HEVA: Cooperative Localization Using A Combined Non-Parametric Belief Propagation and Variational Message Passing Approach

Panagiotis-Agis Oikonomou-Filandras, and Kai-Kit Wong

***Abstract*—This paper proposes a novel cooperative localization method for distributed wireless networks in 3-dimensional (3D) global positioning system (GPS) denied environments. The proposed method, dubbed as hybrid ellipsoidal variational algorithm (HEVA), combines the use of non-parametric belief propagation (NBP) and variational Bayes (VB) to benefit from both the use of the rich information in NBP and compact communication size of a parametric form. In HEVA, two novel filters are employed. The first one mitigates non-line-of-sight (NLoS), Time-of-Arrival (ToA) messages, permitting it to work well in high noise environments with NLoS bias while the second one decreases the number of calculations. Simulation results demonstrate that HEVA significantly outperforms traditional NBP methods in localization and at the same time requires only about 50% of their complexity. The superiority of VB over other clustering techniques is also illustrated.**

***Index Terms*—Cooperative localization, Non-line-of-sight, Non-parametric belief propagation, Variational Bayes.**

I. Introduction

Location awareness (or localization) has become an important feature in both military and civilian wireless networks, with applications ranging from search-and-rescue missions to team coordination and logistics in military scenarios, and even efficient use of the spectrum in cognitive radio technologies. In fact, there has been an explosive growth in the number of location-based services (LBSs) and so has the need for mobile devices to quickly self-localize in arbitrary and potentially very unfriendly environments. The current *de facto* technology for localization is the global positioning system (GPS) [1], which uses satellite-derived metrics for a device to estimate its location. GPS works well in outdoors scenarios, but unfortunately, the efficiency of GPS is quite limited in closed environments such as building complexes, underground or heavily canopied forests. This precipitates the urgent need for the development of localization technologies for indoors environments.

One of the major differences in localization between indoors and outdoors is the availability of devices willing to assist in the localization process. In the case of using GPS, a device is required to communicate with at least four and usually seven or more different satellites in order to localize. For indoors, however, the number of suitable neighbors is highly random, and there is no guarantee that it will be adequate. To tackle this, the idea of cooperation between devices (or nodes) has been developed. Nodes can achieve localization by exchanging positioning information [2]. In this case, nodes should share their position estimates and derive distances metrics, such as time-of-arrival (ToA), time-difference-of-arrival (TDA), angle-of-arrival (AoA), or received-signal strength (RSS), etc, from the signals of the exchanged messages to refine their position estimates. Also, ultra-wideband (UWB) signals are especially suited for this purpose [3], since they can provide reliable fine resolution distance measurements in indoors environments.

Given a wireless network, some nodes, which are usually referred to as anchors, have precise location estimates, but some, referred to as agents, aim to self-localize using two types of information. Firstly they share their believed location with their neighbors and secondly they derive metrics of the relative distance between themselves and their neighbors from the physical properties of the corresponding transmissions. Each position estimate and distance metric an agent possesses for a corresponding neighbor defines a geometric area (e.g., ring, sphere or hyperbole) for its whereabout. As a consequence, the location of the agent can then be estimated by the intersection of all such geometric areas defined from all its neighbors, a process called multilateration. The advantage is twofold. First of all, the requirement of having at least 4 (in the 3D case) anchors to communicate with an agent is overcome. Secondly, by sharing information, anchor estimates can diffuse to agents that are out of range of any anchor, thereby virtually extending the communication range of all nodes. Cooperative localization for GPS-denied environments is currently a hot research topic and there has been extended interest from the robotic, optimization and wireless communication communities.

The fundamental limits of cooperation in localization are also of tremendous interest. In [4], an accuracy measure for a node's localization was introduced which takes into account both the measurement noise variance and the geometry of the neighboring noise in order to derive a Cramer-Rao lower bound (CRLB). Most recently in [5], Shen *et al.* derived the squared position error bound (SPEB), a CRLB specifically for localization in multi-path Rayleigh fading channels.

The shared information in a cooperative network can either be analyzed centrally [6], or in a distributed manner. However, a much greater importance has been dedicated to distributed solutions since they provide a much more flexible and robust framework. In the distributed case a number of algorithms have been developed and they can be divided into deterministic and probabilistic algorithms. For deterministic algorithms, nodes consider localization as an optimization problem trying to find

This work was supported by the EPSRC [Grant Number EP/H011536/1].

The authors are with the Department of Electronic and Electrical Engineering, University College London, WC1E 7JE, United Kingdom.

the best point estimate. A well-known example of this is the IPPM algorithm reported in [7]. IPPM treats each message from a neighboring node as a distinct optimization problem, and then tries to find the solution to minimize the average of all the neighboring optimization problems. Other well-known deterministic methods are the least-square (LS) method in [2] and the multidimensional scaling (MDS) algorithm in [8].

For the probabilistic methods, instead of making an initial guess and then attempting to converge to the true solution, they assign probability distributions for the whole space and attempt to avoid the trap of local optima. There have been a number of probabilistic techniques to address the localization problem. One family uses particles to approximate the localization distribution. A representative particle-based method is the non-parametric belief propagation (NBP) approach [9]. Later, [10] studied the effect of different kernels and argued the choice of spherical Gaussian kernel with variances calculated using the "rule of thumb". The SPAWN algorithm proposed in [2] was a general framework encompassing algorithms based on NBP, introducing a factor graph message passing analysis. Other methods include Monte-Carlo chains in [11], or variational message passing (VMP) in [12]. VMP approximates the real localization distributions with a simpler one, which is more tractable and easier to handle. The use of clustering techniques has also been proposed to create a parametric form of the transmitting messages, e.g., expectation maximization (EM) and k-means have been suggested in [13].

Also, it should be noted that in the family of message passing techniques, ameliorating the effects of loopy propagation (i.e., non-guaranteed convergence, and overpowering effect of a few nodes via loops) is an open research topic. Research suggests that the use of asynchronous message passing scheduling offer solutions in the aforementioned problems [14, 15] and a solution for the localization problem based on tree-reweighed belief propagation has also been proposed in [16].

Finally, non-line-of-sight (NLoS) mitigation is a key issue in indoors localization and has been a major research topic. NLoS message identification for UWB systems has been considered in [17, 18] and NLoS mitigation for the deterministic case has been extensively addressed in [17, 19, 20]. A survey for UWB NLoS mitigation techniques can be found in [21]. The issue of NLoS is especially important when using ToA measurement methods, as NLoS bias errors are always positive and much larger than the distance measurement, c.f. [22], causing large estimation errors. ToA NLoS mitigation has been extensively studied and a survey can be found in [23]. Nevertheless, most studies on NLoS localization mentioned above only consider single-node localization, and are based on linearizing the measurement model which is not applicable in the distributed and cooperative case [7]. To the authors' knowledge, only [7, 22, 24] explicitly considered the cooperative and distributive case, the first two being truly distributed solutions.

Nevertheless, critical issues remain in localization. Most of the algorithms in the literature are limited in scope and cannot be extended to more realistic and complex scenarios. First, although most algorithms state that they can be extended to the 3D case, few results can be found in the literature. In particular, particle-based algorithms such as NBP will quickly become computationally prohibitive in the 3D case due to the exponential increase in the number of particles. Likewise, for VMP, despite avoiding the pitfall of the number of particles, it requires a huge increase in the required number of messages passed between the nodes until convergence.

The main contribution of this paper is a novel hybrid ellipsoidal variational algorithm (HEVA) that overcomes the pitfalls of the aforementioned algorithms and is computationally tractable in the 3D case. Our proposed algorithm can achieve higher localization accuracy, getting more nodes self-localized within given accuracy and at much less complexity than NBP [9]. Although IPPM in [7] and LS in [2] are less complex, their accuracies are much inferior, especially under high noise environments. HEVA combines the strengths of both VMP and NBP to reach a fine balance between the information used in the probabilistic inference, the message size and convergence speed. The proposed HEVA manages to decrease the amount of lost useful information in the approximations in the inference calculations and as such localizes with high resolution. Also, HEVA introduces a cooperative NLoS mitigation technique for distributed networks using ToA distance measurements that filters out unhelpful NLoS messages.

The remainder of this paper is organized as follows. Section II formulates the bayesian localization problem. Section III describes the HEVA algorithm, including analysis for complexity and communication costs. Simulation results are provided in Section IV and we conclude the paper in Section V.

II. Problem Formulation

We consider a set of nodes in a 3D environment of size $X \times Y \times Z$ m^3. The nodes consist of N agents and M anchors, where $|\mathcal{V}| = N + M$, $M \geq 4$ and $N \gg M$.[1] Let $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_{N+M}]$ denote the locations of all nodes, with $\boldsymbol{\theta}_i = [x_i, y_i, z_i]$ being the coordinates of node i, and $\boldsymbol{\Theta}_i$ be the respective random variable. The nodes communicate wirelessly and it is assumed that the maximum communication range for each node is $r_{\max}$. Time is slotted and time slots are denoted by the time index superscript (t) for $t = 1, \ldots, \infty$.

A network of such can be viewed as a graph. The wireless nodes are represented by the set $\mathcal{V}$ of vertices of the graphical model. If two nodes, say node i and node j, are within range, there will be an edge $e_{ji} \in \mathcal{E}$, connecting the two nodes. The set of all nodes j with edges e_{ji} to node i is denoted as the neighborhood $\mathcal{N}_i$. A simple network graph example with agents, represented as "a" labelled red circles and anchors, represented as "A" labelled green circle, is depicted in Fig. 1.

Let $p^{(t)}(\boldsymbol{\Theta}_i)$ be the probability density function (pdf), i.e., the belief that node i has about its location at time t. Initially, the belief for the agents can be an information-less uniform pdf over the grid, while the anchors' pdfs would be spherical multivariate Gaussians with mean being their exact locations and a covariance matrix of $\sigma_a^2\mathbf{I}$ for some noise power σ_a^2.

Nodes can measure a corresponding distance estimate via ranging. For example, node i receiving a message from node j

[1] Typically the number of agents is assumed much larger than the number of anchors [2].

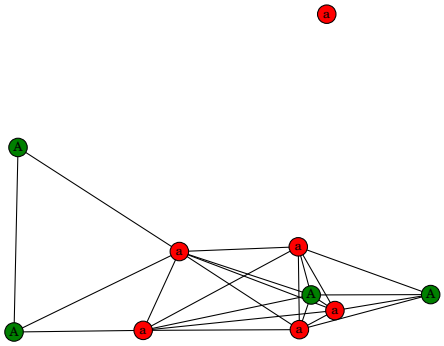

Fig. 1. An example of a network where agents are represented by lowercase "a" (red circles), anchors are represented by capital "A" (green circles) and branches (or edges) correspond to communication between them.

at time slot t can derive a noisy estimate $r_{j\to i}^{(t)}$ of the distance between them. It is assumed that the measurement taken from node i receiving a message from node j and that from node j receiving a message from node i are the same and as a result, the direction of the message plays no role, i.e., $r_{j\to i}^{(t)} = r_{i\to j}^{(t)} = r_{ji}^{(t)}$. In practice, distance measurements will differ and the nodes can share their measurements and use the average. We assume that the nodes use ToA distance measurements, as in [25] and we define the random variable r_{ji} as

$$r_{ji} = d_{ji} + \eta_{ji} \equiv \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\| + \eta_{ji}, \tag{1}$$

where η_{ji} is a noise factor following a Gaussian distribution with variance $K_e\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^{\beta_{ji}}$, in which K_e is a proportionality constant capturing the combined physical layer and receiver effect [26], and β_{ji} denotes the path loss exponent. In the case of line-of-sight (LoS) η_{ji} is assumed to have zero mean, and $\beta_{ji} = 2$, i.e., $\eta_{ji} \sim \mathcal{N}(0, K_e\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^2)$. Alternatively, in the case of NLoS, the Gaussian random variable has a positive mean $b_{ji} \gg s_{ji}^2$, where $b_{ji} \sim \mathcal{U}(\frac{d_{ji}}{3}, \frac{2d_{ji}}{3})$ (i.e., a uniform distribution) and $\beta_{ji} = 3$, i.e., $\eta_{ji} \sim \mathcal{N}(b_{ji}, K_e\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|^3)$.

Let $p^{(t)}(R_{ji} = r_{ji}^{(t)}|\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j)$ be the conditional pdf (cpdf) of observing distance $r_{ji}^{(t)}$ at time slot t, given the location beliefs $p^{(t)}(\boldsymbol{\Theta}_i)$ and $p^{(t)}(\boldsymbol{\Theta}_j)$ for node i and node j, respectively. Assuming statistically independent noise and statistically independent priors between nodes, the joint pdf of the whole probabilistic model with $\boldsymbol{\vartheta} \triangleq (\{\forall \boldsymbol{\Theta}_i \in \mathcal{V}\})$ and $\boldsymbol{R} \triangleq (\{R_{ji}\}_{\forall i,j\in\mathcal{E}})$, for a given time slot t, is found as

$$\begin{aligned} p^{(t)}(\boldsymbol{\vartheta}, \boldsymbol{R}) &= p^{(t)}(\boldsymbol{R}|\boldsymbol{\vartheta})p^{(t)}(\boldsymbol{\vartheta}) \\ &= \prod_{i,j\in\mathcal{E}} p^{(t)}(R_{ji} = r_{ji}|\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j) \prod_{\boldsymbol{\Theta}_i\in\mathcal{V}} p^{(t)}(\boldsymbol{\Theta}_i). \end{aligned} \tag{2}$$

Our aim is to find the most probable position for every node given the observed positions and prior information, i.e., find the maximum a posteriori (MAP):

$$\hat{\boldsymbol{\vartheta}} = \arg\max_{\boldsymbol{\vartheta}} p^{(t)}(\boldsymbol{\vartheta}|\boldsymbol{R} = \boldsymbol{r}), \tag{3}$$

where $\boldsymbol{r}$ is a vector representing all the observed distances. To accomplish this, we employ a message passing algorithm [14], in which information from the graph can be summarized in local edge information, allowing for an efficient distributed algorithm, despite its lack of guaranteed optimal solution or even convergence for the given random graph geometry.

A. Belief Message Passing

The network can be modelled by a cluster graph and a loopy belief message passing algorithm can be applied. We choose the Bethe cluster graph [27] which is composed of two types of factors. The lower factors, which represent the node beliefs, are composed of univariate potentials $\psi(\boldsymbol{\Theta}_i)$, whereas the upper region, which represents the interactions between the node variables, is composed of "large" factors equal to $\psi(\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j, r_{ji})$. An example of the network in Fig. 1 is shown in Fig. 2. The lower factors are set to the initial node beliefs for the given time slot (t), and the upper factors to the corresponding cpdfs, i.e.,

$$\psi(\boldsymbol{\Theta}_i) = p^{(t)}(\boldsymbol{\Theta}_i), \tag{4}$$

$$\psi(\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j, r_{ji} = r_{ji}^{(t)}) = p^{(t)}(r_{ji} = r_{ji}^{(t)}|\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j). \tag{5}$$

Messages are then passed between nodes for multiple iterations until the node beliefs have converged, or a predetermined number of iterations has passed. The message from node j to node i at BP (belief propagation) iteration $(s+1)$ is found by

$$\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i) = \int \psi(\boldsymbol{\Theta}_i, \boldsymbol{\Theta}_j, R_{ji} = r_{ji}^{(t)}) \frac{b_{j\to i}^{(s)}(\boldsymbol{\Theta}_j)}{\delta_{i\to j}^{(s)}(\boldsymbol{\Theta}_j)} \mathrm{d}\boldsymbol{\Theta}_j, \tag{6}$$

where $r_{ji}^{(t)}$ is the observed value of the distance between the nodes, at time slot t. Intuitively, a message $\delta_{j\to i}^{(s+1)}(\Theta_i)$ is the belief that node j has about the location of node i and is a function of $\boldsymbol{\Theta}_i$. After it calculates all the incoming messages, each node updates its belief by

$$b_i^{(s+1)}(\boldsymbol{\Theta}_i) = \lambda\psi(\boldsymbol{\Theta}_i) \prod_{k\in\mathcal{N}_i} \delta_{k\to i}^{(s+1)}(\boldsymbol{\Theta}_i) + (1-\lambda) b_i^{(s)}(\boldsymbol{\Theta}_i), \tag{7}$$

in which $\lambda \in [0, 1]$ is a dampening factor used to facilitate convergence, c.f. [14].[2] The algorithm continues until convergence or a set number of iterations $s_{\max}$ has ellapsed. The belief that represents an aproximation to the true marginal, for each node is given by (7), i.e.,

$$p^{(t+1)}(\boldsymbol{\Theta}_i) = b_i^s(\boldsymbol{\Theta}_i). \tag{8}$$

This can be thought of as a process of merging every node's belief about a specific node's location to get the best estimate. This message passing analysis naturally leads to a distributed cooperative system because each node is only required to do calculation concerning its local factors and messages.

A major issue in calculating the messages (6) and (7) is the computational cost. If $\boldsymbol{\Theta}_i$ can take one of D discrete values, then the messages and marginals in the 3D case will

[2]Dampening is not required for 2D localization, but simulations show that it helps in 3D localization where the increased ambiguity results in oscillating behaviors more often.

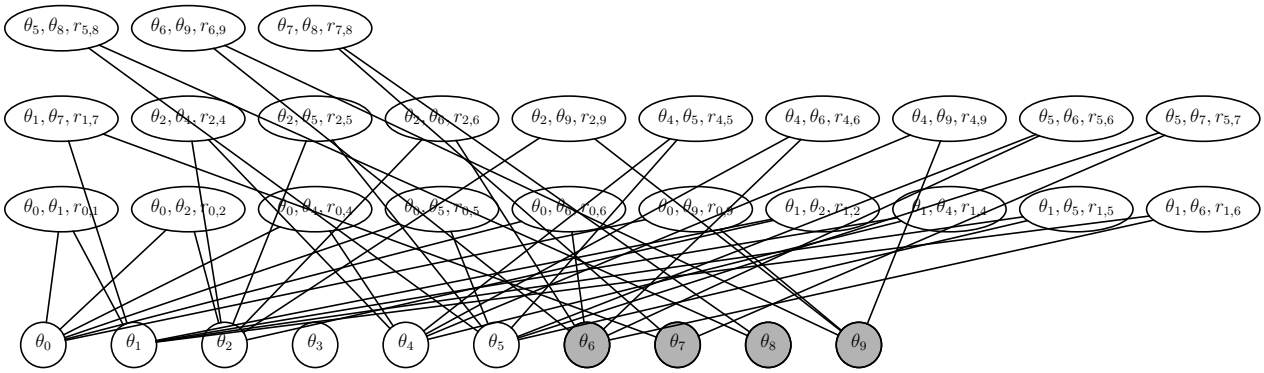

Fig. 2. A Bethe cluster graph of the network in Fig. 1.

be represented by dimensional vectors of cardinality D. The integral in (6) becomes a matrix-vector product and this in general requires $\mathcal{O}(D^2)$ operations [28], making the problem intractable because the dimensions of the grid and the resolution of the required localization will increase quadratically. This makes the use of approximation methods necessary.

In the literature, various methods have been proposed, such as particle filtering methods [29, 30], Monte-Carlo methods [11], and the more general NBP, e.g., [9, 31, 32]. They try to approximate the cpdf by a Gaussian mixture of particles, achieving a complexity of $\mathcal{O}(L^2)$, where L denotes the number of particles, and $L \ll D$. If the density of the number of particles is to remain approximately the same in the 3D case as in 2D, then the number of particles will increase by a factor of $2d_{ji}$ (d_{ji} being the exact distance). This phenomenon is known as the curse of dimensionality [33]. This would require finding the product of large Gaussian mixtures, which is already an expensive operation and can quickly become prohibitively expensive. Also it requires a considerable amount of overhead, since the parameters for all the components of each particle message need to be communicated. Another important issue is the convergence of the algorithm, although this paper does not directly tackle this issue. Due to the loopy characteristics of the graph, beliefs of factors will move around closed loops, creating opportunities for some factors to overpower their beliefs and oscillations to occur in the graph, essentially affecting the convergence. In practice, most nodes would actually converge to a local optimum with only a few nodes oscillating unable to converge [14]. In the next section, we will present HEVA to overcome those critical issues.

III. HEVA

A. Overview

HEVA is a hybrid method that combines elements of both parametric and non-parametric approximations for optimizing the local computational cost at each node, the communication overhead, and the convergence speed. In HEVA, the factors of $\psi(\mathbf{\Theta}_i)$ are first approximated by a Gaussian mixture, while the factors of the cpdf's $\psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, R_{ji} = r_{ji})$ are approximated by weighted particles with a spherical Gaussian kernel density. Two novel filters are proposed here. The first filter provides NLoS mitigation by removing the messages with positive bias characteristics in their respective ranging measurements. The second one is designed to intelligently decrease the number of particles in each mixture in order to minimize calculations and is defined as an ellipsoid filter which utilizes the intuition that the location of a node will be close to the intersections between two belief cpdfs. Finally, after calculating the weights of the particles, i.e., utilizing mixture importance sampling (MIS) to solve the product in (7), we use a clustering technique called variational Bayes (VB) to obtain the parameters of the Gaussian mixture $\psi(\mathbf{\Theta}_i)$. In so doing, only the parameters of the Gaussian mixture distribution require transmission.

The algorithm is summarized in Algorithm 1. It begins by all nodes broadcasting their initial beliefs. In the first iteration, the agents have a uniform distribution and hence skip transmitting their beliefs, as it would not add any useful information. As more iterations pass, agents begin to have non-uniform beliefs about their locations and start to transmit. After each agent receives the messages (7) from their neighbors, the next step is to pass them from the NLoS mitigation filter.

B. NLoS Mitigation Filter

In order to understand the effect of the NLoS mitigation filter, it is important to understand the impact of NLoS propagation in ToA measurements. ToA measurements are affected by positive bias and removing biased messages does not affect the CRLB of the localization error, c.f. [34]. Assuming a simple 2D example with three anchors communication with one agent, it is easy to visualize that in the case of no noise, the incoming message "ring"-shaped pdfs will combine to a single Gaussian centred on the intersection of the messages. Alternatively, if one anchor is NLoS, then due to the positive bias, the measured distance estimate will account to a larger

Algorithm 1 HEVA

Initialize beliefs $p^{(0)}(\mathbf{\Theta}_i) \forall i \in$ Nodes
for $t = 0$ to T **do**
  for all $i \in$ Nodes **do**
    Broadcast current belief $p^{(t)}(\mathbf{\Theta}_i)$
    for all $j \in \mathcal{N}_i$ **do**
      Collect distance estimates $r_{ji}^{(t)}$
    end for
  end for
  Initialize $\psi(\mathbf{\Theta}_i) = p^{(t)}(\mathbf{\Theta}_i)$
  Initialize

$$\psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, R_{ji} = r_{ji}) = p^{(t)}(R_{ji} = r_{ji}^{(t)} | \mathbf{\Theta}_i, \mathbf{\Theta}_j)$$

  repeat
    for all $i \in$ Nodes **do**
      for all $j \in \mathcal{N}_i$ **do**
        Receive $b_j^{(s)}(\mathbf{\Theta}_j)$
        if $b_j^{(s)}(\mathbf{\Theta}_j)$ passes NLOS filter (Algorithm 2) **then**
          Calculate $\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i)$ using Spherical Gibbs sampling (Algorithm 3)
        end if
      end for
      Calculate $b_i^{(s+1)}(\mathbf{\Theta}_i)$ using the proposed ellipsoid filter (Algorithm 4) and VB clustering
      Check for convergence
      Broadcast $b_i^{(s+1)}(\mathbf{\Theta}_i)$
    end for
    s = s+1
  until all messages have converged or the maximum number of iterations is reached
  Update belief $p^{(t+1)}(\mathbf{\Theta}_i)$, using (8)
end for

radius in the "ring"-shaped pdf message skewing the centre of the Gaussian further away from the intersection, and possibly creating new components in the Gaussian mixture.

To mitigate that, the proposed NLoS filter checks all incoming messages in the following manner. Each node compares the distance between itself and the corresponding neighbour computed using the current beliefs of both nodes with the range estimate. If it is larger than the estimate, the corresponding marginal (6) will be calculated and used in estimating the product (7); otherwise it will be dropped. The intuition behind is simple. As stated in Section II, NLoS measurements are affected by a positive bias, which means that they will be greater than the real estimates. Let the estimated position of node i be $\hat{\boldsymbol{\theta}}_i^{(t)}$, that of node j be $\hat{\boldsymbol{\theta}}_j^{(t)}$, and the respective distance measurement be $r_{ji}^{(t)}$. As a result, if $\|\hat{\boldsymbol{\theta}}_i^{(t)} - \hat{\boldsymbol{\theta}}_j^{(t)}\| \geq r_{ji}^{(t)}$, then the corresponding messages will be calculated as normal; otherwise, the message from node j will be dropped.

To take full advantage of the information in the distribution, the following idea is proposed. Firstly, we draw L weighted samples $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^L$ from $b_j^{(s)}(\mathbf{\Theta}_j)$ $\forall j \in \mathcal{N}_i$ and another L weighted samples $\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^L$ from $\psi(\mathbf{\Theta}_i)$. Given the samples from the belief pdfs in $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{j=1,l=1}^{|\mathcal{N}_i|,L}$ and $\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^L$, the convex hull of each node set is calculated and the maximum distance between the two convex hulls of node i and any of its neighbors, node j, is compared with the distance measurement $r_{ji}^{(t)}$. This is done by using the maxdist algorithm in [35]. For $j \in \mathcal{N}_i$, if

$$\mathsf{maxdist}\Big(\mathsf{convex\_hull}(\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^L), \mathsf{convex\_hull}(\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^L)\Big) \geq r_{ji}^{(t)}, \quad (9)$$

then the message is kept and added to the set $\mathcal{Q}_i$; otherwise, it is dropped. Note that $\mathcal{Q}_i$ will hold only the messages that will be used in subsequent calculations to reduce the complexity. By using this condition, NLoS messages will be used in the first iterations of the algorithm where there is not enough information about the belief of the nodes, but in the later iterations, NLoS messages will tend to be dropped. It should be noted that no effort is made to identify if the message is NLoS or not by using any NLoS identification technique. This means that there is a probability that both LoS message could be ignored and NLoS message might pass the filter. The NLoS filter algorithm is summarized in Algorithm 2.

Algorithm 2 NLoS Filter

Require: non-uniform $b_{i\to\forall j\in\mathcal{N}_i}^{(s)}(\mathbf{\Theta}_i)$
Sample $\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^L \sim b_{i\to\forall j\in\mathcal{N}_i}(\mathbf{\Theta}_i)$
Initialize $\mathcal{Q}_i$ to empty
for all $j \in \mathcal{N}_i$ **do**
  Sample $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^L \sim b_{j\to i}^{(s)}(\mathbf{\Theta}_j)$
  CH_i=convex_hull($\{w_i^{(l)}, \boldsymbol{\theta}_i^{(l)}\}_{l=1}^L$)
  CH_j=convex_hull($\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^L$)
  if $\mathsf{maxdist}(\mathsf{CH}_i, \mathsf{CH}_j) > r_{ji}^{(t)}$ **then**
    Add $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^L$ to $\mathcal{Q}_i$
  end if
end for
return $\mathcal{Q}_i$

C. Filtering Operation: $\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i)$

The next step of Algorithm 1 (HEVA) is for each node, say node i, to compute the received messages $\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i)$ from its neighbors, where $\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i))$ is the product of the distance cpdf with the neighbor location belief integrated over the neighbourhood variable $\mathbf{\Theta}_j$, for BP iteration $(s+1)$, i.e.,

$$\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i) = \int \frac{\psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, R_{ji} = r_{ji}^{(t)}) b_{j\to i}^{(s)}(\mathbf{\Theta}_j)}{\delta_{i\to j}^{(s)}(\mathbf{\Theta}_j)} d\mathbf{\Theta}_j \quad (10)$$

with $\psi(\mathbf{\Theta}_i, \mathbf{\Theta}_j, r_{ji} = r_{ji}^{(t)}) = p^{(t)}(r_{ji} = r_{ji}^{(t)} | \mathbf{\Theta}_i, \mathbf{\Theta}_j)$. Although node i has received the beliefs $b_j^{(s)}(\mathbf{\Theta}_j)$ and obtained the measurements $r_{ji}^{(t)}$ for deriving the cpdf, for complexity reasons, this will be done by using particle filtering. Specifically, given the set of particles $\mathcal{Q}_i$, for each subset j of particles $\{w_j^{(l)}, \boldsymbol{\theta}_j^{(l)}\}_{l=1}^L$, we calculate a set of parameters

$$\mathcal{G}_{j\to i} \triangleq \left\{ w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \mathbf{\Sigma}_{j\to i} \right\}_{l=1}^L \quad (11)$$

that approximate

$$\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i) \simeq \sum_l w_{j\to i}^{(l)} \mathcal{N}(\boldsymbol{\Theta}_i; \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}), \qquad (12)$$

where $w_{j\to i}^{(l)}$ is a weighting factor and $\mathcal{N}(\boldsymbol{\Theta}_i; \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i})$ refers to a Gaussian distribution in $\boldsymbol{\Theta}_i$ with mean vector $\boldsymbol{\mu}_{j\to i}^{(l)}$ and covariance matrix $\boldsymbol{\Sigma}_{j\to i}$. As such, the estimated lth sample $\boldsymbol{\theta}_i^{(l)}$ will be close to the surface of a sphere with radius $r_{ji}^{(t)}$ around the sample $\boldsymbol{\theta}_j^{(l)}$ and the mean vector for the mixture $\delta_{j\to i}^{(s+1)}(\boldsymbol{\Theta}_i)$ is given by

$$\boldsymbol{\mu}_{j\to i}^{(l)} = \boldsymbol{\theta}_j^{(l)} + r_{ji}^{(t)} + \nu^l \begin{bmatrix} \sin(\rho^{(l)})\cos(\phi^{(l)}) \\ \sin(\rho^{(l)})\sin(\phi^{(l)}) \\ \cos(\rho^{(l)}) \end{bmatrix}, \qquad (13)$$

where $\phi^{(l)} \sim \mathcal{U}[0, 2\pi), \rho^{(l)} \sim \mathcal{U}[-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\nu^{(l)} \sim p_\nu$, in which $\mathcal{U}(\cdots)$ denotes a uniform distribution and p_ν is the noise pdf, with a standard deviation of σ_ν. In addition, a covariance matrix $\boldsymbol{\Sigma}_{j\to i}$ is assigned to all Gaussians (independent of l) and is calculated as

$$\boldsymbol{\Sigma}_{j\to i} = \frac{1}{5.991} \left[\frac{4\sigma_\nu \left(3(r_{ji}^{(t)})^2 + 4\sigma_\nu^2 \right)}{L} \right]^{\frac{2}{3}} \mathbf{I}. \qquad (14)$$

The analysis leading to the above expression is given in the appendix. On the other hand, the weight of the lth sample is given by

$$w_{j\to i}^{(l)} \propto \frac{w_j^{(l)}}{\delta_{i\to j}^{(s)}(\boldsymbol{\theta}_j^{(l)})}. \qquad (15)$$

This can be considered as the belief pdf of node j with the influence of node i from the previous iteration being removed (i.e., $\delta_{i\to j}^{(s)}(\boldsymbol{\Theta}_j)$), in order to avoid overpowering of a node's belief due to loops [31]. Alternatively, we can concentrate the samples taking advantage of the angle in the same manner as Parsimonious NBP, c.f. [31]. We call this this variation of HEVA as Parsimonious HEVA, or PHEVA.

After calculating the approximations of all cpdfs for all the incoming messages, we add all sets $\mathcal{G}_{j\to i}$ to the superset $\mathcal{G}_i$. This set can be viewed as $|\mathcal{N}_i|$ Gaussian mixtures with L components each, and will be used to calculate the product of all incoming beliefs with the belief of node i (7). The steps are summarized in Algorithm 3. Note that the terms "component" and "particle" will be used interchangeably thereafter.

D. Product Operation: Gaussian Mixture Product Calculation

A commonly used approach to prevent the huge computational cost for evaluating the product of Gaussian mixtures is the use of the MIS technique [10]. As each Gaussian mixture in the product (7) has L components, the computational cost will be proportional to L^2 [10]. Unfortunately, even in the case of MIS, the cost can become prohibitive in the 3D case, because of the increased number of components per mixture, as discussed in Section I. To avoid this, we propose the use of a novel filter on the particles of the incoming messages, minimizing the total number of particles in MIS and hence the number of calculations, thereby making MIS feasible.

Algorithm 3 Spherical Gibbs Sampling

```
Initialize G_i as an empty set
for all j ∈ Q_i do
    Sample {φ^(l)}_{l=1}^L ~ U[0, 2π]
    Sample {ρ^(l)}_{l=1}^L ~ U[-π/2, π/2]
    Sample {ν^(l)}_{l=1}^L ~ N(0, σ_ν^2)
    Calculate the mean vector, using (13)
    Calculate the covariance matrix Σ_{j→i}, using (14)
    Calculate the weights w_{j→i}^(l), using (15)
    Add {w_{j→i}^(l), μ_{j→i}^(l), Σ_{j→i}}_{l=1}^L to G_i
end for
return G_i
```

The filter takes advantage of the intuition that the relevant particles should be close to the intersection of the "sphere"-shaped particle sets. This intersectional area can be enclosed inside an ellipsoid, and therefore the filter is referred to as an ellipsoid filter. Fig. 3 shows a 2D example of the filter. As the majority of the particles are not near the ellipsoid, and in no way near the intersections, they can be safely removed to avoid a huge amount of unnecessary computations.

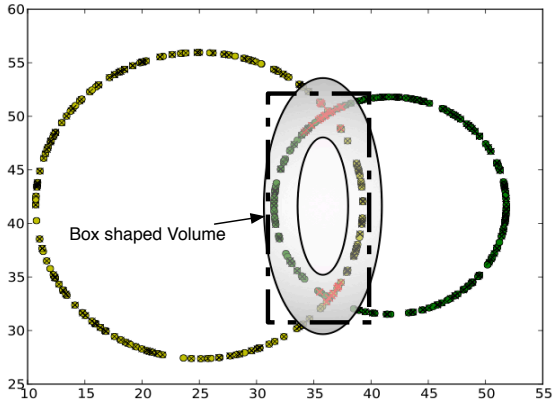

Fig. 3. A 2D example showing the ellipsoidal filter in operation. The mean and covariance for the ellipsoid are calculated based on the particles inside the box, and then resampling with replacement is done for each message proportional to their distance from the ellipsoidal area.

The proposed ellipsoidal filter is a "soft" decision probabilistic filter. Instead of simply removing all particles that are outside the filter area, it weights them based on their Euclidean distance from the ellipsoid and then resamples the $V = \alpha L$ particles from each message, where $0 < \alpha < 1$. This gives the filter a greater flexibility allowing to consider for situations, where the real location is further away from the intersection due to high noise. We define a box-shaped volume as shown in Fig. 3, with the edges defined by the coordinates:

$$\begin{cases} x_{\min} = \max_j(\min_l(x_j^{(l)})), \\ y_{\min} = \max_j(\min_l(y_j^{(l)})), \\ z_{\min} = \max_j(\min_l(z_j^{(l)})), \end{cases} \text{and} \begin{cases} x_{\max} = \min_j(\max_l(x_j^{(l)})), \\ y_{\max} = \min_j(\max_l(y_j^{(l)})), \\ z_{\max} = \min_j(\max_l(z_j^{(l)})). \end{cases} \tag{16}$$

From the superset $\mathcal{G}_i$, we select all the particles that reside inside the box. From these particles, we calculate the mean vector $\boldsymbol{\mu}_{\mathsf{box}}$ and the covariance matrix $\boldsymbol{\Sigma}_{\mathsf{box}}$. Then we calculate the ellipsoid weight for every particle $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\}_{l=1}^{L}$ in every message set $\mathcal{G}_{j\to i}$, by using the mean of the particle in the following function and normalize the weights for each message set. Mathematically, that is,

$$w_{\mathsf{e},\boldsymbol{\mu}_{\mathsf{j}\to\mathsf{i}}}^{(l)} = w_{\mathsf{ellipsoid}}(\boldsymbol{\mu}_{j\to i}^{(l)}) \propto \exp\left(-\frac{\gamma}{2}\left(\sqrt{(\boldsymbol{\mu}_{j\to i}^{(l)} - \boldsymbol{\mu}_{\mathsf{box}})^T \boldsymbol{\Sigma}_{\mathsf{box}}^{-1}(\boldsymbol{\mu}_{j\to i}^{(l)} - \boldsymbol{\mu}_{\mathsf{box}})} - 1\right)^2\right), \tag{17}$$

where the variable γ is defined as the filter precision, and it controls the steepness of the filter. We add the relevant weight to every element in the superset $\mathcal{G}_i$:

$$\left\{\left\{w_{\mathsf{e},\boldsymbol{\mu}_{\mathsf{j}\to\mathsf{i}}}^{(l)}, w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\right\}_{l=1}^{L}\right\}_{j\in\mathcal{N}_i}. \tag{18}$$

After that, V particles are randomly sampled with replacement, from each message set $\mathcal{G}_{j\to i}$, according to their weights (17), giving a new particle set $\{w_{j\to i}^{(v)}, \boldsymbol{\mu}_{j\to i}^{(v)}, \boldsymbol{\Sigma}_{j\to i}^{(v)}\}_{v=1}^{V}$ that is added to the set $\mathcal{J}_i$. The ellipsoid filter is summarized and formally described in Algorithm 4.

Algorithm 4 Ellipsoid Filter

1: Given all particles $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\}_{l=1}^{L} \in \mathcal{G}_i$, find (16)
2: Calculate box borders based on $\begin{bmatrix} x_{\min} & x_{\max} \\ y_{\min} & y_{\max} \\ z_{\min} & z_{\max} \end{bmatrix}$
3: Calculate $\boldsymbol{\mu}_{\mathsf{box}} = \mathsf{mean}(\boldsymbol{\mu}_{j\to i}^{l})\ \forall\ \boldsymbol{\mu}_{j\to i}^{l}$ inside box
4: Calculate $\boldsymbol{\Sigma}_{\mathsf{box}} = \mathsf{covariance}(\boldsymbol{\mu}_{j\to i}^{l})\ \forall\ \boldsymbol{\mu}_{j\to i}^{l}$ inside box
5: Weight all samples using (17)
6: Initialize $\mathcal{J}_i$ to be an empty set
7: **for all** $j \in \mathcal{Q}_i$ **do**
8: Re-sample from $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\}_{l=1}^{L}$ proportionally to their weights $\{w_{\mathsf{e},\boldsymbol{\mu}_{\mathsf{j}\to\mathsf{i}}}^{(l)}\}_{l=1}^{L}$, aL times with replacement.
9: Add $\{w_{j\to i}^{(l)}, \boldsymbol{\mu}_{j\to i}^{(l)}, \boldsymbol{\Sigma}_{j\to i}\}_{l=1}^{aL}$ to $\mathcal{J}_i$
10: **end for**
11: **return** $\mathcal{J}_i$

Additionally, we draw V samples from $\psi(\boldsymbol{\Theta}_i)$ to complete the calculation (7). Essentially, we have a new product of Gaussian mixtures, but with V components, instead of L. In order to compute the product, we employ MIS [10]. From each message $j \in \mathcal{N}_i$, we draw V samples $\boldsymbol{\theta}_j^{(v)}$ and weight them by

$$w_j^{(v)} = \frac{\psi(\boldsymbol{\theta}_j^{(v)}) \prod_{j\in\mathcal{N}_i} \delta_{j\to i}^{(s+1)}(\boldsymbol{\theta}_j^{(v)})}{\psi(\boldsymbol{\theta}_j^{(v)}) + \sum_{j\in\mathcal{N}_i} \delta_{j\to i}^{(s+1)}(\boldsymbol{\theta}_j^{(v)})} \frac{1}{w_{\mathsf{ellipsoid}}(\boldsymbol{\theta}_j^{(v)})}. \tag{19}$$

It should be noted that we divide by the weight $w_{\mathsf{ellipsoid}}(\boldsymbol{\theta})$ to cancel out the asymptotic effect of drawing samples from the ellipsoid filter. After that, L particles are sampled with replacement from the $(|\mathcal{N}_i|+1)V$ samples $\{w_j^{(v)}, \boldsymbol{\theta}_j^{(v)}\}_{j=1,v=1}^{|\mathcal{N}_i|+1,V}$ proportional to the weights $w_j^{(v)}$, to give $\{\boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$. As we keep only V particles for each incoming message, the computational cost can be drastically reduced, while the computational cost of weighting all the particles is linear with the total number of particles, having complexity of $\mathcal{O}(V^2(|\mathcal{N}_i|+1))$ compared to the $\mathcal{O}(L^2(|\mathcal{N}_i|+1))$ if using all components. The MIS algorithm is summarized in Algorithm 5.

Algorithm 5 Mixture Importance Sampling (MIS)

for all $j \in \mathcal{J}_i$ **do**
  Initialize $p_{i\to j}(\boldsymbol{\theta}_i) = \sum_{k=1}^{aL} w_{j\to i}^{(k)} \mathcal{N}\left(\boldsymbol{\theta}_i, \boldsymbol{\mu}_{j\to i}^{(k)}, \boldsymbol{\Sigma}_{j\to i}\right)$
end for
Initialize $q_i(\theta_i) = \frac{1}{|\mathcal{J}_i|} \sum_{j'\in\mathcal{J}_i} p_{i\to j'}(\boldsymbol{\theta}_i)$
Draw KaL samples $\{\boldsymbol{\theta}_i^{(n)}\}_{n=1}^{KaL} \sim q_i(\boldsymbol{\theta}_i)$, where $K > 1$
Weight them by $w_i^{(n)} = \frac{\prod_{j'\in\mathcal{J}_i} p_{i\to j'}(\boldsymbol{\theta}_i^{(n)})}{q_i(\boldsymbol{\theta}_i^{(n)})}$
Re-sample from $\{w_i^{(n)}, \boldsymbol{\theta}_i^{(n)}\}_{n=1}^{KaL}$ proportionally to their weights, aL times with replacement.

As such, we have a particle approximation of $b_i^{(s+1)}(\boldsymbol{\theta}_i)$. From this current belief, we randomly choose λL samples and $(1-\lambda)L$ samples from the calculated belief in the previous iteration, in order to estimate the dampened belief as in (7).

Given the set of the L particles that approximate $b_i^{(s+1)}(\boldsymbol{\theta}_i)$, the final step is to convert the non-parametric kernel representation of the belief in a parametric form using a Gaussian mixture with parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)$, mean vectors $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$ and covariance matrices $\{\boldsymbol{\Sigma}_k\}_{k=1}^{K}$ so that

$$\delta_{i\to\forall j}^{\text{lower}\to\text{upper}}(\boldsymbol{\theta}_i) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{\theta}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{20}$$

where K is regarded as the number of clusters. The optimization of the parameters is addressed in the next subsection.

E. VB

VB (variational Bayes) is a Bayesian learning algorithm that finds the parameters that best fit an approximate distribution to a dataset [36]. The idea is similar to the well-known EM algorithm but with the major difference that the parameters of the mixtures are considered themselves stochastic variables derived from prior distributions.

The advantage of this is threefold. First, singularity issues encountered in EM can be completely avoided. Also, there is no issue of fitting too many clusters, as even if a large number of initial clusters are initially considered, the superfluous ones

will degenerate to zero through self-optimization. This allows the algorithm to start with a large K that will decrease as the algorithm progresses and more information received resolves ambiguities. Essentially, this gives a compromise among the message size, computational cost and information loss. Finally, the optimal number of clusters can be obtained without using techniques such as cross-validation. For more information, interested readers are referred to [36, Chapter 10]. These prior pdfs are chosen to be conjugate priors of the corresponding distributions in order to facilitate the derivations.

In VB, the parameters in (20) are considered stochastic and their priors are a Dirichlet distribution for $\boldsymbol{\pi}$ and a Gaussian-Wishart distribution for $\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k$, where $\boldsymbol{\Lambda}_k = \boldsymbol{\Sigma}_k^{-1}$. That is,

$$p(\boldsymbol{\pi}) = \mathsf{Dir}(\boldsymbol{\pi}|\mathbf{a}) = C(\mathbf{a}) \prod_{k=1}^{K} \pi_k^{a_k - 1} \tag{21}$$

$$p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\boldsymbol{\mu}|\boldsymbol{\Lambda}) p(\boldsymbol{\Lambda}) \tag{22}$$

$$= \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{\mu}_k|\mathbf{m}_k, (\beta_k \boldsymbol{\Lambda}_k)^{-1}) \mathcal{W}(\boldsymbol{\Lambda}_k|\mathbf{W}_k, \nu_k), \tag{23}$$

where $\mathbf{a} = \{a_1, \ldots, a_K\}$ can be interpreted as the effective prior number of observations associated with each component, $\mathbf{W}_k$ is a positive definite matrix, and ν_k denotes the number of degrees of freedom of the distribution for the kth cluster. The parameters of the priors are called hyper-parameters. In the rest of the section the steps of the algorithm are presented, but for a complete derivation of the algorithm, see [36].

The method to optimize the parameters in (20) is iterative and cycles between two steps, similar to the EM steps, with some initialization of the hyper-parameters:

$$a_k = a_0 \ \forall k, \tag{24}$$

$$\beta_k = \beta_0 \ \forall k, \tag{25}$$

$$\mathbf{m}_k = \mathbf{m}_0 \ \forall k, \tag{26}$$

$$\mathbf{W}_k = \mathbf{W}_0 \ \forall k, \tag{27}$$

$$\nu_k = \nu_0 \ \forall k. \tag{28}$$

In the first step, referred to as the E-step, the parameters of the distributions are considered constant and the responsibilities of each component (cluster k) for each sample (particle $\boldsymbol{\theta}_l$ obtained in Section III-D), denoted as r_{lk}, are calculated. Very briefly, r_{lk} indicates how probable it is for particle l to belong to cluster k. In more detail, the E-step calculates

$$\ln \tilde{\boldsymbol{\Lambda}}_k = \sum_{i=1}^{F} \psi\left(\frac{\nu_k + 1 - i}{2}\right) + F \ln 2 + \ln \mathsf{det}(\mathbf{W}_k), \tag{29}$$

$$\ln \tilde{\pi}_k = \psi(a_k) - \psi\left(\sum_{k=1}^{K} a_k\right), \tag{30}$$

$$r_{lk} \propto \tilde{\pi}_k \tilde{\boldsymbol{\Lambda}}_k^{\frac{1}{2}} \exp\left\{-\frac{F}{2} - \frac{\nu_k}{2}(\boldsymbol{\theta}_l - \mathbf{m}_k)^T \mathbf{W}_k (\boldsymbol{\theta}_l - \mathbf{m}_k)\right\}, \tag{31}$$

where $\psi(\cdot)$ is the digamma function, and F is the number of features. In 3D cases, $F = 3$ but in 2D, $F = 2$.

In the M-step, given the responsibilities r_{lk}, we recalculate the parameters for maximizing the log-likelihood by

$$L_k = \sum_{l=1}^{L} r_{lk}, \tag{32}$$

$$\bar{\boldsymbol{\theta}}_k = \frac{1}{L_k} \sum_{l=1}^{L} r_{lk} \boldsymbol{\theta}_l, \tag{33}$$

$$\mathbf{S}_k = \frac{1}{L_k} \sum_{l=1}^{L} r_{lk} (\boldsymbol{\theta}_l - \bar{\boldsymbol{\theta}}_k)(\boldsymbol{\theta}_l - \bar{\boldsymbol{\theta}}_k)^T, \tag{34}$$

$$a_k = a_0 + L_k, \tag{35}$$

$$\beta_k = \beta_0 + L_k, \tag{36}$$

$$\mathbf{m}_k = \frac{1}{\beta_k}(\beta_0 \mathbf{m}_0 + L_k \bar{\boldsymbol{\theta}}_k), \tag{37}$$

$$\mathbf{W}_k^{-1} = \mathbf{W}_0^{-1} + L_k \mathbf{S}_k + \frac{\beta_0 L_k}{\beta_0 + L_k} (\boldsymbol{\theta}_l - \mathbf{m}_0)(\boldsymbol{\theta}_l - \mathbf{m}_0)^T, \tag{38}$$

$$\nu_k = 1 + \nu_0 + L_k. \tag{39}$$

The algorithm continues to iterate between the E- and M-steps until either the parameters or the log-likelihood converge to meet a certain precision. Finally, the values of the optimized parameters in (20) are then chosen as the expectations of the corresponding distributions, i.e.,

$$\pi_k = \frac{a_k}{\sum_{k=1}^{K} a_k}, \tag{40}$$

$$\boldsymbol{\mu}_k = \mathbf{m}_k, \tag{41}$$

$$\boldsymbol{\Sigma}_k = (\nu_k \mathbf{W}_k)^{-1}. \tag{42}$$

It is well understood that VB has the advantage of achieving a high log-likelihood quickly, and gives a good parametric approximation of $p(\boldsymbol{\theta}_i)$. In HEVA, the distribution consists of a relatively large number of components evenly spread out in the area already calculated in for the ellipsoid filter. There are two advantages. In case there is a large space of possible locations, the large number of components will provide flexibility for the approximation to fit well with the data. In addition, if there is little or no ambiguity, most of the components degenerate to zero allowing for a small Gaussian mixture (i.e., small K).

F. Computational Convergence

A cluster graph is not guaranteed to achieve the optimal solution, but if the running intersection property and the family preservation property are held [14], typically most clusters will converge to a local optimum with just a few clusters oscillating unable to achieve convergence. HEVA uses a dampened belief propagation message passing, which empirically helps even the oscillating nodes converge to a local optimum. An added complexity is inserted in the algorithm because of the use of VB. For this reason, it is important to explicitly define a convergence criterion. As location beliefs are approximated by Gaussian mixtures, Kullback-Leibler (KL) divergence between the belief of the previous and present iterations is calculated. If the difference is below a prescribed threshold, the node is said to have converged to a solution. As there is no easy way to

calculate the KL divergence between two Gaussian mixtures, the approximation method proposed in [37] is used.

G. Complexity

The complexity of HEVA is dominated by three processes: (1) the filtering operation, (2) the product operation and (3) the VB algorithm, i.e., the clustering operation. In order to analyze the complexity, we let

$|\mathcal{N}_i|$ the number of incoming messages to a specific node,
L the number of drawn particles,
α the ratio of particles kept in the product calculation,
S the number of components in the Gaussian mixture,
F the number of features,
I_{VB} the maximum number of iterations VB will run,
I_{HEVA} the number of iterations HEVA will run.

At every iteration of the algorithm, each node first draws particles from each incoming message, namely, the filtering operation, an operation that scales with the number of neighbors, and the number of drawn particles (therefore scales with complexity $L|\mathcal{N}_i|F$). Then the samples are passed through the ellipse function and every particle is weighted. Thus, this operation scales with $L|\mathcal{N}_i|F$. The next step is to draw αL samples for each message with replacement, with a complexity of $\alpha L|\mathcal{N}_i|$. Finally the MIS algorithm is used to calculate the product. This operation's complexity scales with $(\alpha L)^2|\mathcal{N}_i|F$ [10]. Finally, a parametric form is found with VB, an operation that scales with $\alpha L S I_{\mathsf{VB}} F^3$ [36]. The computational complexity of the above operations is summarized in Table I.

Next we compare HEVA with various alternative algorithms used for distributed cooperative localization. In NBP and its non-parametric variants, the squared L factor in calculating MIS makes the product operation (7) computationally dominate the algorithm. As such, the complexity cost is bounded by $\mathcal{O}(L^2|\mathcal{N}_i|F)$. NBP variations like Parsimonius NBP (PNBP) [31] and Box-NBP (BNBP) [32] aim to decrease the number of particles L required, by focusing the energy mass of the pdfs, but the computational cost remains dominated by (7). Similarly they are also computationally bounded by $\mathcal{O}(L^2|\mathcal{N}_i|F)$.

In HEVA, we make α small to decrease the computational cost of the product operation. By choosing a suitably low α, the decrease in computational cost will be $\frac{(\alpha L)^2}{L^2} \Rightarrow \alpha^2$. For example, if $\alpha = 0.2$, there will be more than 96% decrease in complexity for the message product operation. Essentially this ameliorates the bottleneck of the product operation. This lowers the complexity bound of HEVA by almost an order of L approaching aproximately $\mathcal{O}(L|\mathcal{N}_i|F)$, if one chooses $\alpha^2 \approx \frac{1}{L}$. Finally, for PHEVA, as will be shown in the sequel, the focused sampling provides betters results for a given number of particles L, albeit at a computational cost.

In Fig. 4, we present the average simulation time for a 3D network with 25 nodes, and average connectivity 4.9 for different numbers of L. Computation was done on an Intel core 2 duo at 2.33GHz. All algorithms were implemented in python using the numpy and scipy libraries [38]. For each scenario, 100 iterations were run and the average time was calculated.

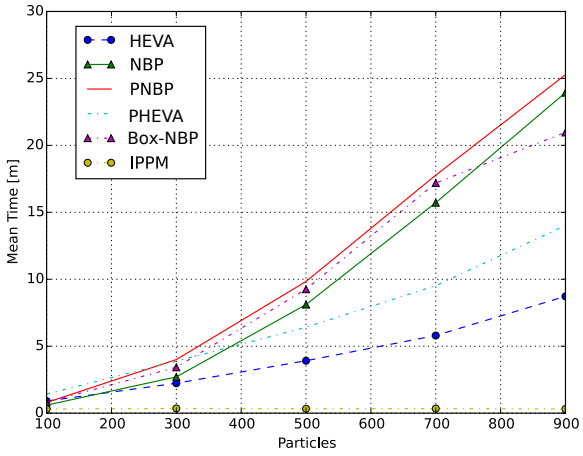

Fig. 4. The average time versus the number of particles when the average node connectivity is 4.9.

TABLE I
COMPUTATIONAL COMPLEXITY OF HEVA STEPS FOR A EACH AGENT (SAY THE i-TH)ON ONE ITERATION

Computation	Cost
The following operations are repeated I_{HEVA} times	
Draw samples from messages	$L\|\mathcal{N}_i\|F$
Weight using Ellipse function	$L\|\mathcal{N}_i\|F$
Draw weighted samples with replacement	$\alpha L\|\mathcal{N}_i\|$
Compute particle product	$(\alpha L)^2\|\mathcal{N}_i\|F$
Run VB	$\alpha L S I_{\mathsf{VB}} F^3$

As can be seen, for NBP, PNBP and BNBP, the processing time all increases quadratically with L, while HEVA is much smoother, and almost approaches a linear increase. In contrast, PNBP has a much steeper increase in processing time than HEVA, providing a compromise between the speed of HEVA, for slightly better accuracy. Finally, it can be seen that IPPM is much faster than the aforementioned family of belief message passing algorithms, but at a higher error rate. IPPM does not use particles, thus the mean time does not vary with the number of particles. Unfortunately, this provides lower accuracy and requires a larger communication overhead, in order to decide on the initial values before the optimization starts, as will be discussed in Section III-H. The complexity costs of the various algorithms are given in Table II.

TABLE II
COMPUTATIONAL COMPLEXITY FOR EACH AGENT (SAY THE i-TH) ON ONE ITERATION OF DIFFERENT ALGORITHMS

Algorithm	Complexity	Typical Values
NBP	$\mathcal{O}(L^2(\|\mathcal{N}_i\|))$	$L = 10^4 - 10^6$
PNBP	$\mathcal{O}(L^2(\|\mathcal{N}_i\|))$	$L = 10^2 - 10^3$
Box-NBP	$\mathcal{O}(L^2(\|\mathcal{N}_i\|))$	$L = 10^2 - 10^3$
HEVA	$\mathcal{O}((aL)^2\|\mathcal{N}_i\|)$	$L = 10^2 - 10^3$
PHEVA	$\mathcal{O}((aL)^2\|\mathcal{N}_i\|)$	$L = 10^2 - 10^3$
IPPM	$\mathcal{O}(\|\mathcal{N}_i\|)$	–

The average computational time for HEVA, and other algorithms is provided in Fig. 5 for different sizes of neighboring nodes. The same experiment was conducted but with varying

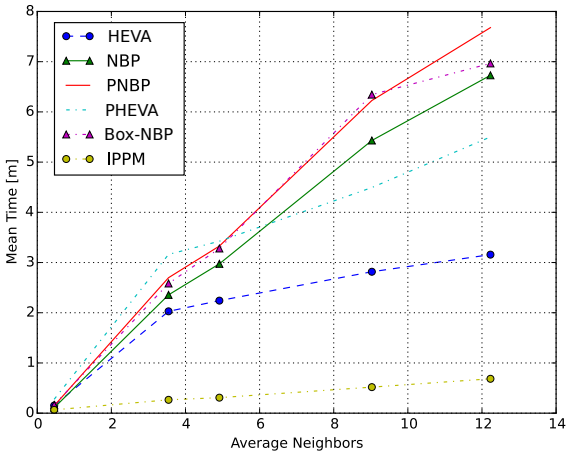

Fig. 5. The average simulation time versus mean node connectivity. Number of particles is $L = 300$.

communication range R, while keeping the number of particles constant $L = 300$, hence increasing the average number of neighbors. All techniques increase linearly with time, but the gradient of HEVA is much smaller $0.27s/$neighbor, compared to $0.77s/$neighbor for NBP, scaling much better when the average number of neighbors increases. The steepness that can be seen in the graph for less than 3 neighbors on average can be explained by the lack of computations as there are not enough neighbors to localize. The variations of NBP take similar time to NBP but require slightly more computations as they also need to calculate the angles of the samples.

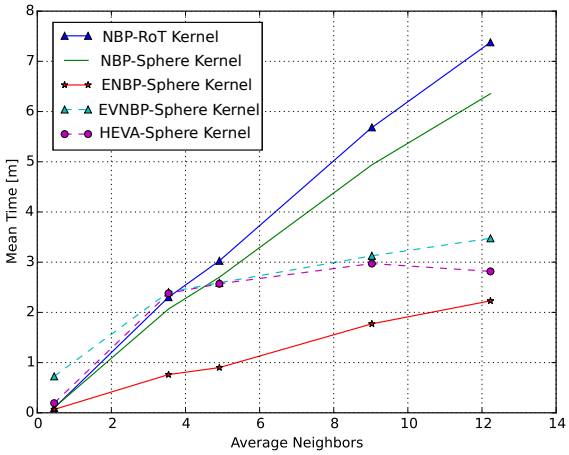

Fig. 6. The average simulation time versus mean node connectivity. Number of particles is $L = 300$.

In Fig. 6, results for the average computational time of NBP and HEVA are compared with the respective algorithms of adding to NBP one component of HEVA at a time, for the same 3D scenario as before. First, the new kernel is added, namely "NBP Sphere Kernel", decreasing the computational cost slightly. Then the ellipsoid filter is added, namely "ENBP" or "NBP+EF", which manages to decrease the cost considerably. Then if VB is applied, this will actually increase the computation cost, and finally the NLoS filter, which relatively keeps the cost equal to "EVNBP" or "NBP+EF+VB". It should be noted that as the number of average neighbors increases the computational time of HEVA decreases relative to "EVNBP", as messages will be dropped, from the NLoS filter.

H. Communication Overhead

In a distributed algorithm, most of the energy is spent on the local computations and broadcasting messages to one-hop neighboring nodes. In the literature focus is mostly given in the broadcasting part as the energy required for transmission is much higher than the energy required for a single floating point operation, c.f. [39]. Following the analysis in [24], we assume that all nodes are uniformly distributed over a 3-D unit square grid, and all real values are represented in double precision floating point format (64-bit precision). Hence, the total energy consumed for communication by any cooperative localization algorithm can be written as

$$\mathbb{E}(|\mathcal{V}|) = b(|\mathcal{V}|)h(|\mathcal{V}|)e(|\mathcal{V}|), \tag{43}$$

where $\mathbb{E}(\cdot)$ returns the expectation, $b(|\mathcal{V}|)$ denotes the total number of transmitted bits for $|\mathcal{V}|$ nodes, $h(|\mathcal{V}|)$ is the average number of hops required for transmitting one bit to the destination and $e(|\mathcal{V}|)$ is the average amount of energy required for transmitting one bit over one hop. As all communication is assumed to be done only by broadcasting to one-hop neighbors we set $h(|\mathcal{V}|) = 1$. The total number of real values transmitted in one iteration for one agent is approximately $L(1 + F)$ for non-parametric algorithms and $L_{\text{VB}}(1 + F + F^2)$ for HEVA, where L_{VB} is the average number of non zero components in VB. Therefore, we have that $b_{\text{NBP}}(|\mathcal{V}|) = \mathcal{O}(L(|\mathcal{V}|)(1 + F))$ and $b_{\text{HEVA}}(|\mathcal{V}|) = \mathcal{O}(L_{\text{VB}}(|\mathcal{V}|)(1 + F + F^2))$.

IPPM nodes on the contrary only transmit their point estimate, broadcasting F real values, which gives $b_{\text{IPPM}}(|\mathcal{V}|) = \mathcal{O}((|\mathcal{V}|)F)$. Given a fixed total number of nodes and ignoring $e(|\mathcal{V}|)$, which is assumed the same for all algorithms, we get the energy cost bounds presented in Table III.

TABLE III
ENERGY CONSUMED BY ALL NODES ON ONE ITERATION OF DIFFERENT ALGORITHMS

Algorithm	Energy Cost	Typical Values
NBP	$\mathcal{O}(L((\|\mathcal{V}\|))$	$L = 10^4 - 10^6$
PNBP	$\mathcal{O}(L((\|\mathcal{V}\|))$	$L = 10^2 - 10^3$
BNBP	$\mathcal{O}(L((\|\mathcal{V}\|))$	$L = 10^2 - 10^3$
HEVA	$\mathcal{O}(L_{\text{VB}}(\|\mathcal{V}\|))$	$L_{\text{VB}} = 10^0 - 10^1$
IPPM	$\mathcal{O}(\|\mathcal{V}\|)$	–

Note that as was shown in Section III-G, even though simply using ENBP is computationally cheaper than EVNBP and HEVA, by transmitting the belief in parametric form using VB. Based on the above analysis, there is a significant decrease in communication overhead. Simulations illustrate that for 3D positioning on average, VB uses $L_{VB} \simeq 4$ and assuming $L = 800$, as in Section IV, we have approximately a 98% decrease in communication overhead throughout the network. Even if NBP, used only $L = 100$ particles, the communication

cost of HEVA would still be approximately 87% less, showcasing the importance of transmitting a parametric form when minimizing communication throughput is the priority.

The ability to actually achieve consistently better accuracy than NBP and its variants, with a large decrease in computational cost, ameliorating the bottleneck of MIS, as well as a large decrease in communication overhead is the main contribution of HEVA, and important, as it paves the path of practically using probabilistic techniques in 3D localization.

IV. Simulation Results

In this section, we present simulation results for HEVA for cooperative localization. The root-mean-square (RMS) error is compared with those of NBP and Parsimonious NBP [9] as well as boxed NBP [32]. In experiments with NLoS edges we compare HEVA with IPPM [7] and ECM [24].

PNBP is a variant of NBP that uses the angle of particles between iterations to focus the sampling instead of using a uniform sampling. The equations from [9] have been extended for the 3D case in order for the simulation to work. BNBP is another variant that uses information of anchors in order to box the sampling in a specific area, in the same spirit as HEVA, even though in this case the limit is "hard", as all samples inside the box are taken, and all samples outside are dropped. ECM uses a distributed expectation conditional maximization algorithm while IPPM is a deterministic optimization method, based on the parallel projection method (PPM).

The RMS error can be defined as

$$\varepsilon_{\text{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}(\|\hat{\boldsymbol{\theta}}_i - \boldsymbol{\theta}_i\|^2)}, \tag{44}$$

where $\mathbb{E}(\cdot)$ denotes the expectation over noise realizations. As a summary, we will show that HEVA provides consistently better results than NBP and its variants with a large decrease in computational cost. In the 3D case, the advantages become much more pronounced, while NBP and its variants become computationally prohibitive. The biggest advantage of HEVA can be seen in the case of NLoS. We will define the edge NLoS probability as the probability that any network communication edge $e_{ji} \in \mathcal{E}$ is in NLoS for a given experiment, and present results for varying the edge NLoS probabilities. Finally, the variants of HEVA using EM and K-means will be shown, compared to our proposal of VB. Note that the higher errors here are due to the spherical Gaussian distributions for the anchor beliefs, thus increasing the inherent noise. The important thing to note is the comparison between algorithms.

We consider a three dimensional scenario of a $95 \times 95 \times 20m^2$ grid with $N = 125$ agents uniformly distributed over the grid and $M = 8$ anchors placed near the 8 corners of the grid. The maximum communication range is $15m$ for every node and the average node connectivity is $|\bar{\mathcal{N}}_i| \approx 6.1$. For each noise level, 300 independent simulations were run and the RMS error was calculated for all algorithms. Also, $L = 800$ and HEVA/PHEVA used a factor $\alpha = 0.2$.

Initially, a comparison will be made between NBP and the various components of HEVA. Thus, NBP is compared to NBP with the novel kernel, NBP with the ellipsoid filter, i.e., ENBP, ENBP with VB and finally the complete HEVA.

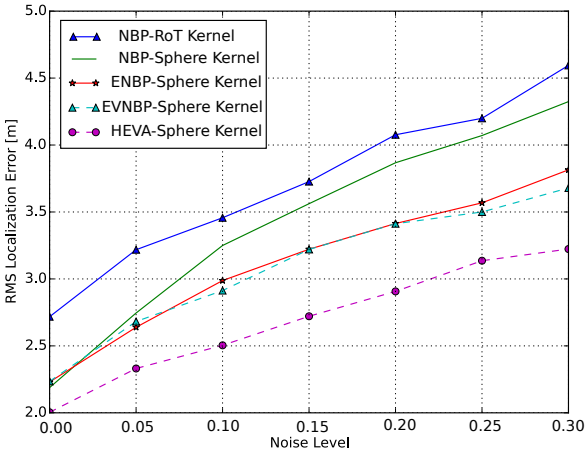

Fig. 7. The rms error versus the amount of range estimation noise K_e when the average node connectivity is 6.1 and no NLoS.

In Fig. 7, results are provided to show that an improvement is achieved as we add the various components of HEVA on NBP. Firstly we see an improvement by adding the kernel to NBP. It should be noted here, that in the 2D case, there is no noticeable improvement in the accuracy by using the kernel compared to the rule of thumb, but a reduction in the computational cost as the kernel covariance matrix is simpler to compute than the rule of thumb kernel covariance matrix. Also, we see that VB does not improve or deteriorate the results so is used purely in order to convert the pdf to a parametric form. Finally, it is also interesting to note an improvement by using the NLoS filter even when there are no NLoS edges in the network. This is explained as the filter does not only remove messages from NLoS edges but omits all messages that do not facilitate converge in the later iterations, hence helping convergence.

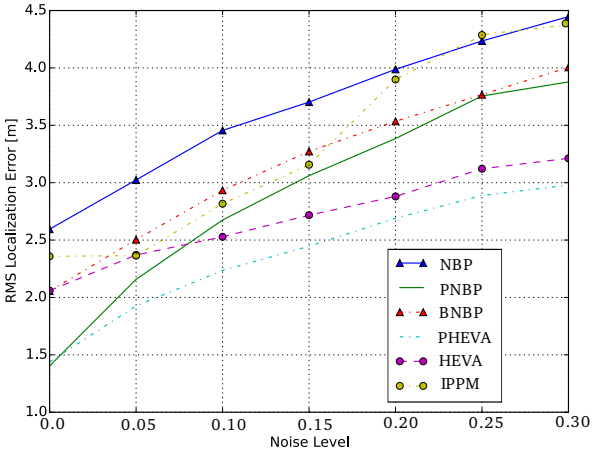

Fig. 8. The rms error versus the amount of range estimation noise K_e when the average node connectivity is 6.1 and no NLoS.

Next we compare HEVA with the aforementioned competing algorithms in Fig. 8. As expected, all algorithms that focus their particles outperform NBP. Further, HEVA and PHEVA outperform all other methods especially in higher noise levels, with HEVA achieving almost equal accuracy to PHEVA for much less computational cost as was shown in Fig. 5.

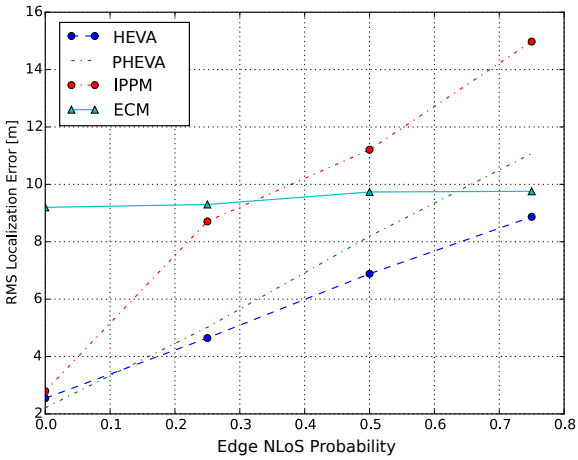

Fig. 9. The average error versus the Edge NLoS Probability, when the average node connectivity is 6.1 .

In Fig. 9, we compare the algorithms for various edge NLoS probabilities. HEVA and PHEVA are compared to IPPM and ECM [24], both of which have NLoS mitigation capabilities. As can be seen, both HEVA and PHEVA outperform the other algorithms even in very high NLoS scenarios. As ECM tries to approximate the noise pdf fitting a non-parametric pdf to the measured distances, it manages to keep the RMS error constant at all edge NLoS probabilities. Finally, it is interesting to note that in the case of high edge NLoS probability, HEVA outperforms PHEVA. We believe that as the high noise creates more ambiguity, it is easier for PHEVA to initially focus the particles in the wrong regions, increasing the RMS error.

A. Comparison with Other Clustering Techniques

As there is a number of possible clustering techniques that could be used instead of VB, in this subsection, we provide simulation results for two other clustering methods, namely, K-means and EM [36] for comparison. Fig. 10 provides the average RMS error results against the number of iterations. As is expected, VB and EM have almost the same results, for both the error cdfs and the average RMS errors and they both outperform K-means. A close observation for the results further shows that VB performs slightly better than EM. As explained earlier in Section III, the advantage of VB over EM is not only performance but VB can avoid singularity issues, and can automatically optimize its clustering even if it starts with an arbitrary large number of clusters.

V. Conclusions

In this paper, we have presented a novel cooperative localization method which we refer to it as HEVA. Based on NBP,

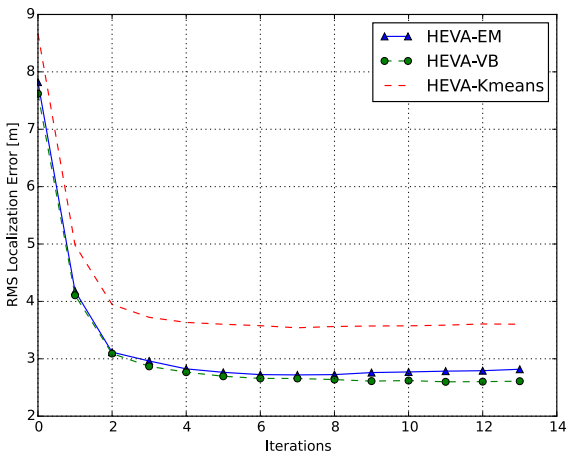

Fig. 10. The average RMS error versus the number of iterations with $K_e = 0.1$

HEVA uses intelligent filtering of the particles to both improve the localization accuracy and decrease the computational cost. Through VB, it also decreases drastically the communication overhead, without sacrificing the convergence speed or accuracy. It has been shown that it outperforms all other methods, and offers considerable computational advantages to existing methods. It was also shown to consistently achieve accurate position estimates even in high noise scenarios with NLoS, even in the more ambiguous 3D case. In the future, we will incorporate node mobility, and a number of different kernels will be considered to reduce the cost even further.

Appendix

In order to find a good approximation for the particle kernel covariance the following novel idea is suggested. Assuming a message from an anchor, whose location is precisely known, we can formulate $\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i)$ as

$$\delta_{j\to i}^{(s+1)}(\mathbf{\Theta}_i) = \frac{1}{\sqrt{2\pi}\sigma_\nu}\exp\left(\frac{-1}{2\sigma_\nu}\sqrt{x_i^2+y_i^2+z_i^2}-r_{ji}^{(t)}\right)^2. \quad (45)$$

Essentially, this is a Gaussian distribution and what we want is to approximate a spherical hull with a volume that captures the 0.95 confidence interval of the Gaussian. This spherical hull volume is equal to

$$V_c = \frac{4}{3}\pi\left[\left(r_{ji}^{(t)}+2\sigma_\nu\right)^3-\left(r_{ji}^{(t)}-2\sigma_\nu\right)^3\right]. \quad (46)$$

Also, we want to distribute this equally to L particles where each particle is a 3D multivariate Gaussian. The volume corresponding to the 0.95 confidence level of a multivariate Gaussian is given by

$$V_s = \frac{2\pi^{\frac{p}{2}}}{p\Gamma\left(\frac{p}{2}\right)}(\chi_{p,\alpha}^2)^{\frac{p}{2}}|\mathbf{\Sigma}_{j\to i}|^{\frac{1}{2}}, \quad (47)$$

where $\chi_{p,\alpha}^2$ is taken from the χ-square distribution table and in this case as $p=3, \alpha=0.95$, we have $\chi_{3,0.95}^2 = 5.991$.

Also we assume that the Gaussians are spherical and the covariance matrix for the 0.95 ellipsoid is given by $\mathbf{\Sigma}_{j\to i,0.95} = \sqrt{\lambda\chi^2_{3,0.95}} \times \mathbf{I}$, where λ are the eigenvalues of the covariance matrix. Equating $V_c = LV_s$ and solving for λ, after some algebra, we finally get

$$\mathbf{\Sigma}_{j\to i} = \frac{1}{5.991}\left[\frac{4\sigma_\nu\left(3(r_{ji}^{(t)})^2 + 4\sigma_\nu^2\right)}{L}\right]^{\frac{2}{3}}\mathbf{I}. \qquad (48)$$

Calculation of (48) is faster than having to calculate the weight covariance matrix as is required, e.g., in the rule of thumb technique [10], and simulations show that using this approximation consistenly achieves better results.

REFERENCES

[1] R. Zekavat and R. M. Buehrer, *Handbook of Position Location: Theory, Practice and Advances (IEEE Series on Digital & Mobile Communication)*. Wiley-IEEE Press, 2011.

[2] H. Wymeersch, J. Lien, and M. Win, "Cooperative localization in wireless networks," *Proc. IEEE*, vol. 97, no. 2, pp. 427–450, Feb 2009.

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

[4] M. Spirito, "On the accuracy of cellular mobile station location estimation," *IEEE Trans. Veh. Commun.*, vol. 50, no. 3, pp. 674–685, May 2001.

[5] Y. Shen and M. Win, "Fundamental limits of wideband localization; part i: A general framework," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 4956–4980, Oct 2010.

[6] X. Bao and J. Li, "Cotar: An accurate, cost-effective cooperative wireless localization strategy for mobile nodes," Tech. Rep., 2010.

[7] T. Jia and R. Buehrer, "A set-theoretic approach to collaborative position location for wireless networks," *IEEE Trans. Mobile Comput.*, vol. 10, no. 9, pp. 1264–1275, Sept 2011.

[8] Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 11, pp. 961–974, Nov 2004.

[9] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-calibration in sensor networks," in *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, April 2004, pp. 225–233.

[10] A. T. Ihler, "Inference in sensor networks: Graphical models and particle methods," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 2005. [Online]. Available: http://www.ics.uci.edu/ ihler

[11] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1, pp. 99–141, 2001.

[12] C. Pedersen, T. Pedersen, and B. Fleury, "A variational message passing algorithm for sensor self-localization in wireless networks," in *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, July 2011, pp. 2158–2162.

[13] V. Savic and S. Zazo, "Cooperative localization in mobile networks using nonparametric variants of belief propagation," *Ad hoc networks*, vol. 11, no. 1, pp. 138–150, 2013.

[14] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. Cambridge, MA: MIT Press, 2009.

[15] P.-A. Oikonomou-Filandras, K.-K. Wong, and Y. Zhang, "Informed scheduling by stochastic residual belief propagation in distributed wireless networks," *"IEEE Wireless Commun. Lett.*, vol. 4, no. 1, pp. 90–93, Feb 2015.

[16] H. Wymeersch, F. Penna, and V. Savic, "Uniformly reweighted belief propagation for estimation and detection in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 11, no. 4, pp. 1587–1595, April 2012.

[17] I. Guvenc, C.-C. Chong, and F. Watanabe, "Nlos identification and mitigation for uwb localization systems," in *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, March 2007, pp. 1571–1576.

[18] S. Venkatesh and R. Buehrer, "Non-line-of-sight identification in ultra-wideband systems based on received signal statistics," *Microwaves, Antennas Propagation, IET*, vol. 1, no. 6, pp. 1120–1130, Dec 2007.

[19] ——, "Nlos mitigation using linear programming in ultrawideband location-aware networks," *IEEE Trans. Veh. Commun.*, vol. 56, no. 5, pp. 3182–3198, Sept 2007.

[20] S. Marano, W. Gifford, H. Wymeersch, and M. Win, "Nlos identification and mitigation for localization based on uwb experimental data," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 7, pp. 1026–1035, September 2010.

[21] J. Khodjaev, Y. Park, and A. S. Malik, "Survey of nlos identification and error mitigation problems in uwb-based positioning algorithms for dense environments," *annals of telecommunications-annales des télécommunications*, vol. 65, no. 5-6, pp. 301–311, 2010.

[22] R. Vaghefi and R. Buehrer, "Cooperative sensor localization with nlos mitigation using semidefinite programming," in *9th Workshop on Positioning Navigation and Communication (WPNC), 2012*, March 2012, pp. 13–18.

[23] I. Guvenc and C.-C. Chong, "A survey on toa based wireless localization and nlos mitigation techniques," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 3, pp. 107–124, rd 2009.

[24] F. Yin, C. Fritsche, D. Jin, F. Gustafsson, and A. Zoubir, "Cooperative localization in wsns using gaussian mixture modeling: Distributed ecm algorithms," *IEEE Trans. Signal Process.*, vol. 63, no. 6, pp. 1448–1463, March 2015.

[25] R. Buehrer, T. Jia, and B. Thompson, "Cooperative indoor position location using the parallel projection method," in *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on*, Sept 2010, pp. 1–10.

[26] Y. Qi and H. Kobayashi, "On relation among time delay and signal strength based geolocation methods," in *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, vol. 7, Dec 2003, pp. 4079–4083 vol.7.

[27] D. Barber, *Bayesian reasoning and machine learning*. Cambridge, NY: Cambridge University Press, 2011.

[28] J. Lien, U. J. Ferner, W. Srichavengsup, H. Wymeersch, and M. Z. Win, "A comparison of parametric and sample-based message representation in cooperative localization," *International Journal of Navigation and Observation*, vol. 2012, 2012.

[29] N. Kantas, S. Singh, and A. Doucet, "Distributed self localisation of sensor networks using particle methods," in *Nonlinear Statistical Signal Processing Workshop, 2006 IEEE*, Sept 2006, pp. 164–167.

[30] S. Movaghati and M. Ardakani, "Particle-based message passing algorithm for inference problems in wireless sensor networks," *IEEE Sensors J.*, vol. 11, no. 3, pp. 745–754, March 2011.

[31] A. Ihler, J. Fisher, R. Moses, and A. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 4, pp. 809–819, April 2005.

[32] V. Savic and S. Zazo, "Nonparametric boxed belief propagation for localization in wireless sensor networks," in *Sensor Technologies and Applications, 2009. SENSORCOMM '09. Third International Conference on*, June 2009, pp. 520–525.

[33] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, and R. Tibshirani, *The elements of statistical learning*. Springer, 2009, vol. 2, no. 1.

[34] Y. Qi, H. Kobayashi, and H. Suda, "Analysis of wireless geolocation in a non-line-of-sight environment," *IEEE Trans. Wireless Commun.*, vol. 5, no. 3, pp. 672–681, March 2006.

[35] G. T. Toussaint and J. A. McAlear, "A simple o (n log n) algorithm for finding the maximum distance between two finite planar sets," *Pattern Recognition Letters*, vol. 1, no. 1, pp. 21–24, 1982.

[36] C. M. Bishop and C. M. Bishop, *Pattern recognition and machine learning*. New York, NY: Springer, 2006.

[37] J. Hershey and P. Olsen, "Approximating the kullback leibler divergence between gaussian mixture models," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, April 2007, pp. IV–317–IV–320.

[38] H. P. Langtangen, *Python scripting for computational science*. Berlin, Heidelberg: Springer, 2006, vol. 3.

[39] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, May 2000. [Online]. Available: http://doi.acm.org/10.1145/332833.332838