COOPERATIVE MOBILE NETWORK LOCALIZATION VIA SUBSPACE TRACKING

Hadi Jamali Rad, Alon Amar, and Geert Leus

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology (TU Delft)
e-mail:{h.jamalirad, a.amar, g.j.t.leus}@tudelft.nl

ABSTRACT

Two novel cooperative localization algorithms for mobile wireless networks are proposed. To continuously localize the mobile network, given the pairwise distance measurements between different wireless sensor nodes, we propose to use subspace tracking to track the variations in signal eigenvectors and corresponding eigenvalues of the double-centered distance matrix. We compare the computational complexity of the new algorithms with a recently developed algorithm exploiting the extended Kalman filter (EKF) and show that our proposed algorithms are computationally efficient, and hence, appropriate for practical implementations compared to the EKF. Simulation results further illustrate that the proposed algorithms are more accurate when the distance errors are small (low noise scenarios) in comparison with the EKF, while being more robust to the sampling period in high noise scenarios.

Index Terms— Wireless sensor networks, cooperative mobile localization, multidimensional scaling, subspace tracking.

1. INTRODUCTION

Cooperative localization in mobile networks has recently received a large amount of attention due to its practical importance in wireless sensor networks (WSNs). To accomplish cooperative localization in static networks, pairwise distance measurements based on received signal strength are taken between the nodes of the network while other measurements like angle or time of arrival measurements have also been employed. One popular solution to find the relative locations of the nodes based on distance measurements is to use multidimensional scaling (MDS) or its distributed version for large scale networks [1].

Surprisingly, the problem of cooperative localization for completely mobile sensor networks has not been efficiently solved yet. In [2], an anchorless localization scheme for mobile (dynamic) networks called SPAWN is proposed based on the theory of factor graphs. In this scheme, each node requires knowledge about its own movement model as a probability distribution in order to do predictions, which is not so simple to be acquired in a real application and additionally increases the computational complexity significantly. Moreover, [2] does not provide a computational complexity analysis of the proposed scheme. In [3], an EKF based method is developed to incorporate the nodes' locations as well as their velocities in a state-space model. Although velocity measurements of the nodes aid cooperative network localization, it requires the use of Doppler sensors, which increases the implementation cost, and hence, we avoid using them.

Inspired by the simplicity and robustness of classical MDS localization in static WSNs, we propose MDS based cooperative network localization for a mobile network. However, for a mobile network, computationally intensive eigenvalue decomposition (EVD) calculations should be conducted in each snapshot of the network. To avoid this problem, we propose to use two novel subspace tracking algorithms to track the variations in the signal eigenvectors and corresponding eigenvalues due to variations in the double-centered distance matrix. We show that this can enable us to estimate the next location of the moving nodes in the network given their previous location estimates, if the variations are small. The main advantages of the proposed algorithms can be described as follows. First, the proposed algorithms are computationally efficient, and hence, are suitable choices for practical implementations. Besides, they have a good positioning accuracy in low noise conditions. Finally, they do not rely on the movement model of the nodes (i.e., they are non-parametric) and can be applied to many practical scenarios.

The remainder of this paper is organized as follows. In Section 2, we present the system model underlying our analysis and evaluations. Section 3 describes the proposed cooperative localization algorithms based on subspace tracking. Section 4 compares the computational complexity of the algorithms under consideration in this paper. Section 5 provides simulation results for mobile sensor networks with different network parameters. Concluding remarks are presented in Section 6.

2. SYSTEM MODEL

We consider a network of N mobile wireless sensor nodes, living in a D-dimensional space ($D < N$). Let $\{\boldsymbol{x}_{i,k}\}_{i=1}^{N}$ be the actual vector coordinates of the sensor nodes, or equivalently, let $\boldsymbol{X}_k = [\boldsymbol{x}_{1,k}, \ldots, \boldsymbol{x}_{N,k}]$ be the matrix of coordinates at snapshot k. By collecting the squared pairwise distance measurements $d_{i,j,k}^2$ between the nodes in a distance matrix $\boldsymbol{D}_k$, i.e. $[\boldsymbol{D}_k]_{i,j} = d_{i,j,k}^2$, the double-centered distance matrix can be calculated as $\boldsymbol{B}_k = -1/2\boldsymbol{\Upsilon}\boldsymbol{D}_k\boldsymbol{\Upsilon}$, where $\boldsymbol{\Upsilon}$ is the centering operator [1]. In case of a network with fixed nodes, $\boldsymbol{B}_k$ can be used in the classical MDS to recover the locations of the nodes $\boldsymbol{X}_k$ (up to a translation and orthogonal transformation) by means of the EVD as described in [1].

One trivial solution for a mobile scenario is to perform these computationally intensive EVD calculations for every snapshot of the mobile network. Instead, we propose two low-complexity localization algorithms. The proposed algorithms are anchorless in the sense that the relative positions of the mobile nodes can continuously be calculated without requiring information about the anchor nodes. Although, determining the exact location of the nodes (removing the unknown translation and orthogonal transformation) requires a coordinate system consisting of at least $D+1$ anchor nodes with known locations. It is assumed that these anchor nodes are equipped with long distance transmission devices to continuously localize themselves with respect to (w.r.t) a central coordinate system. Since the number of anchors is generally small compared to the total number of nodes in a network, this requirement is not intensive from a computational and power consumption point of view.

This work was supported by NWO-STW under the VICI program (10382).

3. PROPOSED SUBSPACE TRACKING ALGORITHMS

We start by considering the noiseless case. In that case, the double-centered distance matrix $\boldsymbol{B}_k$ will be a symmetric $N \times N$ matrix of rank D. For the k-th snapshot of the mobile network, the trivial approach is to find the locations by means of the EVD as the solution to $\min \|\boldsymbol{B}_k - \tilde{\boldsymbol{X}}^T \tilde{\boldsymbol{X}}\|^2$, where the minimum is taken over all $D \times D$ matrices $\tilde{\boldsymbol{X}}$. The EVD of $\boldsymbol{B}_k$ can be expressed in the following form

$$\boldsymbol{B}_k = \begin{bmatrix}\boldsymbol{U}_{1,k}\, \boldsymbol{U}_{2,k}\end{bmatrix} \begin{bmatrix}\boldsymbol{\Sigma}_{1,k} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix} \begin{bmatrix}\boldsymbol{U}_{1,k}^T \\ \boldsymbol{U}_{2,k}^T\end{bmatrix} = \boldsymbol{U}_{1,k}\boldsymbol{\Sigma}_{1,k}\boldsymbol{U}_{1,k}^T. \quad (1)$$

Now, the location matrix up to a translation and orthogonal transformation can be written as

$$\tilde{\boldsymbol{X}}_k = \boldsymbol{\Sigma}_{1,k}^{\frac{1}{2}} \boldsymbol{U}_{1,k}^T. \quad (2)$$

Although the above procedure can be done for every snapshot of a mobile network, the complexity of computing the EVD in (1) can be quite intensive for large N [4]. The idea behind the proposed subspace tracking algorithms is that in order to calculate the location of the nodes using (2), we only need to update the D signal eigenvectors in $\boldsymbol{U}_{1,k}$ and their corresponding eigenvalues in $\boldsymbol{\Sigma}_{1,k}$. This can be done by more efficient iterative approaches as proposed in the following.

3.1. Perturbation Expansion-Based Subspace Tracking

In this section, we will explain the idea behind the perturbation expansion-based subspace tracking (PEST). If the movement of the nodes satisfies the property that the invariant subspace of the next (perturbed) double-centered distance matrix ($\boldsymbol{B}_k = \boldsymbol{B}_{k-1} + \Delta\boldsymbol{B}_k$) does not contain any vectors that are orthogonal to the invariant subspace of the current double-centered distance matrix ($\boldsymbol{B}_{k-1}$), the two bases respectively spanning the signal and noise subspace of the next double-centered distance matrix follow the expressions [5]

$$\tilde{\boldsymbol{U}}_{1,k}^u = \tilde{\boldsymbol{U}}_{1,k-1} + \tilde{\boldsymbol{U}}_{2,k-1}\boldsymbol{P}_k, \quad (3)$$
$$\tilde{\boldsymbol{U}}_{2,k}^u = -\tilde{\boldsymbol{U}}_{1,k-1}\boldsymbol{P}_k^T + \tilde{\boldsymbol{U}}_{2,k-1}, \quad (4)$$

where $\boldsymbol{P}_k$ is a coefficient matrix, $\tilde{\boldsymbol{U}}_{i,k}$ represents an orthonormal basis spanning the same subspace as the matrix of eigenvectors $\boldsymbol{U}_{i,k}$, and $\tilde{\boldsymbol{U}}_{i,k}^u$ is an unorthonormalized version of $\tilde{\boldsymbol{U}}_{i,k}$. To compute $\boldsymbol{P}_k$ in (3) and (4), we will resort to a first-order approximation. However, since we will continuously use first-order approximations, we can not assume that $\tilde{\boldsymbol{U}}_{1,k-1}$ and $\tilde{\boldsymbol{U}}_{2,k-1}$ in (3) and (4) are exact orthonormal bases spanning respectively the signal and noise subspaces of $\boldsymbol{B}_{k-1}$. And thus, the first-order approximation of $\boldsymbol{P}_k$ in [5] does not hold anymore, and we need to derive a new $\boldsymbol{P}_k$. The value of $\boldsymbol{P}_k$ should satisfy the necessary and sufficient condition for $\tilde{\boldsymbol{U}}_{1,k}^u$ and $\tilde{\boldsymbol{U}}_{2,k}^u$ to be new bases for the new perturbed signal and noise subspaces, and thus we need

$$(\tilde{\boldsymbol{U}}_{2,k}^u)^T \boldsymbol{B}_k \tilde{\boldsymbol{U}}_{1,k}^u = \boldsymbol{0}. \quad (5)$$

We can expand (5) by substituting (3) and (4) as follows

$$(-\tilde{\boldsymbol{U}}_{1,k-1}\boldsymbol{P}_k^T + \tilde{\boldsymbol{U}}_{2,k-1})^T \boldsymbol{B}_k \times (\tilde{\boldsymbol{U}}_{1,k-1} + \tilde{\boldsymbol{U}}_{2,k-1}\boldsymbol{P}_k) = \boldsymbol{0}. \quad (6)$$

After neglecting the second-order terms, we obtain

$$-\boldsymbol{P}_k\tilde{\boldsymbol{U}}_{1,k-1}^T\boldsymbol{B}_{k-1}\tilde{\boldsymbol{U}}_{1,k-1} + \tilde{\boldsymbol{U}}_{2,k-1}^T\Delta\boldsymbol{B}_k\tilde{\boldsymbol{U}}_{1,k-1} + \underbrace{\tilde{\boldsymbol{U}}_{2,k-1}^T\boldsymbol{B}_{k-1}\tilde{\boldsymbol{U}}_{1,k-1}}_{\neq \boldsymbol{0}} + \underbrace{\tilde{\boldsymbol{U}}_{2,k-1}^T\boldsymbol{B}_{k-1}\tilde{\boldsymbol{U}}_{2,k-1}}_{\neq \boldsymbol{0}}\boldsymbol{P}_k = \boldsymbol{0}. \quad (7)$$

Different from the derivations in [5], the third and fourth terms in (7) are not equal to zero and also the value of their elements increases with each iteration due to the fact that we are using first-order approximations in each snapshot. It is notable that (7) is linear in the elements of $\boldsymbol{P}_k$ and can easily be solved w.r.t $\boldsymbol{P}_k$. However, this requires a $DN \times DN$ matrix inverse calculation which is undesirable due to its high complexity. Therefore, we confine our approximation of $\boldsymbol{P}_k$ to the first three terms in (7). By defining

$$\tilde{\boldsymbol{\Sigma}}_{1,k-1} = \tilde{\boldsymbol{U}}_{1,k-1}^T\boldsymbol{B}_{k-1}\tilde{\boldsymbol{U}}_{1,k-1}, \quad (8)$$

this results in

$$\boldsymbol{P}_k = \tilde{\boldsymbol{U}}_{2,k-1}^T\boldsymbol{B}_k\tilde{\boldsymbol{U}}_{1,k-1}(\tilde{\boldsymbol{\Sigma}}_{1,k-1})^{-1}. \quad (9)$$

To avoid updating $\tilde{\boldsymbol{U}}_{2,k}^u$ in (3), we use $\tilde{\boldsymbol{U}}_{1,k-1}\tilde{\boldsymbol{U}}_{1,k-1}^T + \tilde{\boldsymbol{U}}_{2,k-1}\tilde{\boldsymbol{U}}_{2,k-1}^T = \boldsymbol{I}$ ($\boldsymbol{I}$ represents the identity matrix). Together with (9), this allows us to rewrite (3) as

$$\tilde{\boldsymbol{U}}_{1,k}^u = \tilde{\boldsymbol{U}}_{1,k-1} + (\boldsymbol{I} - \tilde{\boldsymbol{U}}_{1,k-1}\tilde{\boldsymbol{U}}_{1,k-1}^T)\boldsymbol{B}_k\tilde{\boldsymbol{U}}_{1,k-1}\tilde{\boldsymbol{\Sigma}}_{1,k-1}^{-1}. \quad (10)$$

Now, to be able to use the above formula in an iterative manner we should normalize it using any possible orthonormalization process like Gram-Schmidt (GS) factorization. We call the orthonormalized result $\tilde{\boldsymbol{U}}_{1,k}$. As described in [5] and as can be seen from the above derivations, $\tilde{\boldsymbol{U}}_{1,k}$ is an approximation of the orthonormal basis which spans the same subspace as its corresponding signal eigenvectors in $\boldsymbol{U}_{1,k}$. However, to be able to calculate the relative locations using (2), we have to find $\boldsymbol{U}_{1,k}$. To this aim, we look for a matrix $\boldsymbol{A}_k$ so that

$$\tilde{\boldsymbol{U}}_{1,k} = \boldsymbol{U}_{1,k}\boldsymbol{A}_k. \quad (11)$$

Note that since $\tilde{\boldsymbol{U}}_{1,k}$ and $\boldsymbol{U}_{1,k}$ are isometries, $\boldsymbol{A}_k$ will be a unitary matrix. To be able to estimate the locations according to (2), we also need to calculate $\boldsymbol{\Sigma}_{1,k}$, which depends on the value of $\boldsymbol{U}_{1,k}$ and $\boldsymbol{A}_k$ as follows

$$\boldsymbol{\Sigma}_{1,k} = \boldsymbol{U}_{1,k}^T\boldsymbol{B}_k\boldsymbol{U}_{1,k}.$$

From (8), and using (11), we finally obtain

$$\begin{aligned}\tilde{\boldsymbol{\Sigma}}_{1,k} &= (\boldsymbol{U}_{1,k}\boldsymbol{A}_k)^T\boldsymbol{B}_k(\boldsymbol{U}_{1,k}\boldsymbol{A}_k), \\ &= \boldsymbol{A}_k^T\boldsymbol{U}_{1,k}^T\boldsymbol{B}_k\boldsymbol{U}_{1,k}\boldsymbol{A}_k, \\ &= \boldsymbol{A}_k^T\boldsymbol{\Sigma}_{1,k}\boldsymbol{A}_k. \end{aligned} \quad (12)$$

From (12), $\boldsymbol{A}_k$ and $\boldsymbol{\Sigma}_{1,k}$ can be calculated by an EVD of $\tilde{\boldsymbol{\Sigma}}_{1,k}$. Note that, our main goal for using perturbation expansion was to avoid computationally intensive EVD calculations, while here we require it again. However, the point is that $\tilde{\boldsymbol{\Sigma}}_{1,k}$ is a $D \times D$ matrix, which is very small in size compared to the $N \times N$ double-centered distance matrix ($\boldsymbol{B}_k$) for large scale sensor networks. The PEST algorithm is summarized in Algorithm 1.

3.2. Power Iteration-Based Subspace Tracking

Power iterations can also be used to efficiently calculate an invariant subspace of a diagonalizable matrix (like $\boldsymbol{B}_k$) [4]. Power iterations are normally used in an iterative manner to reach an acceptable accuracy. Depending on a random initial guess, the number of iterations can be large, which in turn leads to a high computational complexity. Additionally, an inappropriate choice of the initial guess can sometimes lead to instability and divergence problems [4]. To avoid both problems (complexity and divergence) in mobile network localization, we propose to do just one iteration in each snapshot of the mobile network and use the previous estimate of the orthonormal basis as the initial guess for the next estimate. This leads to a scheme that tracks the desired invariant subspace in a similar fashion as PEST, and we call it power iteration-based subspace tracking (PIST). Note that this power iteration-based approach leads to a unique orthonormal basis spanning the desired signal subspace. Thus, the same EVD calculations as in (12) are required to obtain the matrix of eigenvectors. The PIST algorithm is shown in Algorithm 2.

Table 1. Computational Complexity

Algorithm	Mult.	Orthonorm.	SQRT	Matrix inverse	EVD	Total flops
PEST	$4N^2D + 3ND^2 + ND$	$1(N \times D)$	2	$1\ (D \times D)$	$1\ (D \times D)$	$4DN^2 + (5D^2 + D)N + 2D^3 + 6D^2 + 24$
PIST	$2N^2D + 2ND^2 + ND$	$1\ (N \times D)$	2	-	$1\ (D \times D)$	$4DN^2 + (5D^2 + D)N + D^3 + 24$
EKFT	$(D/2)N^5 + (5D^2/2 - D)N^4 + (12D^3 - 5D^2/2 + 2D)N^3 + (4D^2 - D/2 + 2)N^2 - 2N$	-	$DN(N-1)/2$	$2\ (2DN \times 2DN)$	-	$(D/2)N^5 + (5D^2/2 - D)N^4 + (28D^3 - 5D^2/2 + 2D)N^3 + (52D^2 + 11D/2 + 2)N^2 + (-6D - 2)N$

3.3. Extended Kalman Filter Tracking

For the sake of comparison, we also consider cooperative mobile network localization using the EKF proposed in [3]. However, as we do not have velocity measurements in our setup, we simplify the EKF model of [3]. The discrete-time state and measurement equations can be written as

$$\boldsymbol{x}_k = \boldsymbol{\Phi}\boldsymbol{x}_{k-1} + \boldsymbol{w}_k, \tag{13}$$
$$\boldsymbol{d}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k, \tag{14}$$

where $\boldsymbol{x}_k = [\boldsymbol{x}_{1,k}^T, \ldots, \boldsymbol{x}_{N,k}^T, \dot{\boldsymbol{x}}_{1,k}^T, \ldots, \dot{\boldsymbol{x}}_{N,k}^T]$ is the column vector of length $2DN$ containing the nodes' locations and velocities at the k-th snapshot, $\boldsymbol{d}_k = [d_{1,2,k}, d_{1,3,k}, \ldots, d_{(N-1),N,k}]^T$ is the column vector of pairwise measurements of length $N(N-1)/2$ at the k-th snapshot. $\boldsymbol{\Phi} = \boldsymbol{I} + \boldsymbol{F}T_s$, where T_s is the sampling period and $\boldsymbol{F}$ is

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{0}_{DN\times DN} & \boldsymbol{I}_{DN\times DN} \\ \boldsymbol{0}_{DN\times DN} & \boldsymbol{0}_{DN\times DN} \end{bmatrix}.$$

Further, we set $\boldsymbol{w}_k = [\boldsymbol{0}^T, \bar{\boldsymbol{w}}_k^T]^T$, where we assume that $\bar{\boldsymbol{w}}_k$ and $\boldsymbol{v}_k$ are uncorrelated zero-mean white Gaussian noise processes with standard deviations σ_w and σ_v, respectively. To linearize the measurement equations, we take the Jacobian matrix of $\boldsymbol{h}(\boldsymbol{x}_k)$ defined by an $N(N-1)/2 \times 2DN$ matrix $\boldsymbol{H}_k = \nabla \boldsymbol{h}(\boldsymbol{x}_k)$. The EKF tracking (EKFT) algorithm is shown in Algorithm 3. In the algorithm, $\boldsymbol{W}$ and $\boldsymbol{V}$ are respectively the Jacobian matrix of the partial derivatives of the state and measurement functions w.r.t the process and measurement noise. $\boldsymbol{P}$, $\boldsymbol{R}$ and $\boldsymbol{Q}$ are the covariance matrix of the error in the state estimate, the measurement noise, and the process noise, respectively.

Algorithm 1 PEST

1: Start with an initial location guess
2: **for** $k = 1$ to K **do**
3: Calculate $\tilde{\boldsymbol{U}}_{1,k}^u$ using (10)
4: GS orthonormalization $\tilde{\boldsymbol{U}}_{1,k} = GS(\tilde{\boldsymbol{U}}_{1,k}^u)$
5: Calculate $\tilde{\boldsymbol{\Sigma}}_{1,k}$, $\boldsymbol{A}_k$ and $\boldsymbol{\Sigma}_{1,k}$ using (8) and (12)
6: Calculate $\boldsymbol{U}_{1,k}$ using (11)
7: Location estimation using (2)
8: **end for**

Algorithm 2 PIST

1: Start with an initial location guess
2: **for** $k = 1$ to K **do**
3: Calculate $\tilde{\boldsymbol{U}}_{1,k}^u = \boldsymbol{B}_k \tilde{\boldsymbol{U}}_{1,k-1}$
4: GS orthonormalization $\tilde{\boldsymbol{U}}_{1,k} = GS(\tilde{\boldsymbol{U}}_{1,k}^u)$
5: Calculate $\tilde{\boldsymbol{\Sigma}}_{1,k}$, $\boldsymbol{A}_k$ and $\boldsymbol{\Sigma}_{1,k}$ using (8) and (12)
6: Calculate $\boldsymbol{U}_{1,k}$ using (11)
7: Location estimation using (2)
8: **end for**

Algorithm 3 EKFT

1: Start with an initial location guess
2: **for** $k = 1$ to K **do**
3: Next state:
$\hat{\boldsymbol{x}}_k^- = \boldsymbol{\Phi}\hat{\boldsymbol{x}}_{k-1}$
4: Next error covariance:
$\boldsymbol{P}_k^- = \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^T + \boldsymbol{W}\boldsymbol{Q}_{k-1}\boldsymbol{W}$
5: Compute the Kalman gain:
$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{P}_k^-\boldsymbol{H}_k^T + \boldsymbol{V}\boldsymbol{R}_k\boldsymbol{V}^T)^{-1}$
6: Update the state:
$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\big(\boldsymbol{d}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-)\big)$
7: Update the error covariance:
$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^-$
8: **end for**

4. COMPUTATIONAL COMPLEXITY ANALYSIS

We define the computational complexity as the number of operations required to create one estimate of the nodes' locations. For the sake of simplicity, we do not count the number of additions and subtractions as well as the number of multiplications by 1, -1 or powers of 2, due to the negligible complexity in comparison with multiplications. Also, we consider the same complexity for multiplications and divisions, and hence, we present the sum of them as the number of multiplications (Mult.). Also, we only focus on the steps which are different in the three algorithms and relax the complexity calculations for similar steps (e.g., initialization) in all the algorithms. The results are summarized in Table 1.

The last column in the table presents the maximum number of multiplications as the number of flops. To calculate this, we assume that Gauss-Jordan elimination is used to calculate the matrix inverse and $N^3 + 6N^2$ multiplications are required to calculate the inverse of a $N \times N$ matrix. As well, we assume that the Newton method is used to calculate a scalar square root (SQRT) and 12 multiplications are required. Moreover, the GS orthonormalization process (Orthonorm.) is considered which requires $2ND^2$ multiplications for a $N \times D$ matrix. And, for a $D \times D$ matrix EVD computation, we consider a maximum number of D^3 multiplications. As can be seen in the table, both PEST and PIST have a quadratic complexity in N while it is of order 5 in N (using the matrix inversion lemma) for the EKFT. This results in a much higher complexity for the EKFT in comparison with the proposed algorithms. Note that, including the velocity and acceleration in the EKF process further increases the complexity, and makes the implementation impractical.

5. SIMULATION RESULTS

In this section, we compare the performances of the explained algorithms (PEST, PIST, and EKFT) in different mobile network localization scenarios. We consider a network of $N = 20$ mobile sensors, living in a two-dimensional space ($D = 2$). The mobile nodes are considered to be initially deployed in an area of 100m $\times$100m. To obtain a fair comparison, we consider the random walk process

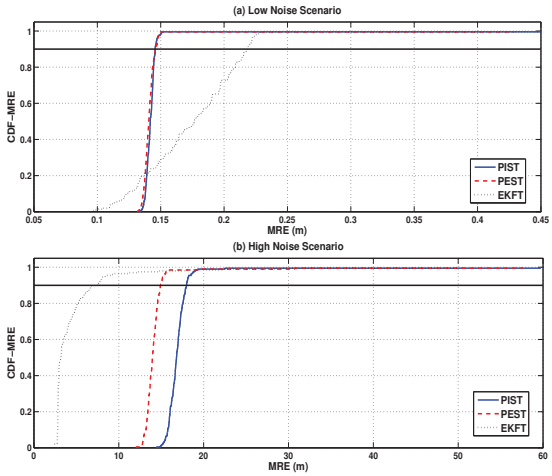

Fig. 1. CDF-MRE performance for $T_s = 0.1$sec, (a) low noise: $\sigma_v = 20$cm, (b) high noise: $\sigma_v = 20$m

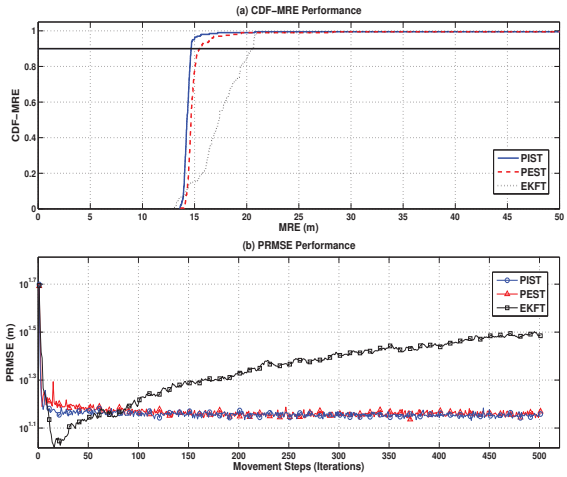

Fig. 2. CDF-MRE (a) and PRMSE (b) performances for $T_s = 10$sec and $\sigma_v = 20$m

(with $\sigma_w = 0.1$) and measurement model as described for the EKFT in Subsection 3.3. Further, we resolve the unknown translation and orthogonal transformation of our obtained location estimates for all three algorithms by considering 3 anchor nodes using Procrustes analysis as explained in [6]. To be able to quantify the performances of the three algorithms, we define the positioning root mean squared error (PRMSE) as $(\frac{1}{M}\sum_{m=1}^{M}(\frac{1}{N}\sum_{i=1}^{N} e_{i,m,k})^2)^{1/2}$, and the mean radial error (MRE) as $\frac{1}{M}\frac{1}{N}\sum_{m=1}^{M}\sum_{i=1}^{N} e_{i,m,k}$, where $e_{i,m,k}$ represents the distance between the real location of the i-th node for the m-th Monte Carlo (MC) trial at the k-th snapshot and its estimated location. Further, M represents the number of independent MC trials, which is $M = 100$ in our simulations.

Figs. 1a and b illustrate the cumulative density function of the MRE (CDF-MRE) for the three algorithms in two different scenarios with the same sampling period $T_s = 0.1$sec. The first scenario (a) is a low noise scenario for which $\sigma_v = 20$cm, while $\sigma_v = 20$m in the second high noise scenario (b). As is clear from Fig. 1a, more than 90% of the estimated locations using PEST and PIST have an MRE of about 15cm, which is about 22cm for the EKFT. This means that under low noise conditions the proposed algorithms clearly outperform the EKFT. From Fig. 1b, we see that more than 90% of the estimated locations using PEST, PIST and EKFT have an MRE of about 15m, 18m and 7m, respectively. This means that under high noise conditions the EKFT outperforms the proposed algorithms and this can be explained by the sensitivity of MDS based localization schemes to noise. However, as can be seen in this case, the PEST outperforms the PIST, which makes it more suitable for high noise scenarios. This is because in noisy scenarios the value of σ_3 (third eigenvalue of $\boldsymbol{B}_k$) will be non-zero and can increase with the amount of noise. Hence, following [4], we conclude that the asymptotic error of the PIST (defined by $(\sigma_3/\sigma_2)^2$) will increase with noise.

Fig. 2 illustrates the same high noise scenario depicted as in Fig. 1b but for a larger sampling period ($T_s = 10$sec). As is clear from the CDF-MRE performance (Fig. 2a), if the sampling period increases (decrease in the computational cost) the performance of the EKFT degrades significantly while PIST and PEST are less sensitive to this parameter. This makes the proposed algorithms more suitable for practical implementations in comparison with the EKFT. A comparison between Fig. 2a and Fig. 1b reveals that the performance of the PEST is also degraded, although much less than the EKFT. This can be justified by the explanation given in Subsection 3.1 which indicates that the perturbations should be small and the fact that the sampling period directly affects the amount of perturbation. The PRMSE performance (Fig. 2b) depicts the convergence process for the three algorithms. Again, this figure shows that the EKFT moves toward divergence with an increase in the sampling period.

6. CONCLUSIONS

Classical MDS is a popular cooperative localization scheme in static wireless networks. However, computing an EVD for each snapshot of a mobile network is computationally intensive. To overcome this problem, we have proposed two novel algorithms based on subspace tracking to track the variations in the signal eigenvectors and corresponding eigenvalues of the double-centered distance matrix. It has been shown that the proposed algorithms have a low computational complexity and outperform a comparable approach using the EKF in terms of localization accuracy in low noise scenarios. Furthermore, it has been illustrated that the proposed algorithms are less sensitive to the sampling period of the measurements, which again leads to cost-efficiency for practical implementations. Future work will be conducted on the distributed realization of the proposed algorithms thereby focusing on networks with partial connectivity.

7. REFERENCES

[1] J. A. Costa, N. Patwari, and A. O. Hero, "Distributed weighted mds for node localization in sensor networks," *ACM Trans. on Sensor Networks*, vol. 2, pp. 39–64, Feb. 2006.

[2] U. Ferner, H. Wymeersch, and M. Z. Win, "Cooperative anchorless localization for large dynamic networks," *Proc. of Intl. Conf. on Ultra-Wideband. ICUWB*, pp. 181–185, Sept. 2008.

[3] L. Dong, "Cooperative network localization via node velocity estimation," *Proc. of IEEE WCNC*, pp. 1–6, April 2009.

[4] G. H. Golub and C. V. Loan, *Matrix computations*. Baltimore, MD: Johns Hopkins Univ. Press, 3rd ed., 1996.

[5] R. J. Vaccaro, "Weighted subspaces fitting using subspace perturbation expansion," *Proc. ICASSP*, pp. 1973–1976, May 1998.

[6] K. W. Cheung and H. C. So, "A multidimensional scaling framework for mobile location using time-of-arrival measurments," *IEEE Trans. on Signal Proc.*, vol. 53, pp. 460–470, Feb. 2005.