{t-1}^i is the set of all messages $m_{t-1}^{k,i}$ to node i from nodes $k \in \Gamma_i$. The message $m_t^{i,j}$ from the node i to j at time t is computed by the marginal:

$$m_t^{i,j} = \alpha \int p(X_{j,t}|X_{i,t}) \left(p(X_{i,t-1}|Z_t^i) \prod_{k\in\Gamma_i\setminus j} m_{t-1}^{k,i} \right) dX_i \tag{5}$$

where $\Gamma_i \setminus j$ is the set of observed neighbors to node i excluding node j and $p(X_{j,t}|X_{i,t})$ represents node i's belief

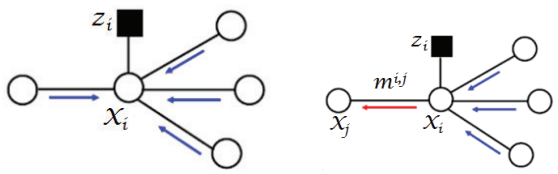

Fig. 4. (Left) An illustration revealing which messages (local measurements and messages from neighbors) are combined to generate each node's local belief $X_{i,t}$. (Right) An illustration revealing which messages are combined to form the marginalized belief sent to node j from node i in the message $m_t^{i,j}$.

of node j's position. To clarify the messages and their representation within our system, let use first take a look at Equation (5) in detail. A message from node i to node j is simply the marginal belief of node j computed with respect to node i's belief. To put it simply, each message contains the distribution that represents 'node i's estimate of node j's position'. Upon receiving each such message, each node updates its estimate based on the messages, before performing any future measurement updates as shown in Equation (4). This allows for proper flow of information through the network. The inherent distributed nature of this message-passing algorithm, lends itself to a decentralized implementation where the problem of global network localization is solved independently, in small parts, by each individual node. Figure 4 presents an illustration of what information and messages from neighbors are used to compute the local belief of node i, $X_{i,t}$, as well as what information and messages are encoded within a message from node i to node j, $m_t^{i,j}$.

The key difference in our approach, compared with standard loopy BP algorithm (originally presented by Pfeffer and Tai (2005)) is in the state that is being estimated by each node. It is common practice that each node in the network maintains and estimates a common state vector consisting of the positions of all of the nodes in the network. In this case, upon convergence, the state vector of each node will be identical and the computation and memory requirements would be no less than the centralized solution. This approach, also known as the 'trivially decentralized' method, is not practical due to the large memory and computational load it places on each node in the network. However, in our implementation, the state vector of each node is a subset of the full state vector and contains only the position of the node itself and its immediate one-hop neighbor's positions. In other words, node i maintains an estimate of node j in its state vector as long as there exists connectivity between node i and node j. Since connectivity in the graph is directly correlated with the presence of range measurement between the two nodes, it is straightforward to identify whether a given node's position is represented within another node's state vector. Upon convergence, node i's position will be known and stored within the state of itself and its immediate one-hop neighbors. An illustration of how our proposed decentralized loopy BP algorithm can better capture the cross-correlations represented within the centralized EKF is shown in Figure 5. As shown in the figure, the traditional approach only captures the strictly block diagonal elements of the centralized covariance matrix. However, our approach captures a much large portion of the covariance matrix, thereby achieving a better approximation of the centralized solution.

The belief maintained by each node is represented by a mean vector and covariance matrix. This belief is updated using an EKF and the motion and measurement models described earlier. Adopting this formulation, it is easy to see that the memory requirement on each node for maintaining only its own and immediate neighbor's position is considerably lower than the 'trivially decentralized' approach. However, one might consider going one step further and only storing each node's own position within its state vector. While this might seem to require the minimal amount of memory, it has a critical drawback.

Only storing each node's own position within its state vector, we fail to capture the correlations between the different nodes in the network. This will treat each node's position as completely independent of the positions of other nodes in the network. This approximation is not valid because it ignores the correlation between the nodes' position introduced by the range measurements. Failing to properly represent these correlations could yield a suboptimal solution in many cases. By maintaining all neighbor's estimate within each node's state vector, the information encoded within the cross-correlation terms of the covariance matrix in the EKF are not completely lost. By applying this extension, we not only gain a benefit in computation costs (as compared with the trivially decentralized approach) but the extra information encoded within the cross terms of the covariance matrix provides a better estimate of the true distribution.

One particular drawback of extending the belief of each node to include its neighbor's estimate is that in a fully connected network, the computational requirement for each node will be equivalent to running the centralized filter at each node. Fortunately, in most real-world applications, it is near-impossible to achieve a fully connected network. Even guaranteeing rigidity, which requires the average degree of connectivity to be four, is difficult and not always possible. Therefore, the decentralized filter presented here is suitable to most real-world applications where a high degree of connectivity between the nodes cannot be guaranteed.

Revisiting Equations (4) and (5), we can observe the following. Node i's belief can be written as follows:

$$p(X_{i,t}|Z_t^i, m_{t-1}^i) \sim \mathcal{N}(q_{i,t}, \Sigma_{i,t}) \tag{6}$$

$$q_{i,t} = \begin{bmatrix} q_{i,t}^i \\ \vdots \\ q_{i,t}^j \\ \vdots \end{bmatrix}, \Sigma_{i,t} = \begin{bmatrix} \Sigma_{i,t}^i & \dots & (\Sigma_{i,t}^{i,j})^{\mathrm{T}} & \dots \\ \vdots & \ddots & \vdots & \dots \\ \Sigma_{i,t}^{i,j} & \dots & \Sigma_{i,t}^j & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{7}$$

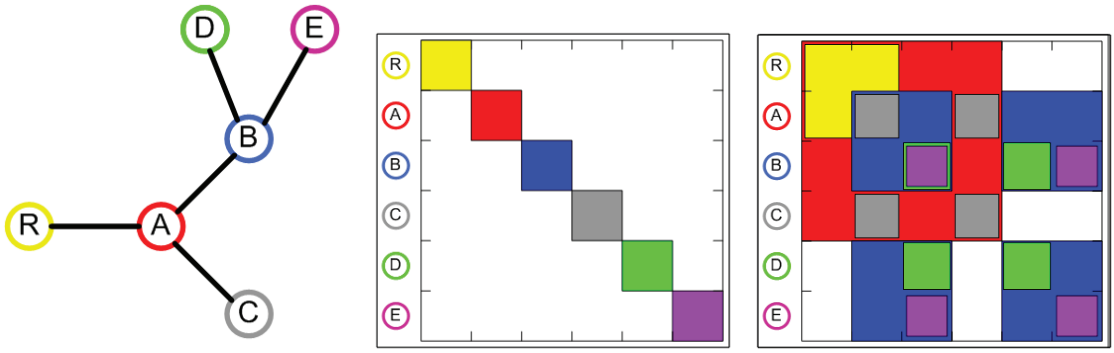

Fig. 5. An illustration highlighting the components of the 'full' covariance matrix represented in the centralized EKF that are also represented by each node within the proposed decentralized loopy BP algorithm. (Left) The graph of the network revealing the connectivity between the nodes is shown. (Middle) The traditional loopy BP approach, where each node only maintains an estimate of its own pose will result in the block diagonal covariance matrix shown. (Right) The covariance matrix for our proposed approach, which requires that each node maintains an estimate of both its own and its immediate neighbors' pose, is shown. The individually colored blocks reveal the component of the full covariance matrix that are represented within the local estimate at each node. For example, node A will have a local belief which contains an estimate of nodes A, R, B, C. Its corresponding covariance matrix is shown in 'red'. As can be seen, the union of each individually colored covariance matrices captures a bigger portion of the centralized equivalent covariance matrix than the traditional loopy BP approach.

where $q_{i,t}$ and $\Sigma_{i,t}$ are the mean and covariance of the belief maintained by node i and $q^j_{i,t}$ and $\Sigma^j_{i,t}$ are the mean and covariance of node j as estimated by node i for all $j \in \Gamma_i$. If we marginalize over node i, the message from node i to node j can be written as

$$\begin{aligned} p(X_{i,t}|Z^i_t, m^{i*}_{t-1}) &= \alpha p(X_{i,t-1}|Z^i_t) \textstyle\prod_{k\in\Gamma_i\setminus j} m^{k,i}_{t-1} \\ &\sim \mathcal{N}(q^*_{i,t}, \Sigma^*_{i,t}) \end{aligned} \qquad (8)$$

$$m^{i,j}_t \sim \mathcal{N}(q^{j*}_{i,t}, \Sigma^{j*}_{i,t})$$

where $q^*_{i,t}$ and $\Sigma^*_{i,t}$ are the mean and covariance of the belief maintained by node i given measurements from time t and messages from time $t-1$ from all neighbors except node j, $\Gamma_i \setminus j$. Therefore, $q^{j*}_{i,t}$ and $\Sigma^{j*}_{i,t}$ are the marginalized mean and covariance of node j as estimated by node i given messages from all nodes $k \in \Gamma_i \setminus j$. The messages shared between nodes are simply a mean vector and covariance matrix representing the position of node j as estimated by node i.

There are a few important things that must be clarified for implementation of the above-described loopy BP on real systems. First, in the above formulation, loopy BP necessarily iterates many times until the messages passed between nodes converge. To fully understand the limits of the system, further examination of the behavior in the case where convergence is not reached is necessary but this falls outside the scope of this article. In implementation, we simply enforce a limit in the number of iterations to ensure that the filter does not continue to oscillate forever in case of non-convergence. In addition, when implementing this algorithm on a real system, it is impossible to ensure that all possible measurements within the network can be observed at a single instant, time t. This makes it difficult to generate the graph needed to perform inference. In practice, in order to ensure seamless integration of the loopy BP algorithm with a real system, it is necessary to initially collect/gather measurements for a short period of time such that all (if not a majority) of measurements within the network can be observed at least once before attempting to share messages across the graph. After this initial phase, messages can be shared across the links of the graph without worry. In a dynamically changing network, this graph can be recomputed over a window, in the background, to ensure that any changes to the network graph is properly dealt with when performing loopy BP.

While the approach described here provides satisfactory results in most cases, there is no guarantee that this method will converge to the correct solution. In the loopy BP, the convergence of the belief is only guaranteed for trees. In other words, in the presence of loops (as is the case in most sensor network applications) some information can be counted twice, making it less likely to accurately converge to the centralized solution.

4.1. Convergence in belief propagation

The idea of propagating messages in loopy graphs was first proposed in the early days of the field, in parallel with the introduction of the first exact inference algorithms (Pearl, 1988). As was noted early in the field, when loops are present, the network is no longer singly connected and a local propagation scheme will invariably run into trouble. Ignoring the existence of loops and permitting the nodes to continue communicating with each other as if the network were singly connected, will cause messages to circulate indefinitely around the loops and the process may not converge to a stable equilibrium.

As a consequence of this, one of the main problems with loopy BP is non-convergence. Several approaches have been used for addressing this non-convergence issue. Some are fairly simple heuristics. A common observation with loopy BP is that, often, non-convergence is a local problem. In many practical cases, most of the beliefs in the network do converge, and only a small portion of the network remains problematic. In such cases, it is often quite reasonable simply to stop the algorithm at some point (for example, when some predetermined amount of time has elapsed) and use the beliefs at that point, or a running average of the beliefs over some time window. This heuristic is particularly reasonable when we are not interested in individual beliefs, but rather in some aggregate over the entire network (e.g. temperature estimation of a room with a network of temperature sensing nodes). However, this is not the case for our network localization problem, where we want all of the nodes' beliefs to converge.

It is for this reason, we turn to look at a variant of BP that operates on a tree, rather than a graph, and schedules messages in a synchronous and guided way to ensure proper convergence of the solution.

4.1.1. Synchronous belief propagation To better analyze the convergence property of Loopy BP we turn to a variant of BP called synchronous BP. The simplest form of BP is designed for the special case when the network graph is a tree (i.e. no cycles/loops). In this case the algorithm computes exact marginals, and terminates after two steps. Before starting, the graph is oriented by designating one node as the *root* node. Any non-root node which is connected to only one other node is called a *leaf* node. Each node in a tree has zero (i.e. leaf node) or more child nodes, which are below it in the tree. A node that has a child is called the child's parent node and a node can have at most one parent.

In the first of two steps, messages are passed inwards: starting at the leaves, each node passes a message along the (unique) edge towards the root node. The tree structure guarantees that it is possible to obtain messages from all other adjoining nodes before passing the message on. Therefore, each node waits to receive all messages from its child nodes before merging their messages with its own local measurements and then sending a message to its parent node. This continues until the root node has obtained messages from all of its child nodes. The second step involves passing the messages back out. Starting with the root node, messages are passed in the reverse direction. Each node, upon receiving a message from its parent node, then computes and sends a message to its child nodes. The algorithm is completed when all leaves have received their messages from their parent nodes.

The message structure remains the same as in loopy BP described above in Equation (5). A message from a node to its parent, in the first step, will consist of an estimate of the parent's position given any local measurements the node observed and the messages from its child nodes. Similarly, in the second step, a message from a node to its child nodes will consist of an estimate of the child's position given any local measurements the node observed, the message from the node's parent and the messages from its other child nodes. Adopting this strategy guarantees that upon completion of the two steps, all of the nodes in the graph will have converged to the correct centralized-equivalent solution.

Comparing synchronous BP described here and the loopy BP discussed earlier, we can see a couple of key differences. First, synchronous BP, as the name indicates, requires synchronization of the message passing while loopy BP is asynchronous in nature. This is an important feature because, in most real systems, it can be difficult to precisely synchronize messages without experiencing some uncharacterized delay. This delay can limit the use of such synchronous approaches in applications were synchronization cannot be achieved. Second, synchronous BP is limited to network graphs that are trees. While this might seem like a major drawback, it is precisely due to this limitation that synchronous BP is always able to guarantee convergence. Furthermore, as we noted earlier, a key drawback of loopy BP is that in graphs with cycles/loops the solution might not converge. In contrast, it can be shown that in a tree, loopy BP will converge to the same solution as synchronous BP within a number of iterations equal to the diameter of the tree. Therefore, it is clear that, to gain the best of both worlds, we must devise a strategy to reduce an arbitrary network graph into a tree to gain the same convergence property that synchronous BP provides while maintaining the same asynchronous nature of loopy BP.

4.1.2. Tree decomposition of graphs As is the case with most robotics or sensor networks applications, cycles/loops are fairly common in a network of sensor nodes. In these cases, as we have discussed, loopy BP only offers an approximation to the true solution with *no guarantees* on the convergence of the solution. On graphs with cycles, information from a node can loop back to itself resulting in 'double-counting' of some information. Not knowing what and how much information is ignored or 'double-counted', makes the use of loopy BP on graphs with cycles unreliable.

Figure 6 presents an example case of a graph with a cycle, where the arrows depict the information flow from the 'red' node. As can be seen, in a cycle the information sent by the red node loops back to itself causing it to be incorporated into the estimate a second time. This double-counting of information can lead the estimate to become over-confident in itself. A tree decomposition of the graph to produce a spanning tree will break the cycle(s) (by removing the dashed edge), thus, guaranteeing that no information is double-counted. This in turn improves the accuracy of the estimate.

When computing a spanning tree of a graph, it is necessary to first ask the following question. If it is necessary

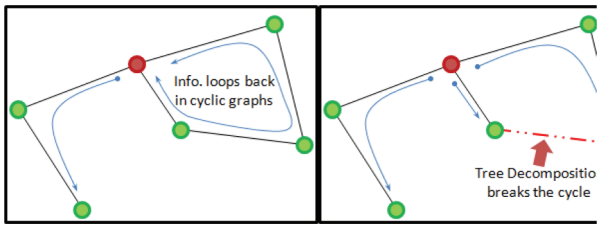

Fig. 6. An example of a graph with a cycle (left) and a sample tree decomposition of the graph (right). The green and red circle represent the nodes and the solid lines represent the edges in the graph. The arrows show the flow of information originating at the red node. In graphs with cycles (left), the information from the red node will loop around the cycle and arrive back to the red node causing it to double-count some information. A tree decomposition of the graph (right), achieved by removing the dashed edge, stops the looping of information, ensuring that no information is double-counted.

to break some edges in the graph, are some edges better to break than others? If so, how do we decide which edges are better to break? Chow and Liu (1968) present a metric by which the usefulness of any edge in the graph can be computed. By using such a metric coupled with an algorithm to compute the minimal spanning tree (such as Prim's algorithm (Prim, 1957)), it is possible to decompose a graph with cycles into a maximally informative tree sub-graph. However, while this offers a reasonable metric to compute the minimal spanning tree, computing the weight/usefulness of a given edge might prove to be expensive.

Remembering that the convergence time for loopy BP on a tree is related to the diameter of the tree, we propose to utilize a much simpler metric to compute the spanning tree of a graph. Since a quicker convergence time is always desirable, choosing a spanning tree that has the minimal diameter is ideal. Therefore, we adopt a distributed algorithm for computing the minimal diameter spanning tree proposed by Bui et al. (2004). At the start of the loopy BP algorithm, it is now necessary for us to compute the minimal diameter spanning tree using Bui et al.'s algorithm. Once the graph is reduced to a tree, loopy BP can be applied directly on the tree, as described in the previous section.

By adopting this strategy, it becomes possible to provide guarantees on the loopy BP's solutions even for graphs with cycles. It should be noted here that the graphs in our system are derived directly from the topology and connectivity of the sensor nodes in our network. In other words, the edges in our graph correspond directly to the range measurements observed within the network. Thus, it is highly likely that this graph might change if the network localization problem is performed over a period of time. If the initially computed minimal spanning tree is no longer valid because an edge in the network graph is no longer connected, then it is necessary to recompute the minimal diameter spanning tree for the new graph. However, in most sensor network and robotics applications, the network graph does not change every time step. Although in the worst case, it becomes necessary for us to recompute the minimal diameter spanning tree every time step, thus, adding extra computation to the algorithm.

Figure 7 presents the steps followed by each node in the network when the proposed asynchronous loopy BP algorithm is used. Note that the term 'local measurements' includes all range measurements observed locally by the node and any motion information available to the moving nodes. Messages received from neighbors in the graph are fused with each node's local belief through a standard EKF update, where the messages are treated as direct observations of the node's state. Thus, the measurement Jacobian matrix H in the 'message update' step is simply the identity matrix.

4.2. Communication requirements

Looking closer at the communication requirements of the proposed approach, it is easy to spot two areas where communication between the nodes occur (ignoring the process of acquiring the measurements themselves). The first communication requirement comes from the computation of the minimum diameter spanning tree (MDST). This requires the communication of a list of node IDs across the network with the list length proportional to the number of nodes in the network. While the need for computing the MDST is by itself additional load on the communication bandwidth, it is not necessary to compute the MDST each iteration. Furthermore, in a mostly static network, the connectivity of the network graph does not change very often. Thus computing the MDST occasionally is acceptable.

The second major communication requirement in the system is due to the *necessary* communication of the beliefs/messages between the nodes. Remembering that a message from node i to j is simply the mean and covariance matrix representing node i's estimate of node j's position; the size of each message is at most $5 + 25$ floating point numbers. For a given MDST with depth D and E edges, at most $4 * E * D$ messages need to be communicated to achieve convergence. Comparing this to performing loopy BP on the same network without first computing the MDST, we see that a total of $4 * M * T$ messages need to be shared instead. Here M is the number of edges in the network graph, with $M \geq E$, and T is the total number of iterations performed. In general $T \geq D$ also, since at least D iterations are needed for information from one end of the network to reach the other end. In the worst case, when $M = E$, the communication performed to compute the MDST (which is the graph itself) becomes unnecessary and wasteful of the communication bandwidth. However, it should be noted that without performing this additional communication it is impossible to guarantee the convergence of the loopy BP algorithm on a general graph. Thus,

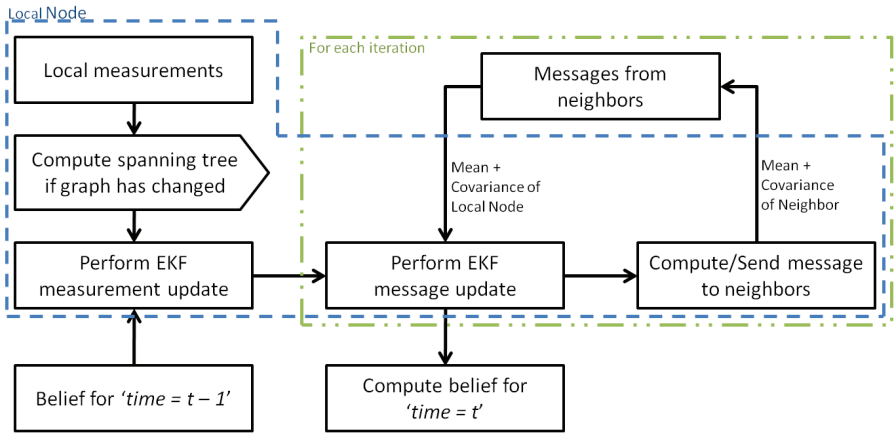

Fig. 7. Flow chart detailing the steps performed by each node at time step t of the proposed asynchronous loopy BP algorithm.

the number of iterations M could be very large or even infinite if convergence is never reached. Given this observation, the proposed approach, while at first glance may appear to require additional communication, on average requires less communication than performing loopy BP on the raw network graph. Further experimental testing is needed to thoroughly evaluate the communication bandwidth requirements of the proposed system on a large real-world sensor network.

5. Results

In this section, we evaluate the performance of the proposed network SLAM algorithm on a variety of networks. First, a detailed examination of the proposed loopy BP algorithm is done on a small 18-node network in simulation. A head-to-head comparison against the centralized and synchronous counterparts is also presented for this network. 'To test the scalability of our proposed algorithm to larger networks, we also present the results of our approach in simulation for 50-node and 100-node networks. Next, the proposed approach is evaluated using real-world experiments with both static networks and networks with a mobile node and the presence of occlusions in the environment introduce significant noise to the measurements. Finally, results of applying the loopy BP algorithm on a real-world sensor network are also presented.

5.1. Simulation results

Let us begin our evaluation of the proposed loopy BP algorithm by first looking at a fairly small network example in simulation. Figure 8 shows the results of running the loopy BP algorithm on an 18-node network. The gray connectivity graph shown in Figure 8(row 2, column 3) shows the true connectivity between the nodes. The graphs shown in each of the other frames, represents the minimal spanning tree used by the corresponding algorithm to share messages. In addition, nodes 1 and 2 are assumed to have absolute positioning capability. Thus, their positions are initialized accurately with low uncertainty. Figure 8(row 1) shows the estimate snapshots of the filter at several iterations (1, 3, and 5) of the loopy BP algorithm. At each iteration the loopy BP algorithm computes new messages to send to its neighbors based on the messages it received from its neighbors in the previous iteration. As information from nodes 1 and 2 propagates across the network, the estimates of the others nodes in the network converge. It should be noted here that while the estimates of some nodes appear to be 'missing' in some frame in Figure 8(row 1), this is not the case. Their estimates simply lack a global reference. Thus, we choose not to plot them for clarity of results. In reality, each node's assumes that it is initially at the origin of its local coordinate frame with large uncertainty. This initial belief is collapsed when information in the global coordinate frame (from anchor nodes 1 and 2) arrives to each node via messages from its neighbors. This results in the behavior observed in Figure 8(row 1).

Figure 8(row 2) presents the final 'converged' solution of the loopy BP algorithm, along with the result produced by the synchronous BP and centralized network localization methods. As can be seen, the loopy BP and synchronous BP algorithms approximate the centralized solution. In addition, note that the uncertainties in the result of the loopy BP and synchronous BP are lower than the centralized; indicating that the two BP algorithms are overconfident. This is due to the approximations in merging linearized beliefs from different nodes while utilizing the ROP parameterization. However, as can be observed this approximation still produces good results. Lastly, it should be noted here that the loopy BP algorithm only takes 10 iterations in this example to fully converge. This is equivalent to the diameter of the spanning tree used by the algorithms. Thus, for any arbitrary network, the number of iterations necessary for the loopy BP algorithm to converge will vary depending on the diameter of the resulting spanning tree.

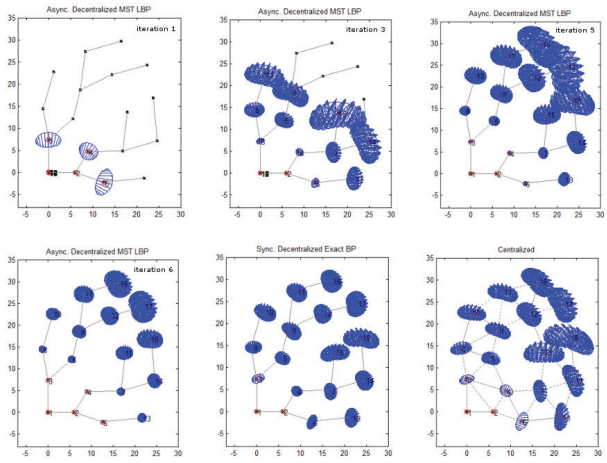

Fig. 8. An illustration of the decentralized loopy BP algorithm on a small 18 node network, compared against the centralized and synchronous BP methods. (Row 1) Shows a few intermediate steps (iterations 1, 3, and 5) of the proposed loopy BP algorithm. (Row 2) Shows the final loopy BP solution (column 1), along with the synchronous BP (column 2) and centralized network localization (column 3) results. The edges shown in centralized solution's plot shows all possible range measurements in the system while the edges in the other plots show the spanning along which messages are passed. In this experiment, the positions of nodes 1 and 2 are known initially (note that only one of the two possible solutions are shown here for clarity, the other solution is simply the 'flipped' about the line formed by nodes 1 and 2). The gray connectivity graphs shown in each frame indicate the graph along which messages are passed for the decentralized approaches. The gray connectivity graph shown in the last frame (row 2, column 3) represents the true connectivity of the network, which also indicates the presence of range measurements between the connected nodes. The loopy BP algorithm converges to the synchronous BP solution within a few iterations. The differences observed between the centralized and synchronous BP solutions are due to the approximation used in merging the beliefs from two different nodes (discussed earlier), each utilizing a different linearization point.

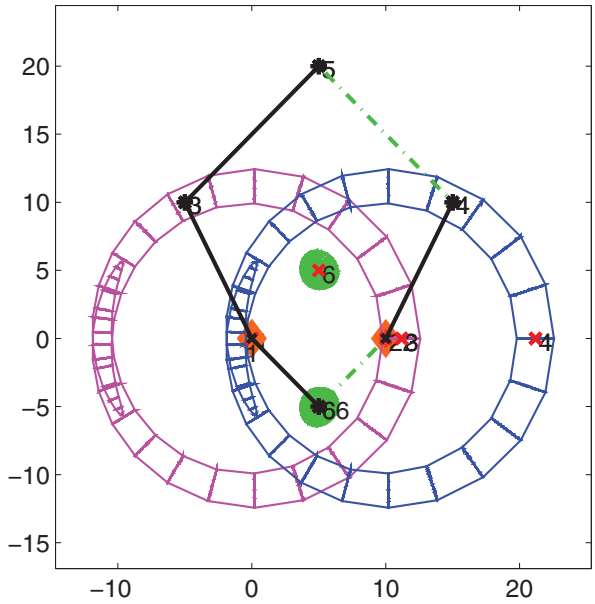

Fig. 9. Plot of the estimate for a simple five-node network with two anchors (nodes 1 and 2) [orange diamond] and a node connectivity of 2. The estimate of two of the nodes (nodes 3 and 4) remain annuli [magenta and blue] and their common neighbor (node 5) lacks a globally aligned solution since it does not receive any messages from its neighbors and did not have any additional *a priori* information (thus, its estimate is not plotted in the figure).

In order to see the strengths and limitations of the proposed approach, we evaluated the method on different randomly generated networks with varying average node connectivity. Figure 9 shows a simple network with two anchors and a node connectivity of two. As can be seen, the estimate produced by the proposed algorithm is not very accurate and fails to estimate the position of one of the nodes (node 5). This is because any time a node's uncertainty in its position is large, its estimate is not used to update the positions of a neighboring node. In other words, if node A's estimate uncertainty is large (e.g. prior to globally aligning itself or when its estimate uncertainty is an annulus), it does not send any messages about its belief to any of its neighbors. This ensures that node B's (a neighbor of node A) estimate does not become dependent (through linearization in the ROP space) on an inaccurate estimate of node A's position. It is due to this constraint imposed during implementation that in Figure 9 the estimate of two of the nodes (nodes 3 and 4) remain annuli and their common neighbor (node 5) lacks a globally aligned solution since it does not receive any messages from its neighbors and did not have any additional *a priori* information (thus, its estimate is not plotted in the figure). Note that in this case, a node i is considered 'globally aligned' when its uncertainties in the states $[c_x^i, c_y^i]$ are less than a threshold.

Figure 10 shows a plot further exploring the relationship between the maximum node connectivity of a graph and the average node mapping error. For each of the different node connectivity values, 10 different networks each with 40 nodes and 5 anchors were randomly generated and used to compute the plot. Note that any time a node's estimate cannot be globally aligned (in the case of being weakly connected to the rest of the network), its error is removed from the computation of the average node mapping error. In our experiments, at least 80% of the nodes in the network are sufficiently connected, such that they can be globally aligned, given an average node connectivity of 3. As can be seen from the figure, with a higher node connectivity, the mapping accuracy increases. Furthermore, since the proposed algorithm computes the minimum depth spanning tree, the depth of the tree decreases, reducing the number of iterations required for the algorithm to converge. However, as can be expected, with a higher connectivity between the nodes in the network, the state stored by each node in the

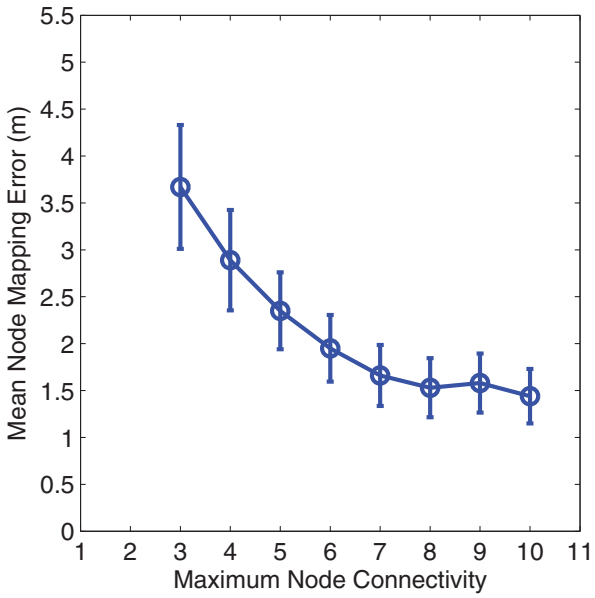

Fig. 10. Plot exploring the relationship between the maximum node connectivity of a graph and the average node mapping error. For each of the different node connectivity values, 10 different networks each with 40 nodes and 5 anchors were randomly generated and used to compute the plot. Anytime a node's estimate cannot be globally aligned, its error is not included in the computation of the average node mapping error. It can be observed that increasing node connectivity decreases the average mapping error.

network also grows, increasing the memory requirement for storing the mean vector and covariance matrix.

To test the scalability of the proposed loopy BP algorithm to large networks, we present two additional simulation experiments with 50 and 100 nodes. In these large networks, the nodes are sparsely connected (with each node connected to at most five other nodes), making it difficult to achieve a unimodal estimate for all the nodes. Figure 11 shows the final result achieved by loopy BP for each of the two large simulated networks.

Table 1 presents some numerical results that compare our proposed decentralized loopy BP algorithm with the centralized implementation. These results reveal that while the decentralized approach does not fully converge to the centralized approach, it provides a reasonable accuracy in our experiments. In particular, the error in the final mapped locations of the nodes in the loopy BP approach is very close to the centralized result, and the estimated path of the robot is only slightly less accurate.

5.2. *Experimental results*

We demonstrate the effectiveness of our proposed network mapping algorithm on two types of experiments. The first experiment is the 'static' mapping experiment where all of the nodes are stationary. Here, we assume the knowledge of a few nodes' true position (i.e. anchors) to help provide a rigid reference to the global coordinate frame and to reduce the ambiguities within the system. The second experiment extends the 'static' mapping experiment to include a single mobile node to the network. The mobile node moves within the limits of the stationary nodes but rarely has line of sight to more than a few stationary nodes at a time. Here, given the information provided by the mobile node, we show that it is possible to accurately estimate the nodes' position with even fewer anchors.

Table 1. Node mapping errors (in meters) for both simulated and real-world experiments and robot path errors for the real-world experiment with both the centralized and decentralized loopy BP implementations.

	Simulation		Real world	
	Map error	Map error	Path error	Map error
Method	50 nodes	100 nodes	14 nodes	
Centralized	2.18 m	1.69 m	0.55 m	0.45 m
Decentralized loopy BP	2.41 m	2.06 m	0.64 m	0.43 m

5.2.1. Noise characteristics In our experiments we utilized the nanoLOC ranging radio nodes from Nanotron Technologies (NanoTron, 2008). The nodes have a maximum range of 120 m in an open area with line of sight. Figure 12 presents the noise characteristics of these ranging radios in an open field with direct line of sight between all nodes in the environment. Calibrating for a linear correction of the measured range, it can be seen that the noise model of these radios is approximated by a zero-mean Gaussian with a 1.7 m standard deviation. While operating in environments with direct line of sight between all of the nodes is sufficient for some applications, in our desired experimental environments, line of sight between radios nodes cannot always be guaranteed.

The nanoLOC radio nodes also provide range measurements through some obstacles, such as thin walls. Unfortunately, in the presence of occlusions that limit the line of sight between radio nodes, the noise characteristics of the radio nodes also changes. Figure 13 presents the noise characteristics of the same radio nodes in an environment with many obstacles that restrict the line of sight between the nodes. Note that in environments with obstacles that occlude direct line of sight ranging, the maximum range of the nodes is limited (in our experiments it was limited to 30 m). As can be observed from the figure, in the presence of occlusions, a significant portion of the range data are subject to additional 'unmodeled' noise. This additional noise in general introduces a positive offset to the range data causing a 'second' peak in the noise histogram (seen in Figure 13). The presence of such 'unmodeled' noise in our measurements introduces an additional challenge to the network SLAM problem. It is, therefore, necessary to include proper

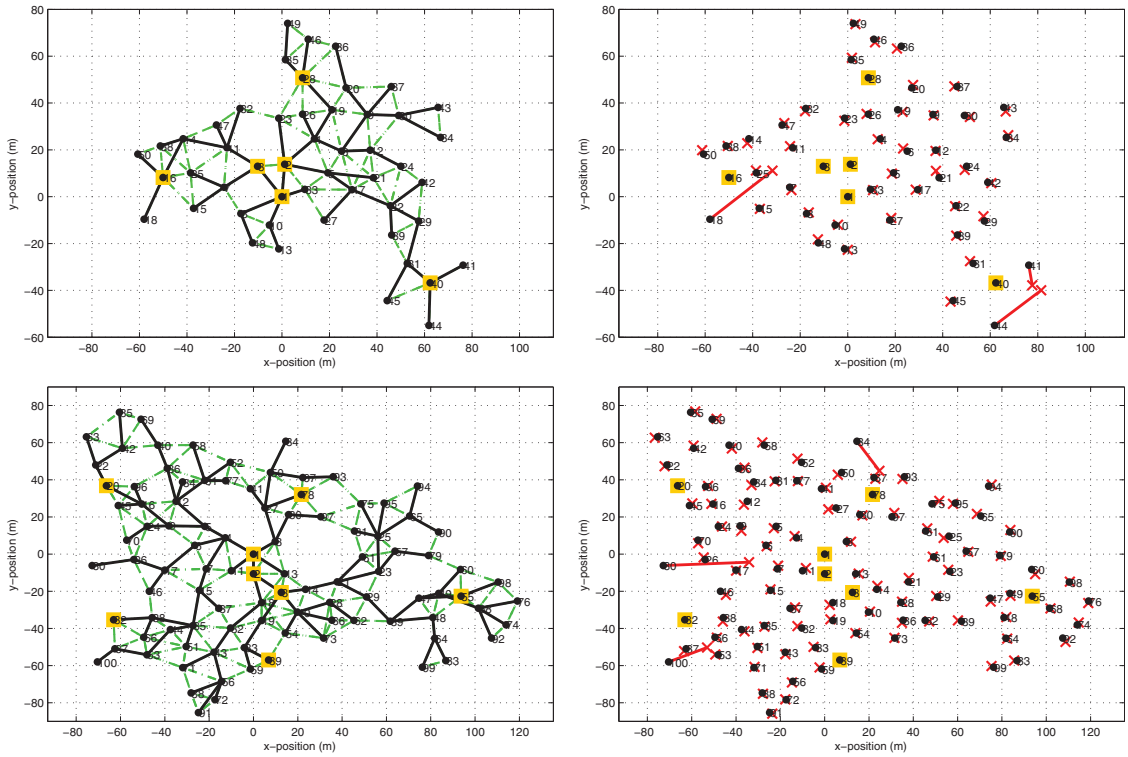

Fig. 11. Results of performing loopy BP on a 50-node and 100-node network. The nodes are sparsely connected (with each node connected to at most five other nodes), making it difficult to achieve an unimodal estimate for all the nodes. (Left) The true connectivity of the network represented by the green lines (also indicating the presence of range measurements between the connected nodes) and the spanning tree used by loopy BP to share messages between nodes (black lines overlayed on top green dashed lines). (Right) The estimated locations of the nodes (red cross marks $\times$) and the error lines (solid red) connecting the estimates to the true location of the nodes (black dots). The yellow squares highlight the anchors nodes whose true location is known *a priori*. In both example networks, some nodes in the network had multiple solutions. While the filter itself reports all of these solutions, the results visualized here correspond to the hypothesis with the lowest variance across all nodes. Note that the nodes with high error also correspond to the nodes with high uncertainty in position due to limited range connectivity to the other nodes.

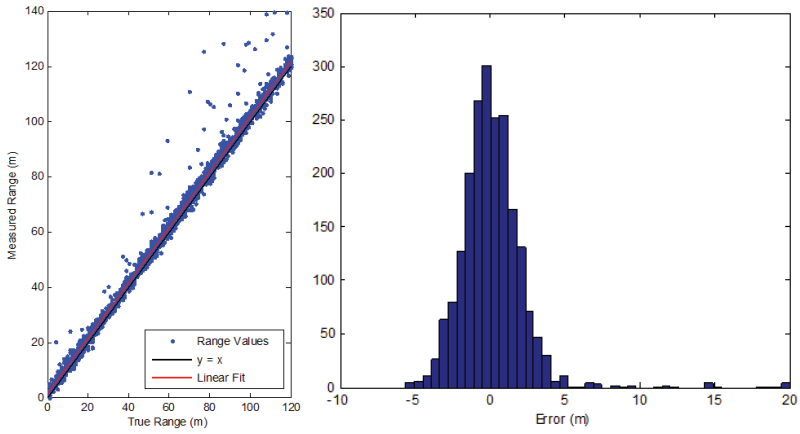

Fig. 12. Noise characteristics of range data collected using the nanoLOC ranging radios in an open field with direct line of sight between all nodes. (Left) Plot of measured range plotted against true range (surveyed with GPS data) along with the $y = x$ line and a linear fit of the data points. (Right) Histogram of the error in range measurements after a linear fit. The data fits a zero-mean Gaussian noise model with a standard deviation of 1.7 m.

measurement gating techniques, such as a chi-squared filter (Brumback and Srinath, 1987), to properly identify and remove/ignore/correct any measurements that do not fit the noise model of our sensor nodes.

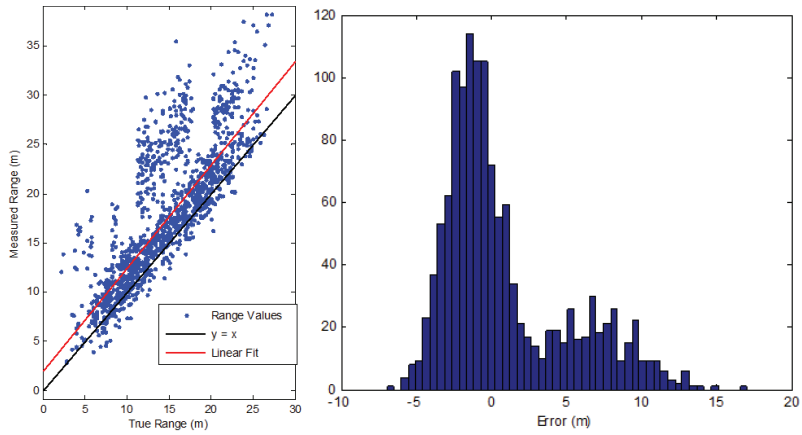

Fig. 13. Noise characteristics of range data collected using the nanoLOC ranging radios in an obstacle-filled environment with *no guarantee* of line of sight between the nodes. (Left) Plot of measured range plotted against true range (surveyed with GPS data) along with the $y = x$ line and a linear fit of the data points. The maximum range is limited due to the occlusions in the environment. (Right) Histogram of the error in range measurements after a linear fit. A 'second' peak in the histogram is evidence of noisy data caused by measuring range through obstacles in the environment.

5.2.2. Experimental setup In our experiments we deployed the radio nodes in an outdoor environment between and around several buildings. Figure 14 shows the floor plan of the environment where the nodes were deployed. Our experiments were conducted using an autonomous robot equipped with a ranging radio and wheel encoders. The robot was also equipped with a laser scanner to enable it perform simple obstacle avoidance and gather laser scan information about the environment (see Figure 15). In addition to ranging radio equipped on the mobile robot, several other stationary ranging radio nodes were arbitrarily deployed within the environment. The connectivity among a network of nodes within the environment, which directly corresponds to the ability to range between a pair of connect nodes, is shown in Figure 16.

Fig. 14. The blue shaded region indicate the free space and the red dots indicate the locations of the stationary nodes placed within the environment. The size of the workspace is 55 m × 70 m.

5.2.3. Static mapping In most real-world applications where pre-deployed infrastructures are available, it is often the case that the infrastructure is already operational before any mobile agent (such as a robot) enters the environment. In these cases, it is desirable if the pre-deployed nodes can themselves begin to solve the network SLAM problem immediately after their initial deployment. Figure 16 presents the results of our proposed method on a static network of nodes without the assistance of any mobile agents. The network consists of 16 nodes that are sparsely connected due to the obstacles in the environment. In addition, some of the range measurements are extremely noisy due to environmental effects such as multi-path.

A particular challenge with static network mapping in sparse networks is determining whether a given measurement is either a good, multi-path, non-line-of-sight (through thin obstacles) or outlier measurement and then dealing with it properly. A standard measurement gating technique, chi-squared gate, combined with the proposed polar parameterization is employed to reduce the effects of such noisy data on the filter's estimate. As can be seen from our results, the proposed approach was able to accurately reason about and reject the noisy measurements it encounters due to its improved representation of the non-linearities in the estimate uncertainty. However, as can be seen for some nodes, the estimates are shifted away from the true locations. This is due to the increased measurements noise observed in the environment as shown in Figure 13. Owing to the effects of multi-path and non-line-of-sight measurements, the noisy measurements cause the estimated positions of some of the nodes to be shifted. Given a purely static network, there is

Fig. 15. Robot equipped with a laser scanner and a ranging radio used in the experiments.

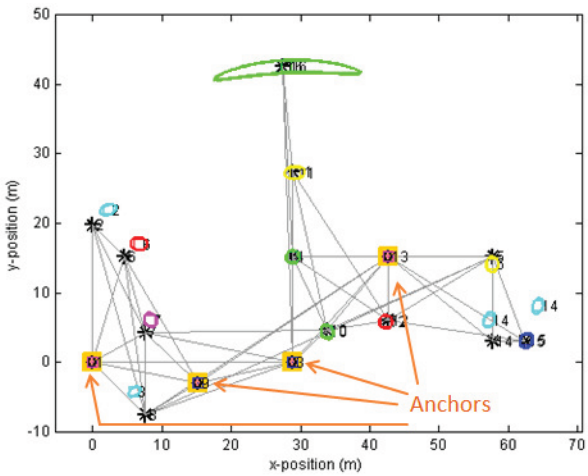

Fig. 16. Estimated node positions for the static mapping scenario. The nodes are plotted with their estimate uncertainties. The true positions of the four anchors nodes are known *a priori* to provide rigidity to the solution. The gray lines indicate the presence of range measurements between two nodes.

not too much more that can be done without completely changing the measurement model.

5.2.4. Mapping with a mobile node To test the influence of mobile nodes on the network mapping problem, we introduced a mobile node into the static network. In addition to the mobile node, there were 11 stationary nodes (a subset of the 16 nodes used in the static mapping experiment) deployed for this experiment. Figure 17 shows the estimated node positions and path of the mobile node using our proposed method. As can be seen, the resultant laser map that

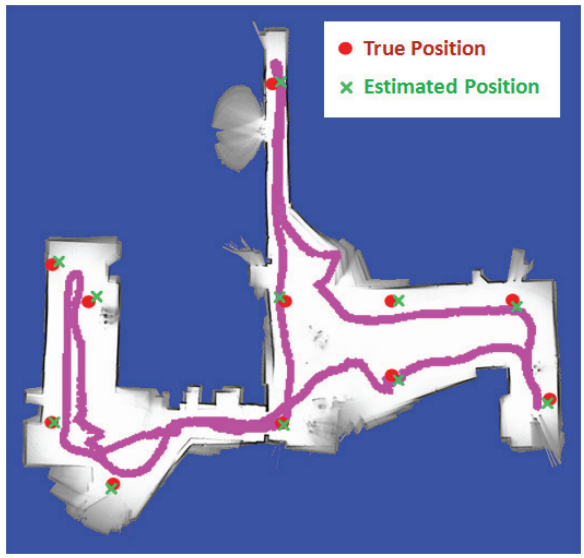

Fig. 17. The laser map generated by overlaying the laser scans from the mobile node on top of the estimated path of the mobile node using our proposed method. The proposed method assumes only two anchors, whose positions were known *a priori*. The resulting map looks closer to the true map and highlights the proposed method's position and heading accuracy.

is generated by overlaying the laser scans from the mobile node on top of the estimated path is very similar to the true floor plan map shown in Figure 14. Note that the map generated by overlaying the laser scans on the estimated robot path provides a good visual evaluation of the accuracy of the estimate provided by the filter. In addition to revealing any errors in estimating the mobile node's position, the map overlay also highlights errors in estimating the heading of the mobile node. It can be observed that much of the error in the resultant map (e.g. blurring of wall edges) is due to errors in estimating mobile node's heading correctly. This is because, in general, heading is difficult to estimate given range-only data. It should be noted here that the laser scans, used to generate the map in Figures 17, are only used to visualize the accuracy of the estimated position and heading of the mobile node (i.e. the laser scans are not used to improve the estimate).

In addition, we can compare the result of our approach to the laser map generated by a simple scan matching algorithm (available in CARMEN (Montemerlo et al., 2003)), Figure 18. The scan matching algorithm fails in this environment due to the lack of features in each scan. This is because the laser scanner had a maximum range of 8 m and in certain areas was only able to see one wall within the environment. Note that the use of a laser scanner with a maximum range of 8 m is not ideal for this environment, and comparing the result of using such a sensor (clearly unsuited for this environment) against our proposed method is most definitely unfair. However, it is also equally important to note that utilizing a laser scanner with a short range in the small environment shown here is analogous to using

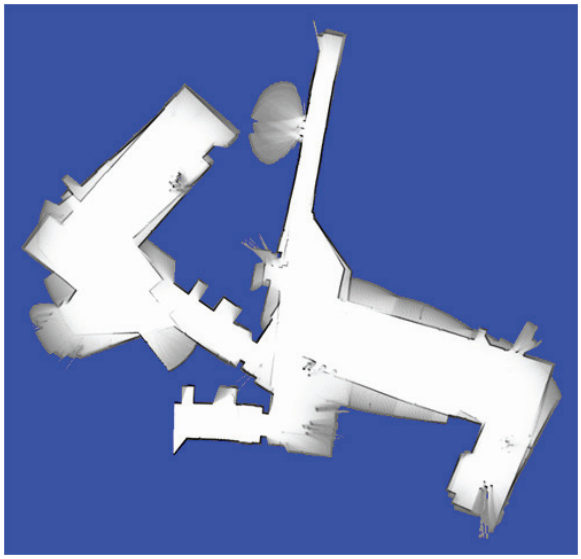

Fig. 18. The laser map generated by the scan matching algorithm within the CARMEN robotics toolkit (Montemerlo et al., 2003). Scan matching fails due to the lack of features visible in the short-sighted laser scans (max range 8 m). Note that the laser scanner was limited to a shorter maximum range of 8 m to highlight the difficulties scan matching algorithms face when operating in environments with sparse features.

Table 2. Average node mapping error and average error in the corner features of the laser map generated using the mobile node's estimated position. Rows 1 and 2 do not utilize the range measurements and are computed using either the odometry or laser scan data. Row 3 presents the result of utilizing only the range data, while rows 4 and 7 present results of performing SLAM using both range and odometry data with varying initial information.

Method	Average node error	Average laser map error
Dead reckoning	—	9.62 m
Scan matching	—	12.27 m
Static mapping (four anchors)	1.89 m	—
SLAM initialized w/ static mapping result	0.82 m	1.86 m
SLAM (two anchors)	1.19 m	2.62 m
SLAM (four anchors)	0.95 m	2.18 m
Localization (all 11 anchors)	0.00 m	0.87 m

a laser scanner with a large range in a larger environment. As such, comparing the proposed ranging radio-based map against the laser scan matching-based map highlights the benefits and utility of using ranging radios as a complimentary sensor to other more commonly used sensors, such as laser scanners, in large environments.

Table 2 presents the numeric results comparing the proposed method against several other strategies, including an initialized version of SLAM, where the static mapping result (shown in Figure 16) is used to initialize the state of the nodes when performing SLAM with a mobile node. The node errors reported in the table above were calculated based on manually surveyed ground truth node locations. The 'laser map' errors reported in the table were calculated based on extracting corner features from the estimated map and comparing it against the locations of those same features within the ground truth floor plan, supplemented with manual measurement of those corner features within the actual environment. For this environment, a total of 23 corner features and 11 node positions were examined to produce the results shown in Table 2. In the cases when a corner feature was blurred within the estimated laser map, the worst-case position of the corner was used. In other words, if the corner in an estimated map is blurry (i.e. the multiple laser scans of the corner are not properly aligned), the corner extracted from the laser scan that was the farthest from the true corner location was used to compute the error.

The results in Table 2 reveal that the addition of motion information from a mobile node significantly improves the overall node mapping accuracy. In addition, it can be seen that the average node position error for performing SLAM initialized with the static mapping solution (which used four anchors) is noticeably lower than the error achieved by performing SLAM from scratch with the same four anchors. While the reason for this improvement might not be obvious at first, it is cause by two key factors. First, the measurements used to compute the initial static mapping solution provide some extra information to the filter, thereby improving the estimate a little. However, the second more important factor responsible for the reduction in the observed error is the fact that the initialized node estimates help limit the error in the mobile node's position caused by drift in its odometry. When no initialized node estimates are available, the mobile node's estimate is used to initialize the non-anchor nodes when they are first observed causing any error due to odometry drift is carried over to the stationary nodes' estimates. Thus, the error accumulated from odometry drift is never corrected when performing SLAM without any initialization. In contrast, when the static mapping solution is used to initialize SLAM, drifts in the odometry can be corrected from the very beginning. It can therefore be concluded that it is better to compute a static mapping solution prior to deploying a mobile node and executing SLAM.

Figure 19 shows the performance of our algorithms as the number of anchors in the environment is varied. As can be expected, when the number of anchors increases, the error in the position of the nodes in the environment decreases. In particular, looking at the effect of adding a single mobile node to the network, it can be observed that the addition

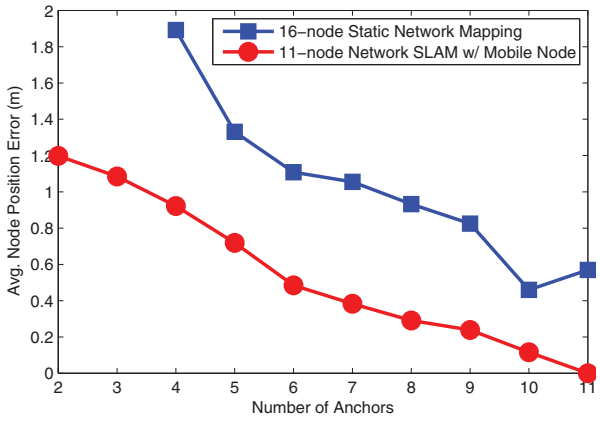

Fig. 19. Plots revealing the effects of varying the number of anchors in the environment is shown. As can be expected, in both the static network mapping case with 16 nodes (blue squares) and the 11-node network mapping case with a mobile node (red circles), as the number of anchors is increased, the overall error in mapping the node locations decreases. However, as can be seen, given the additional information from the odometry of a mobile node, the network can be better localized with much few number of anchors.

information provided by the odometry of the mobile node significantly helps improve the overall node mapping error.

5.2.5. Decentralized estimation results The real-world experiment presented here is a variant of the traditional network localization problem, where not all of the nodes in the network remain stationary. Therefore, the 'network' in this experiment consists of all of the stationary nodes in the environment and a single mobile node/robot. The robot was equipped with a ranging node (an ultra-sound based ranging node was used in this experiment (Zhang et al., 2007)) placed on top of the robot at about 0.3 m above the floor. The robot was driven around within a large indoor office area with partial clutter. Ground truth of the robot's position was estimated using a laser scanner and the Adaptive Monte Carlo Localization (AMCL) algorithm within the Player/Stage (Gerkey et al., 2003) code repository. In addition to the node that was placed on the robot, 13 other nodes were placed around the environment on top of stands, 0.3 m above the floor. Sparse connectivity between the nodes, makes it impossible to achieve an unimodal localization result without motion of the mobile robot. The locations of these nodes were accurately surveyed to allow proper evaluation of the accuracy of our mapping results. The robot was also running a low-level obstacle avoidance scheme that avoided collisions while attempting its best to keep to the planned trajectory.

Figure 20 shows the final localization result achieved by our method when the mobile node moves within the environment. A particular challenge with using real hardware is the slow rate of range measurements. Since the hardware does not support instantaneous range observations from several nodes at once, special considerations must be made to ensure that sufficient constraints exist to resolve ambiguities. To do this, we collect measurements over a period of 1 second and process them together, in order to retain correlations within the sequential observations. In addition, it should be noted here that in this real-world experiment, achieving an accurate estimate of the full network is impossible, without the mobile node, due to the lack of rigidity and sparse connectivity within the network. The numeric mean robot path error and node mapping errors are reported in Table 1.

Figure 21(left) shows the mean mapping error of the nodes over time as the mobile node moves around in the environment. Figure 21(right) shows the mean uncertainty in the position estimate of the nodes over time. We see that the decentralized loopy BP approach initially has a low mean uncertainty but the error in the solution is high. This is because the estimates of the isolated nodes, maintained independently by each node, drift at the start in the absence of sufficient measurements. It is only after the mobile node travels within range of the isolated nodes can their estimates be fixed within the joint coordinate frame (without which their estimates remain relatively accurate but free-floating). In contrast, the centralized method has to deal with more ambiguities (large multi-modal distributions) at the start when fewer measurements are available. And since the method tries to jointly estimate the positions of all of the nodes within the same coordinate frame, the estimates of the isolated nodes do not drift.

Computation complexity Table 3 reveals the average dimension of the state vector maintained by each node in the network and the average computation time utilized by each node while running on separate threads on a 2.4 GHz Intel quad-core processor. The decentralized filtering code is not fully optimized and so the computation times reported here can be further improved if the code is fully optimized. Looking closer at the numerical values, we see that the computation times required at each node by the decentralized filter is much lower than the centralized filter. In addition, if the connectivity of the network remains the same with the addition of more nodes, the computational requirement for each node in the network remains the same in the decentralized filter. In contrast the computational requirements for centralized filter increases as more and more nodes are added to the network regardless of the connectivity.

6. Conclusions

In this paper, we examined the problem of motion-aided network SLAM. The method we proposed offers an alternate polar parameterization that is better suited for dealing with the non-linear measurement distributions evident in range-only data. We show that through the use of this improved parameterization, and measurement filtering techniques, it is possible to achieve improved localization and mapping of a sparsely connected network of nodes in the

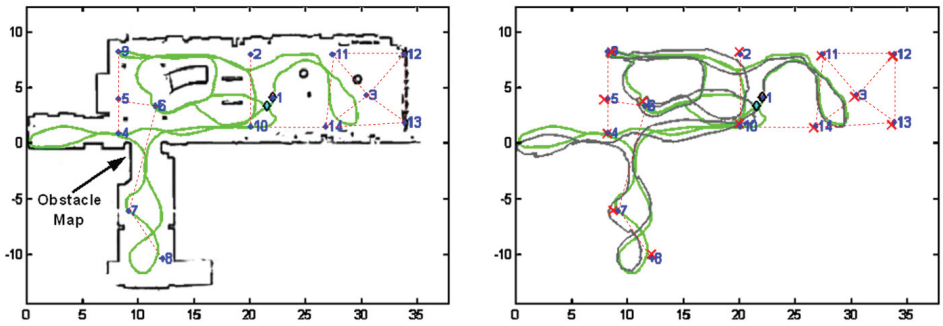

Fig. 20. Result of performing loopy BP on a real system with a single mobile agent is shown. The map of the environment for the real-world example is overlaid on top of the true node locations and path. As the robot moves, the loopy BP solution from the previous time step is used as prior for each consecutive time step. (Left) The true locations of the nodes (∗), all of the inter-node measurements received (dashed black line) and the true path the robot (ID #1) took (dotted green line). (Right) The error lines connecting true and estimated positions of the nodes along with the estimated path of the robot (solid gray line). The red cross marks (×) the estimated location of a node and the error lines (solid red) connect the estimates to the true location of the nodes (black dots).

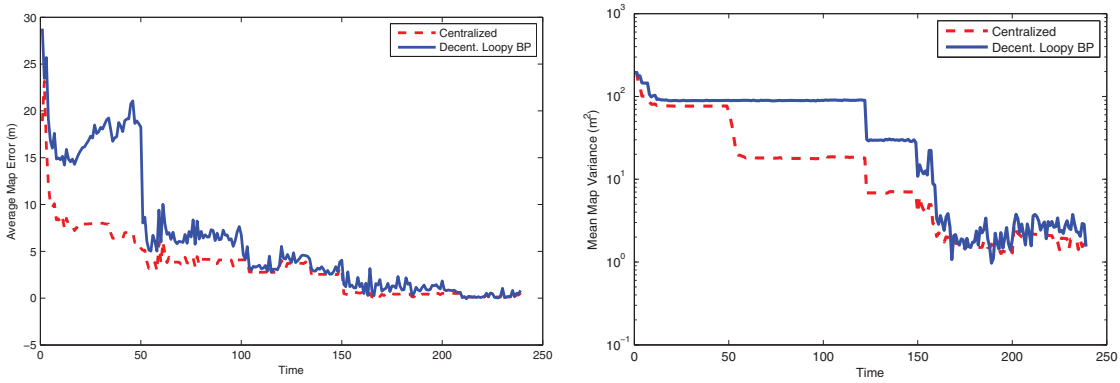

Fig. 21. (Left) Mean position error and (right) mean uncertainty (i.e. mean variance) of all of the nodes in the real-world experiment for both the centralized (dashed red line) and decentralized (solid blue line) implementations. In the decentralized approach, the estimates of the isolated nodes drift at the start in the absence of sufficient measurements, increasing its mean error.

Table 3. The decentralized method while it requires additional communication between each node, it offers significant improvements in computational times and reduced state dimension per node.

	Simulation				Real world	
Method	Average number of nodes in state vector	Computation time/node	Average number of nodes in state vector	Computation time/node	Average number of nodes in state vector	Computation time/Node
Centralized	50	97.61 s	100	145.87 s	14	20.96 s
Decentralized loopy BP	4.12	9.33 s	4.50	10.52 s	4.1	8.97 s

presence of noisy range-only measurements. In addition, the centralized network SLAM algorithm was adapted into a distributed and decentralized solution. This approach extends the traditional loopy BP algorithm by reducing the network graph to a spanning tree that also offers better convergence guarantees. The proposed loopy BP approach efficiently stores and computes part of the global network localization problem at each node, achieving accuracy similar to the centralized solution with little computation performed on each node in the network. This approach, designed to be asynchronous, was shown to be adaptable to changes in the network graph.

The scalability of the proposed approach was tested on several large and small networks, including a 100 node network and on a smaller 14-node real-world sensor network. It was shown that in all of these networks, the loopy BP algorithm requires significantly little computation to be done on each in the network, compared with the computation required for the centralized approach. In addition, it was shown that in the presence of a single moving node, which

dynamically alters the connectivity of the network graph over time, the proposed algorithm was able to accurately estimate both the path of the mobile node and the positions of the other stationary nodes.

Furthermore, comparing the map reconstruction results of our approach with laser-based scan matching techniques, we found that while the proposed range-based mapping solution has difficulty in estimating heading in some cases, it offers a good complimentary solution to traditional laser-based mapping techniques. In particular, when mapping environments that are challenging for existing laser-based techniques due to its scale and lack of environmental features, our proposed range-based mapping solution is a good alternative. Lastly, we also examined the effects of adding anchors nodes with known prior location to assist in mapping other nodes and initializing the SLAM algorithm with the static mapping result. Our results revealed that the addition of a mobile node and the use of the static mapping result to initialize SLAM offer significant improvements to the overall node mapping accuracy

Funding

This work is funded in part by Boeing Research & Technology.

References

Bahl P and Padmanabhan V (2000) RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the IEEE Infocom*.

Borg I and Groenen P (1997) *Modern Multidimensional Scaling: Theory and Applications*. New York: Springer.

Boyen X and Koller D (1998) Tractable inference for complex stochastic processes. In *Proceedings of Uncertainty in Artificial Intelligence*.

Brumback B and Srinath M (1987) A chi-square test for fault-detection in Kalman filters. *IEEE Transactions on Automatic Control* 32(6): 552–554.

Bui M, Butelle F and Lavault C (2004) A distributed algorithm for constructing a minimum diameter spanning tree. *Journal of Parallel and Distributed Computing* 64: 571–577.

Cao M, Morse A and Anderson B (2006) Agreeing asynchronously in continuous time. In *IEEE Conference on Decision and Control*.

Cao M, Morse A and Anderson B (2008) Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization* 47: 575–600.

Chow C and Liu C (1968) Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* 14: 462–467.

Djugash J and Singh S (2008) A robust method of localization and mapping using only range. In *International Symposium on Experimental Robotics*.

Djugash J, Singh S and Grocholsky B (2009) Modeling mobile robot motion with polar representations. In *International Conference on Intelligent Robots and Systems*.

Djugash J, Singh S and Grocholsky BP (2008) Decentralized mapping of robot-aided sensor networks. In *IEEE International Conference on Robotics and Automation*.

Djugash J, Singh S, Kantor G and Zhang W (2006) Range-only SLAM for robots operating cooperatively with sensor networks. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Eustice RM, Singh H and Whitcomb LL (2011) Synchronous-clock one-way-travel-time acoustic navigation for underwater vehicles. *Journal of Field Robotics* 28: 121–136.

Faloutsos C and Lin K-I (1995) FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Carey MJ and Schneider DA (eds), *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, CA, pp. 163–174.

Funiak S, Guestrin CE, Sukthankar R and Paskin M (2006) Distributed localization of networked cameras. In *Fifth International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 34–42.

Gerkey B, Vaughan R and Howard A (2003) The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, 30 June–3 July 2003, pp. 317–323. http://playerstage.sourceforge.net/.

NanoTron (2008) *nanoLOC TRX Transceiver (NA5TR1)*. Datasheet NA-06-0230-0388-2.00, NanoTron.

Ihler AT, Fisher III JW, Moses RL and Willsky AS (2004) Nonparametric belief propagation for self-calibration in sensor networks. In *Information Processing in Sensor Networks*.

Kamath S, Meisner E and Isler V (2007) Triangulation based multi target tracking with mobile sensor networks. In *Proceedings IEEE International Conference on Robotics and Automation*.

Makarenko A and Durrant-Whyte H (2004) Decentralized data fusion and control in active sensor networks. In *Proceedings of the Seventh International Conference on Information Fusion*.

Montemerlo M, Roy N and Thrun S (2003) Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In *Proceedings IEEE/RSJ International Conference Intelligent Robots and Systems*, pp. 2436–2441.

Moore D, Leonard J, Rus D and Teller S (2004) Robust distributed network localization with noisy range measurements. In *SenSys'04:Proc 2nd international conference on Embedded networked sensor systems*. New York: ACM Press, pp. 50–61.

Olson E, Leonard J and Teller S (2004) Robust range-only beacon localization. In *Proceedings of Autonomous Underwater Vehicles*.

Pearl J (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann.

Pfeffer A and Tai T (2005) Asynchronous dynamic bayesian networks. In *Proceedings of Uncertainty in Artificial Intelligence*.

Prim R (1957) Shortest connection networks and some generalizations. *Bell Systems Technical Journal* 36: 1389–1401.

Priyantha N, Chakraborty A and Balakrishman H (2000) The Cricket location support system. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*.

Rosencrantz M, Gordon G and Thrun S (2003) Decentralized sensor fusion with distributed particle filters. In *Proceedings of Uncertainty in Artificial Intelligence*.

Smith A, Balakrishnan H, Goraczko M and Priyantha NB (2004) Tracking moving devices with the Cricket location system. In

2nd International Conference on Mobile Systems, Applications and Services (Mobisys 2004).

Yang P, Freeman R and Lynch K (2007) Distributed cooperative active sensing using consensus filters. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, pp. 405–410.

Zhang W, Djugash J and Singh S (2007) *Parrots: A Range Measuring Sensor Network*. Technical Report CMU-RI-TR-06-05, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Zhao F, Liu J, Liu J, Guibas L and Reich J (2003) Collaborative signal and information processing: An information directed approach. *Proceedings of the IEEE* 91: 1199–1209.