Distributed On-line Multidimensional Scaling for self-Localization in Wireless Sensor Networks
Gemma Morral, Ndeye Amy Dieng, Pascal Bianchi

To cite this version:

HAL Id: hal-00960694
https://hal.archives-ouvertes.fr/hal-00960694
Submitted on 18 Mar 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
DISTRIBUTED ON-LINE MULTIDIMENSIONAL SCALING
FOR SELF-LOCALIZATION IN WIRELESS SENSOR NETWORKS

Gemma Morral†, Ndeye Amy Dieng† and Pascal Bianchi†

†Institut Mines-Télécom, Télécom ParisTech, CNRS LTCI
46, rue Barrault, 75634 Paris Cedex 13, France
Email: morralad,pascal.bianchi,ndeye.dieng@telecom-paristech.fr

ABSTRACT
Consider a wireless network formed by fixed or mobile nodes. Each node seeks to estimate its own position based on noisy measurements of the relative distance with other nodes. In a centralized batch mode, positions can be retrieved (up to a rigid transformation) by applying Principal Component Analysis (PCA) on a so-called similarity matrix built from the relative distances. In this paper, we propose a distributed on-line algorithm allowing each node to estimate its own position based on limited exchange of information in the network. Our framework encompasses the case of sporadic measurements and random link failures. We prove the consistency of our algorithm in the case of fixed sensors. Our numerical results also demonstrate the attractive performance of the algorithm for tracking the positions of mobile sensors. Simulations are conducted on a wireless sensor network testbed.

Index Terms— Distributed algorithms, On-line algorithms, Localization, Principal Component Analysis, Wireless Sensor Networks.

1. INTRODUCTION
Self-localization in Wireless Sensor Networks (WSN) have raised a great deal of attention in the signal processing community during the last decades (see [1] or [2] for an overview). In many situations, sensors are unaware of their true positions, but are able to measure their distance with other neighboring sensors. For instance, the inter-sensor distances can be estimated by means of the Received Signal Strength (RSS) intensity, which is directly related to the distance. In such situation, the aim is to build a map of the sensors’ positions on the relative distances. In the literature, the task is often made easier by the assumption that the network contains a significant number of anchor-nodes whose position is perfectly known. In other situations, especially in distributed mobile sensor networks, such anchor nodes may not be present, or too far away. One should therefore rely on anchor-free method.

The problem is often referred to as multidimensional scaling (MDS). The MDS-MAP algorithm is based on the analysis of the principal components (PCA) of a so-called similarity matrix constructed from the relative sensors distances (see [3], [4] for a complete description and analysis). Alternative approaches for MDS on the localization context include: stress function majorization (see [5] Chapter 8), the so-called local-MDS-MAP algorithm [6] which relies on merging local maps and several optimization techniques using semidefinite programming (see [7, 8]), a mass-spring model as in [9] or kernel regression methods as in [10].

Standard algorithms such as those mentioned above are batch algorithms in the sense that the positions are estimated only once – after the prior estimation of relative distances. In this paper, we focus on on-line approaches: each node makes use of a new observation to update the current estimate of its position, and delete its observation after use. This context is specially useful in the case when the sensors need to keep track of their position. In that case, an online version of the MDS-MAP algorithm can be obtained by using the so-called Oja’s algorithm [11, 12] which extracts a sought principal eigenspace.

More recently, the emergence of distributed sensor network raised the question of implementing such algorithms in a distributed fashion [13]. A distributed (batch) algorithm for stress function majorization based on a round-robin communication scheme has been proposed in [14]. On-line distributed gossip-based algorithms have been proposed in [15] and [16]. A distributed version of [9] can be found in [17]. More recently, several attempts have been made to extend Oja’s algorithm to a distributed asynchronous setting [18] [19] but the applicability of such approaches to the problem of self-localization remains unexplored.

In this paper, we propose a distributed online anchor-free method for self-localization. Our algorithm is asynchronous and encompasses the case of random link failures and random noisy and sporadic RSS measurements. The paper is organized as follows. Section 2 introduces the framework and recalls the standard centralized batch MDS approach for localization. Section 3 investigates the case of centralized on-
2. THE FRAMEWORK

2.1. Context and objective

Consider $N$ agents seeking to estimate their respective positions $\{x_1, \ldots, x_N\}$. We assume that $x_i$ are row-vectors in $\mathbb{R}^{1 \times p}$ for any $i$, where $p < N$ (in practice, $p = 2$ or $3$). We assume that agents have access to noisy measurements of relative distances $d_{i,j} = \|x_i - x_j\|^2$ for all $i, j \in \{1, \ldots, N\}$. Based on these noisy measurements, the objective of the network is to estimate the agent’s positions each time new observations are made.

It is worth noting that the above problem is in fact ill-posed. Exact positions are identifiable only up to a rigid transformation. Indeed, quantities $d_{i,j}$ are preserved when an isometry is applied to the agents’ positions. The problem is generally circumvented by assuming that some of the agents (called anchor nodes) are perfectly aware of their position.

2.2. Centralized batch MDS

Although this paper focuses on online localization, it is useful for the sake of completeness to first mention the standard batch MDS approach. Define $S$ as the $N \times N$ matrix of square relative distances $S(i,j) = d_{i,j}^2$. Define $\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$ as the center of mass of the agents. Upon noting that $d_{i,j}^2 = \|x_i - \overline{x}\|^2 + \|x_j - \overline{x}\|^2 - 2 \langle x_i - \overline{x}, x_j - \overline{x} \rangle$, one has:

$$S = c1^T + 1c^T - 2XX^T (1)$$

where $^T$ stands for transpose, $1$ is the $N \times p$ matrix whose components are all equal to one, $c = (\|x_1 - \overline{x}\|^2, \ldots, \|x_N - \overline{x}\|^2)$ and $X$ is a $N \times p$ matrix whose ith line coincides with the row-vector $x_i - \overline{x}$. Otherwise stated, the ith line of $X$ coincides with the barycentric coordinates of agent $i$.

Define $J = 11^T/N$ as the orthogonal projector onto the linear span of $1$. Define $J_\perp = I_N - J$ as the projector onto the space of vectors with zero sum, where $I_N$ is the $N \times N$ identity matrix. It is straightforward to verify that $J_\perp X = X$. Thus, introducing the matrix

$$M \triangleq -\frac{1}{2} J_\perp S J_\perp,$$

equation (1) implies that $M = XX^T$. In particular, $M$ is symmetric non-negative and has rank (at most) $p$. The agents’ coordinates can be recovered from $M$ (up to a rigid transformation) by recovering the principal eigenspace of $M$ i.e., the vector-space spanned by the $p$th principal eigenvectors (see Chapter 12 in [5]). Denote by $\{\sigma_k\}_{k=1}^N$ the eigenvalues of $M$ in decreasing order. In the sequel, we shall always assume that $\sigma_p > 0$, meaning that $M$ has a full column-rank $p$. Denote by $\{u_k\}_{k=1}^p$ the corresponding unit-norm $N \times 1$ eigenvectors. Set $\hat{X} = (\sqrt{\sigma_1} u_1, \ldots, \sqrt{\sigma_p} u_p)$. Clearly $M = \hat{X} \hat{X}^T$ and $X = Q \hat{X}$ for some matrix $Q$ such that $QQ^T = I_N$. Otherwise stated, $\hat{X}$ coincides with the barycentric coordinates $X$ up to an orthogonal transformation. In practice, matrix $S$ is usually not perfectly known and must be replaced by an estimate. This yields Algorithm 1 (Chapter 12 in [5]).

Algorithm 1 Centralized batch MDS for localization

Input: Noisy estimates $\hat{d}_{i,j}$ of $d_{i,j}$ for all $i, j$.
1. Compute matrix $\tilde{S} = (\hat{d}_{i,j}^2)_{i,j=1,\ldots,N}$.
2. Set $M = -\frac{1}{2} J_\perp \tilde{S} J_\perp$.
3. Find the $p$ principal eigenvectors $\{u_k\}_{k=1}^p$ and eigenvalues $\{\sigma_k\}_{k=1}^p$ of $M$.
Output: $\hat{X} = (\sqrt{\sigma_1} u_1, \ldots, \sqrt{\sigma_p} u_p)$

3. CENTRALIZED ONLINE MDS

From now on, we focus on online localization.

3.1. Observation Model

At each time instant $n$, we assume that with probability $q_{i,j}$, an agent $i$ is able to obtain an estimate $S_n(i,j)$ of the square distance with another agent $j \neq i$. The most typical case is when Received Signal Strength (RSS) measurements are made. A traditional model in wireless sensor network [20] assumes that the average attenuation $P_{i,j}$ between nodes $i$ and $j$ is related to the distance $d_{i,j}$ through $P_{i,j} = P_0 - 10 \eta \log_{10} d_{i,j}/d_0$. Parameters $P_0$, $\eta$ and $d_0$ are predetermined constants which depend on the transmission medium. We introduce a collection of independent random variables $(P_n(i,j) : i,j = 1, \ldots , N, n \in \mathbb{N})$ such that $P_n(i,j)$ follows a Gaussian distribution of mean $P_{i,j}$ and variance $\sigma^2$. It is easy to verify from [21] that the quantity

$$D_n(i,j) \triangleq d_{i,j}^2 - \frac{P_n(i,j) - P_0}{\eta} - 10 \frac{\sigma^2 \log_{10} 10}{\log_{10} 50} (2)$$

is an unbiased estimate of $d_{i,j}^2$ i.e., $\mathbb{E}(D_n(i,j)) = d_{i,j}^2$.

We set $D_n(i,i) = 0$. We assume that node $i$ makes an observation and computes the estimate $\hat{D}_n(i,j)$ with probability $q_{i,j}$ and makes no observation otherwise. Thus, one can represent the available observations as the product $B_n(i,j)D_n(i,j)$ where $(B_n)$ is an i.i.d. sequence of random matrices whose components $B_n(i,j)$ follow the Bernoulli distribution of parameter $q_{i,j}$. Stated otherwise, node $i$ observes the $i$th line of matrix $B_n \circ D_n$ at time $n$ where $\circ$ stands for the Hadamard product.

Lemma 1. Assume $q_{i,j} > 0$ for all pairs $i,j$. Set $W := \sum_{i,j} q_{i,j}^N$ and let $B_n$, $S_n$ be defined as above. The matrix

$$S_n \triangleq W \circ B_n \circ D_n \quad (3)$$
3.2. Oja’s online PCA

Our objective is to eventually find a point in the set \( \chi \) of \( N \times p \) matrices whose columns are orthonormal and span the vector space associated with the \( p \) principal eigenvalues of \( M \).

As a consequence of Lemma 1, an unbiased estimate of \( M \) is simply obtained by \( M_n = -\frac{1}{t} J_n S_n J_n \). When faced with random matrices \( M_n \) having a given expectation \( M \), the principal eigenvalues of \( M \) can be recovered by the following algorithm, due to Oja [22] and analyzed in [23]. The algorithm generates a sequence \((U_n)_n\) of \( N \times p \) matrices according to:

\[
U_n = U_{n-1} + \gamma_n \left( M_n U_{n-1} - U_{n-1} U_{n-1}^T U_{n-1} \right),
\]

where \( \gamma_n > 0 \) is a step size.

In order to have more insight, it is convenient to interpret (4) as a Robbins-Monro algorithm (see Chapter 3, [24]) of the form

\[
U_n = U_{n-1} + \gamma_n \left( h(U_{n-1}) + \xi_n \right)
\]

where \( \xi_n \) is a martingale increment noise and \( h \) is the so-called mean field of the algorithm given by \( h(U) = MU - UU^T MU \). It is known that under adequate stability assumptions and vanishing step size \( \gamma_n \), the algorithm converges to the roots of \( h \) (Theorem 2 in [24]). By Theorem 1 of [11] the roots of \( h \) are essentially rotations of matrices whose columns are eigenvectors of \( M \), multiplied by some scalar, including zero. Thus, strictly speaking, the algorithm might converge to a broader set than the sought set \( \chi \). Fortunately, it is known since [25] that all roots of \( h \) outside the set \( \chi \) are unstable. Undesired points can be avoided by standard avoidance of traps methods (see Chapter 4 in [26] and [27]).

In practice, the algorithm (4) is known to suffer from numerical instabilities depending on the initialization [23]. However, since we are expecting convergence to unit-norm vector, these instabilities can be avoided by introducing a projection step:

\[
U_n = \Pi_K \left[ U_{n-1} + \gamma_n \left( M_n U_{n-1} - U_{n-1} U_{n-1}^T U_{n-1} \right) \right],
\]

where \( K \) is any arbitrary compact convex set whose interior contains \( \chi \), and where \( \Pi_K \) is the projector onto \( K \). For completeness, we mention that the authors of [22] proposed an alternative normalization procedure which allows to avoid projection, but which seems difficult to generalize to the distributed context.

3.3. Localization

Let \( u_{n,k} \) denote the \( k \)th column of matrix \( U_n \). If \( (u_{n,k})_n \) converges to one of the eigenvectors of \( M \), then the quantity \( \sigma_{n,k} \) recursively defined by \( \sigma_{n,k} = (1 - \gamma_n) \sigma_{n-1,k} + \gamma_n u_{n-1,k}^T M_n u_{n-1,k} \) converges to the corresponding eigenvalue (see [23]). Finally, an estimate of the barycentric coordinates are obtained by \( \Delta_n = \text{diag}(\sqrt{\sigma_{n,1}}, \ldots, \sqrt{\sigma_{n,p}}) U_n \).

4. DISTRIBUTED ONLINE MDS

4.1. Sparse Asynchronous Communications

It is clear from the previous section that an unbiased estimate of matrix \( M \) is the first step needed to estimate the sought eigenspace. In the centralized setting, this estimate was given by matrix \( M_n = -\frac{1}{t} J_n S_n J_n \). As made clear in Section 3.1, each node \( i \) observes the \( i \)th row of matrix \( S_n \).

As a consequence, node \( i \) has access to the \( i \)th row-average \( \overline{S}_n(i) \equiv \frac{1}{N} \sum_j S_n(i,j) \). This means that matrix \( S_n J_n \) can be obtained with no need to further exchange of information in the network. On the other hand, \( J_n S_n J_n \) requires to compute the per-column averages of matrix \( S_n J_n \). This task is difficult in a distributed setting, as it would require that all nodes share all their observations at any time. A similar obstacle happens in Oja’s algorithm when computing matrix products. To circumvent the above difficulties, we introduce the following sparse asynchronous communication framework.

At time \( n \), we assume that a given node \( \tau_n \) wakes up and transmits a message to other nodes. A given node \( i \neq \tau_n \) receives the latter message with probability \( q \).

**Definition 1** (Asynchronous Transmission Sequence). Let \( q \) be a real number such that \( 0 < q < 1 \). We say that the sequence of random vectors \( T_n = (\tau_n, Z_{n,i} : i \in \{1, \ldots, N\}, n \in \mathbb{N} \) is an Asynchronous Transmission Sequence (ATS) if: i) all variables \( (\tau_n, Z_{n,i})_{i, n} \) are independent, ii) \( \tau_n \) is uniformly distributed on the set \( \{1, \ldots, N\} \), iii) \( \forall i \neq \tau_n, Z_{n,i} \) is a Bernoulli variable with parameter \( q \) i.e., \( \mathbb{P}[Z_{n,i} = 1] = q \) and iv) \( Z_{n,\tau_n} = 0 \).

4.2. The Algorithm

Similarly to (4), our aim is to iteratively generate a sequence of \( N \times p \) matrices \( U_n \). In our distributed framework, each node \( i \) is in charge of the update of the \( i \)th row of \( U_n \) denoted by \( U_n(i) \) (not to be confused with the \( k \)th column previously denoted by \( u_{n,k} \)).

Consider a ATS \((T_n)_n\). Assume that the active node \( \tau_n \) broadcasts its former estimates \( U_{n-1}(\tau_n) \) and \( \overline{S}_n(i) \). All nodes \( i \) such that \( Z_{n,i} = 1 \) receive the estimates. Thus, all nodes compute:

\[
Y_n(i) = \tilde{M}_n(i,i) U_{n-1}(i) + \frac{N}{q} U_{n-1}(\tau_n) \tilde{M}_n(i,\tau_n) Z_{n,i}
\]

(6)

where for any \( i, j \):

\[
\tilde{M}(i,j) = \frac{\overline{S}_n(i) + \overline{S}_n(j) - S_n(i,j) + \delta_n(i)}{2},
\]

\[
\tilde{S}_n(i,j) = \frac{\overline{S}_n(i) + \overline{S}_n(j) + Z_{n,i} Z_{n,j}}{q}.
\]

(7)

(8)

As will be made clear below, the matrix \( Y_n \) whose \( i \)th row coincides with \( Y_n(i) \) can be interpreted as an unbiased estimate of \( MU_{n-1} \) i.e., \( \mathbb{E}(Y_n U_{n-1}) = MU_{n-1} \).
Consider a second ATS \((T'_n)_n\). At time \(n\), node \(i'_n\) wakes up and broadcasts the product \(U_{n-1}(i'_n)Y_n(i'_n)\) to other nodes. Receiving nodes are those \(i\)'s for which \(Z'_{n,i} = 1\). Similarly to (5), we set:

\[
U_n(i) = \Pi_I [U_{n-1}(i) + \gamma_n (Y_n(i) - U_{n-1}(i)\Sigma_n(i))] ,
\]

\[
\Sigma_n(i) = U_{n-1}(i)^T Y_n(i) + \sum_{q} U_{n-1}(q)^T Y_n(q)Z'_{n,q} .
\]

for any \(i\) \(\Pi_I\) is the projector onto the set \(I \triangleq [-r, r]^p\) for an arbitrary \(r > 1\) and \(\Sigma_n(i)\) is a \(p \times p\) matrix.

To obtain the estimate position \(\bar{X}_n(i)\) at each sensor \(i\) as in Section 3.3, the sequence of \(p\) eigenvalues are generated at each \(i\) as the following square matrix \(p \times p\):

\[
\sigma_n(i) = \sigma_n(i-1) + \gamma_n(\text{diag}(\Sigma_n(i)) - \sigma_n(i-1)) ,
\]

then, \(\bar{X}_n(i) = U_n(i) \sqrt{\sigma_n(i)}\) \hspace{1cm} (11)

Algorithm 2 Distributed On-line MDS for localization
At each time \(n = 1, 2, \ldots\)

[Local step]:
- Nodes make sparse measurements of their respective RSS.
- Each node \(i\) evaluates \((S_n(i, j), j = 1, \ldots, N)\) and \(\overline{S}_n(i)\) using (2) and (3).

[Communication step]:
- A node \(i_n\) randomly selected broadcasts \(U_{n-1}(i_n)\) and \(\overline{S}_n(i_n)\) to nodes \(i\) such that \(Z'_{n,i} = 1\).
- Each node \(i\) computes \(Y_n(i)\) by (6).
- A node \(i'_n\) randomly selected broadcasts \(U_{n-1}(i'_n)^T Y_n(i'_n)\) to nodes \(i\) such that \(Z'_{n,i} = 1\).
- Each node \(i\) updates \(U_n(i)\) and \(\bar{X}_n(i)\) by (9) and (10)-(11).

### 4.3. Convergence analysis

Regarding the convergence, we prove that if the sensors’ positions are fixed, the algorithm recovers the latter up to a rigid transformation.

**Assumption 1** (Step size). Sequence \((\gamma_n)_n\) satisfies \(\gamma_n > 0 \rightarrow 0\), \(\sum_n \gamma_n = +\infty\) and \(\sum_n \gamma_n^2 < \infty\).

**Proposition 1.** For any \(U \in \mathbb{R}^{N \times p}\), set \(h(U) = MU - UU^T MU\). Let \(U_n\) be defined by (9). There exists two random sequences \((\xi_n, e_n)_n\) such that, almost surely (a.s.), \(e_n\) converges to zero, \(\sum_n \gamma_n \xi_n\) converges and

\[
U_n = U_{n-1} + \gamma_n(h(U_{n-1}) + \xi_n + e_n) .
\]

\hspace{1cm} (12)

**Theorem 1** (Main result). Let \(U_n\) be defined by (9) and \(\sigma_n,k\) be defined by (11). Under Assumption 1, for any \(k = 1, \ldots, p\), the \(k\)th column \(u_{n,k}\) of \(U_n\) converges to an eigenvector of \(M\) with unit-norm. Moreover, \(\sigma_n,k\) converges to the corresponding eigenvalue.

**Proof.** The proof is an immediate consequence of Proposition 1 along with Theorem 2 of [24]. Sequence \(U_n\) converges a.s. to the roots of \(h\). The latter roots are characterized in [11]. In particular, \(h(U) = 0\) implies that each column of \(U\) is a unit-norm eigenvector of \(M\). The detailed proofs will be given within an extension version of this work. ■

Note that Theorem 1 might seem incomplete in some respect: one indeed expects that the sequence \(U_n\) converges to the set \(\chi\) characterizing the principal eigenspace of \(M\). Instead, Theorem 1 only guarantees that one recovers some eigenspace of \(M\). As discussed in Section 3.2, undesired limit points can be theoretically avoided by introducing an arbitrary small Gaussian noise inside the parenthesis of the lefthand side of (9) (see Chapter 4 in [26]).

### 5. NUMERICAL RESULTS

We consider a set of \(N = 10\) nodes selected from the SensLAB platform [28] (CC2420 sensor nodes which can be managed by uploaded firmwares) which are able to exchange packets and collect corresponding RSS measures. The estimated parameters are : \(\sigma^2 = 16.4, P_0 = -60.3\) and \(\eta = 2.1\). As described in Section 4 we set \(q_{ij} = 0.8\) \(\forall i, j\), \(q = 0.5\) and \(\gamma_t = 0.012\). Figure 1 shows the average estimated relative positions over 100 independent runs after 2000 iterations of Algorithm 2.

![Fig. 1: Denote as : (O) the random initialized positions, (●) the real relative positions and (∙) the average estimated.](image1)

To illustrate the impact of the level noise \(\sigma^2\) and communication parameters \(q_{ij}\) and \(q\), we draw a graph of \(N = 10\) randomly located within the unit square \([0, 1] \times [0, 1]\). Figure 2 displays a comparison about the mean error of both first and second estimated eigenvectors for different sparsity degrees \((q_{ij}, q)\) and considering both noiseless and noisy cases.

![Fig. 2: Root mean square error as a function of the number of iterations \(n\) from the first estimated eigenvector \(u_{n,1}\) (on the right) and the second estimated eigenvector \(u_{n,2}\) (on the left).](image2)
6. REFERENCES


