# A Matrix-Completion Approach to Mobile Network Localization

Qiang Ye[1] Jie Cheng[1] Hongwei Du[2,3] Xiaohua Jia[2] Jing Zhang[2,3]

[1]Dept. of Computer Science and Information Technology
University of Prince Edward Island
Charlottetown, PE, Canada, C1A 4P3
[2]Harbin Institute of Technology Shenzhen Graduate School
Shenzhen, China, 518055
[3]Shenzhen Key Laboratory of Internet Information Collaboration
Shenzhen, China, 518055
{qye, jcheng}@upei.ca, hongwei.du@ieee.org
csjia@hitsz.edu.cn, cshitzhj@gmail.com

## ABSTRACT

Localization in mobile networks is of paramount importance to a variety of pervasive applications. Due to the limitations of GPS, such as high deployment cost, many researchers have devised a variety of different localization schemes based on the measurements of connectivity or distance between neighboring nodes. The existing schemes suffer seriously from either low localization precision or overlong computation time. In this paper, we present a novel localization scheme based on matrix completion, MALL, that utilizes the collected connectivity and distance information to achieve high-precision localization. Since MALL only involves convex optimization and low-complexity non-convex optimization, it can localize mobile nodes at a fast pace. Furthermore, MALL leads to low communication cost. Through intensive simulation and testbed experiments, we found that MALL outperforms the state-of-the-art localization schemes. An in-depth analysis of the time complexity and communication cost of MALL is also included in this paper.

## Categories and Subject Descriptors

C.2.1 [**Computer Communication Network**]: Network Architecture and Design—*Wireless communication*

## General Terms

Algorithm, Performance

## Keywords

Localization; mobile networks; matrix completion; decentralized schemes


*MobiHoc'14,* August 11–14, 2014, Philadelphia, PA, USA.

http://dx.doi.org/10.1145/2632951.2632986.

## 1. INTRODUCTION

Localization in mobile networks is of paramount importance to a variety of pervasive applications, thus having attracted much research effort over the past decades.The Global Position System (GPS) [1] has been widely used to locate individual nodes in mobile networks. However, GPS requires the line of sight to the satellites and therefore stops working when the line of sight is not available (e.g. indoors or in a downtown canyon). In addition, it is often too expensive to equip every mobile node with a GPS. The limitations of GPS have motivated many researchers to devise a variety of different localization schemes based on the measurements of connectivity or distance between neighboring nodes. Most existing localization schemes that utilize the connectivity or distance information are designed for static networks [2][3][4][5]. There are only a limited number of up-to-date localization schemes for mobile networks [6][7][8][9][10][11].

The existing mobile localization schemes can be classified into two categories: centralized and decentralized mechanisms. With centralized mechanisms, a control center collects and processes the connectivity/distance information in order to locate all nodes in the network. Due to the availability of the connectivity/distance information about all mobile nodes, centralized mechanisms tend to result in high-precision localization. However, they suffer seriously from overlong computation and poor scalability because the single control center has to determine the locations of all mobile nodes. TSLRL, TSL, and LRL [10] are three centralized algorithms that need to have the measured distances about the nodes in the network. These schemes also make use of the fact that the coordinate matrix, a matrix consisting of the coordinates of mobile nodes over a period of time, is a low rank matrix that exhibits the temporal stability feature (i.e. the coordinate change of a mobile node tends to be temporally stable).

With decentralized localization mechanisms, each mobile node collects the connectivity/distance information about its neighbors and determines its own location independently. Since only the information about a limited set of nodes is gathered and processed, decentralized algorithms usually lead to fast computation and excellent scalability. However, their localization results tend to be less precise than those

of centralized approaches. Monte Carlo Localization (MCL) [6], MSL* [7], and IMCL [8] are three typical decentralized schemes.

In this paper, we present a novel decentralized localization scheme based on matrix completion for mobile networks, which is formally called "*MA*trix-completion *L*oca*L*ization (MALL)". With MALL, each mobile node uses a 2-step procedure to localize itself precisely at a fast pace in a decentralized manner. The first step uses a matrix completion algorithm inspired by [12]. In [12], Recht *et al.* proposed a matrix completion technique to fill the unknown entries in a matrix if the rank of the matrix is low and the known entries are randomly sampled. In our research, although the matrix to be completed in the first step exhibits the low rank feature, the available entries are not random samples. To complete the matrix successfully, we designed an algorithm based on convex optimization that associates different weights with the available entries and incorporates the temporal stability feature.

The second step of MALL employs an effective matrix completion algorithm to localize mobile nodes. Technically, it takes advantage of the matrix completed in the first step to significantly reduce the computation complexity. In addition, a light-weight constraint that requires the location of each mobile node to be compatible with the locations of its neighoring node is incorporated in the algorithm to improve its precision. As a result, although the algorithm still involves non-convex optimization, it achieves low computation complexity and high precision.

Our experimental results indicate that, MALL outperforms the state-of-the-art decentralized localization schemes in terms of localization error. Compared with the latest centralized schemes, it scales much better and achieves similar localization precision. In comparison to both the existing decentralized and centralized schemes, MALL incurs low communication cost and computation complexity.

The detailed contributions of this paper are summarized as follows. i) We propose a novel localization scheme, MALL, that formulates mobile network localization as a series of matrix completion problems with several carefully-designed constraints. This approach ensures that MALL results in the same high level of precision as centralized localization mechanisms. ii) MALL only involves convex optimization and low-complexity non-convex optimization. As a result, MALL can achieve high-precision localization at a fast pace. iii) MALL only makes use of the information from anchor nodes within two hops and normal nodes within one hop. Consequently, MALL leads to high localization precision at a low communication cost.

The rest of the paper is organized as follows. The localization problem in mobile networks is formulated mathematically in Section 2. Section 3 describes the details of MALL and Section 4 presents our simulation results. The experimental results based on our testbed are summarized in Section 5. Finally, Section 6 concludes the paper.

## 2. PROBLEM FORMULATION

In this section, we formulate the problem of mobile network localization mathematically. Specifically, there are $M$ mobile nodes in the network under investigation. These nodes move in a $d$-dimension Euclidean space. Note that MALL is a generic method that can be used in multidimensional scenarios. In this paper, each of the mobile nodes is assigned a global ID in the range of 1 to $M$. The majority of the nodes are not equipped with built-in GPS and need to be localized using our localization scheme. We call the nodes without GPS "normal nodes" in this paper. The rest of the nodes are called "anchor nodes" because they can use the built-in GPS to determine their precise locations.

In this paper, the concept of neighboring is extended. Specifically, node $i$ is a 1-hop neighbor of node $j$ if it is in $j$'s transmission range. If node $i$ is not $j$'s 1-hop neighbor, but in the transmission range of one of $j$'s 1-hop neighbors, then $i$ is a 2-hop neighbor of node $j$. In this manner, multiple-hop neighbors can be defined. In the rest of the paper, we use $d_{max}$ to denote the transmission range of a mobile node.

With MALL, each normal node collects the location-related information (i.e. connectivity, distances, and coordinates) from its neighborhood to derive its own location. Specifically, each normal node gathers the information from the neighboring anchor nodes within two hops and from the neighboring normal nodes within one hop. The detailed collection method is presented in Section 3. Note that the coordinates of the neighbouring anchors are determined by the built-in GPS while those of the neighboring normal nodes are approximated by MALL. Furthermore, a normal node and its neighbouring nodes (including normal neighbors within one hop and anchor neighbors within two hops) that provide the location-related information form a network, which is formally called the "*local subnet*" of the normal node.

In this paper, time is divided into equal-sized time slots: $t_1$, $t_2$, ... , $t_{final}$. We use $t$ and $t_c$ to refer to any time slot and the current time slot respectively. To achieve high-precision localization, each normal node uses a few matrices (including distance and coordinate matrices) to store the location-related information about its neighbors during the current $T$-slot time window $t_{cw}$. Note that $t_{cw}$ includes the current slot and the past $T$-1 slots. In our research, T is set to 8 for MALL.

With these data matrices, each normal node uses MALL to fill the unknown entries of an object matrix, which contains the coordinates of the nodes in the local subnet of the normal node over the most recent $T_r$-slot time window $t_{rw}$. Note that $t_{rw}$ includes the current slot and the past $T_r$-1 slots. In addition, $T_r$ should be an integer that is much smaller than $T$. In our research, $T_r$ is set to 3 for MALL. Since the coordinates of the normal node at $t_c$ correspond to part of the unknown entries of the object matrix, the normal node can easily localize itself once the object matrix is completed.

Formally, we use the following notations in this paper:

- $G(:, i, t)$ denotes the $d$-dimensional Euclidean coordinates of node $i$ at time slot $t$. Note that the matrix-related notation adopted in this paper is consistent with that used in MATLAB for simplicity purposes.
- $G(k, i, t)$ denotes the $k$-th dimension Euclidean coordinate of node $i$ at time slot $t$. Note that $1 \leq k \leq d$.
- $N_{i,t_c}$ denotes the node set consisting of the nodes in node $i$'s local subnet at the current time slot $t_c$.
- $N_c = |N_{i,t_c}|$ denotes the number of nodes in $N_{i,t_c}$.
- $N_{i,T_r} = \bigcup_{t_c - T_r + 1 \leq t \leq t_c} N_{i,t}$ denotes the node set consisting of all the nodes that have ever appeared in node

$i$'s local subnet over $t_{rw}$. Here $N_{i,t}$ denotes the node set consisting of the nodes in node $i$'s local subnet at the time slot $t$. Note that node $i$'s neighbors might change over time. $N_{i,T_r}$ contains each node that has ever appeared as node $i$'s neighbour over $t_{rw}$ even if the node was node $i$'s neighbour during only one slot within $t_{rw}$.

- $N_r = |N_{i,T_r}|$ denotes the number of neighbor nodes that have appeared in node $i$'s local subnet during $t_{rw}$.
- $N_{i,T} = \bigcup_{t_c-T+1 \le t \le t_c} N_{i,t}$ denotes the node set consisting of all the nodes that have ever appeared in node $i$'s local subnet over $t_{cw}$.
- $N = |N_{i,T}|$ denotes the number of the neighbor nodes that have appeared in node $i$'s local subnet during $t_{cw}$.

In mobile networks, as a node moves around, the set of its neighbors is likely to change over time. That is, the node set consisting of the current and past neighbors tends to grow with time (i.e. $N_{i,t_c} \subseteq N_{i,T_r} \subseteq N_{i,T}$). Consequently, we can arrive at the following conclusion in typical scenarios: $N_c \le N_r \le N$.

In addition to the notations defined previously, the following three matrices are used in this paper:

**Distance Matrix:** For normal node $i$, there are $N_c$ nodes in $i$'s local subnet at $t_c$. Each of them is assigned a local ID that is in the range of 1 to $N_c$. In our research, each normal node $i$ uses an $N_c \times N_c$ matrix $D$ to store the *squared* Euclidean distance between all nodes in $i$'s local subnet at $t_c$.

Formally, an entry of $D$, $D(m,n)$, can be calculated theoretically using Eq. (1):

$$D(m,n) = \|G(:,j,t_c) - G(:,l,t_c)\|^2 \quad (1)$$

where $\|\cdot\|$ denotes the Euclidean norm of a vector and $\|\cdot\|^2$ represents the square of the Euclidean norm; $m$ and $n$ are the local IDs used to identify two nodes in $i$'s local subnet (note that $1 \le m,n \le N_c$); $j$ and $l$ are the corresponding global IDs of $m$ and $n$ respectively.

**Coordinate Matrix:** For each normal node $i$, $N_{i,T}$ consists of $N$ nodes that have been node $i$'s neighbors over $t_{cw}$. In our research, node $i$ uses a 2-dimension coordinate matrix $X$ to store the coordinates of the nodes in $N_{i,T}$.

Globally, we use $G(k,j,t)$ to represent the $k$-th dimension Euclidean coordinate of a node in $N_{i,T}$, where $j$ is the global ID of the node, $t$ is the global slot number, and $1 \le k \le d$. Locally, node $i$ assigns a unique local ID $j^X$, which is in the range of 1 to $N$, to each node in $N_{i,T}$. Note that each local ID $j^X$ corresponds to a unique global ID $j$ for $t_{cw}$. In addition, each slot in the $T$-slot window $t_{cw}$ is given a local slot number $t^X$, which is in the range of 1 to $T$. Note that each $t^X$ corresponds to a unique $t$ for $t_{cw}$. Furthermore, node $i$ uses $X_3(k,j^X,t^X)$ to locally represent the $k$-th dimension Euclidean coordinate of a node in $N_{i,T}$. Obviously, each $X_3(k,j^X,t^X)$ corresponds to a unique $G(k,j,t)$ for $t_{cw}$.

Note that $X_3$ is a 3-dimension array. To take advantage of matrix completion techniques, we can convert $X_3$ into a 2-dimension coordinate matrix by collapsing the first two dimensions of $X_3$ into one dimension. The final coordinate matrix is denoted as $X$=*matricize*$(X_3)$. Formally, $X$ can be obtained using Eq. (2):

$$X(j^X + (k-1)N, t^X) = X_3(k, j^X, t^X) \quad (2)$$

Note that $X$ consists of $N \cdot d$ rows and $T$ columns.

**Object Matrix:** Each normal node $i$ also maintains a 2-dimension object matrix $Y$, which contains the coordinates of the nodes in $N_{i,T_r}$ (note that $N_{i,T_r}$ contains all the nodes that have appeared in $i$'s local subnet over $t_{rw}$). In a manner similar to that used to define $j^X$, $t^X$, and $X_3$, we can define $j^Y$, $t^Y$, and $Y_3$. Note that $1 \le j^Y \le N_r$ and $1 \le t^Y \le T_r$. Then $Y$ is equal to *matricize*$(Y_3)$. Formally, we have:

$$Y(j^Y + (k-1)N_r, t^Y) = Y_3(k, j^Y, t^Y) \quad (3)$$

Note that $Y$ has $N_r \cdot d$ rows and $T_r$ columns, where $N_r = |N_{i,T_r}|$.

As mentioned previously, $N_{i,T_r} \subseteq N_{i,T}$. Hence, the object matrix $Y$ is a submatrix of the coordinate matrix $X$. In this paper, we use $f(\cdot)$ to denote the operation of constructing a matrix by choosing the rows corresponding to the nodes in $N_{i,T_r}$ from another matrix. Consequently, $Y$ and $X$ can be related using Eq. (4):

$$Y = f(X(:, T - T_r + 1; T)) \quad (4)$$

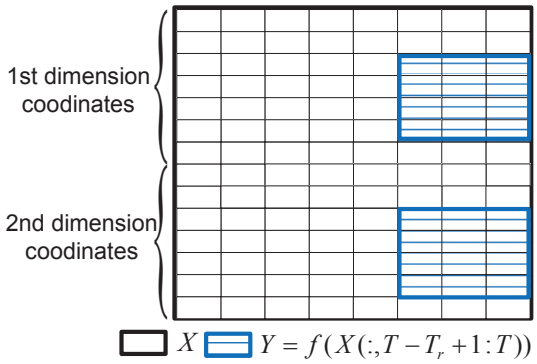



**Figure 1: Relationship between X and Y**

Fig. 1 illustrates the relationship between $X$ and $Y$ using the object and coordinate matrix kept by a normal node $i$ in an example 2-dimension mobile network. Since the network is 2-dimensional, there only exist the $1^{st}$ and $2^{nd}$ dimension coordinates. In addition, as mentioned previously, $T$ and $T_r$ are set to 8 and 3 respectively for MALL. As a result, $X$ and $Y$ always contain 8 and 3 columns respectively. Furthermore, we assume that $N$=7 and $N_r$=3 for the example normal node. Consequently, $X$ has $N\times$ 2 dimensions = 7×2 =14 rows and $Y$ includes $N_r\times$2 dimensions = 3×2 = 6 rows. For clarity purposes, the $1^{st}$ dimension coordinates of the nodes in $N_{i,T_r}$ are placed consecutively in Fig. 1. The $2^{nd}$ dimension coordinates are organized in a similar manner. In practice, the consecutive arrangement is not required.

With MALL, each normal node $i$ uses the collected connectivity, distance, and coordinate information to calculate $Y$. Since node $i$ is a node in its local subnet during $t_{rw}$, $i$'s current coordinates are also included in $Y$. Once $Y$ is available, node $i$ can localize itself easily by retrieving its current coordinates from $Y$.

# 3. MALL: A MATRIX COMPLETION APPROACH

MALL is a decentralized localization scheme based on matrix completion. With MALL, every node detects the connectivity and measures the distance between itself and

its 1-hop neighbors at the current time slot $t_c$. Then the connectivity/distance information is spread in its current local subnet. Furthermore, during the current time slot, each normal node $i$ also forwards its previous coordinates to the nodes in its current local subnet (note that node $i$'s current coordinates are not available yet and will be determined by MALL); each anchor node floods its current local subnet with its current coordinates determined via GPS. As a result, each normal node can collect the connectivity/distance/coordinate information during the time window $t_{cw}$. With the collected information, each normal node $i$ determines its coordinates at the current time slot by calculating the unknown entries in the object matrix $Y$ with a 2-step procedure. An overview of the procedure is presented as follows:

- Step 1: Approximate $X(:, 1 : T-1)$ using a novel matrix completion method that only involves convex optimization. As mentioned previously, the matrix-related notation adopted in this paper is consistent with that used in MATLAB for simplicity purposes. Therefore, $X(:, 1 : T-1)$ denotes a submatrix of the coordinate matrix $X$, which consists of the first $T-1$ columns of $X$.
- Step 2: Calculate the unknown entries in the object matrix $Y$ using a carefully-designed matrix completion method that utilizes the approximated $X(:, 1 : T-1)$. The method results in a high-quality estimation of $Y$. Once the estimation is available, the coordinates of the normal node at $t_c$ can be easily retrieved from the estimated $Y$. Although this method involves non-convex optimization, its complexity is very low.

To find the coordinates at the next time slot, each normal node simply executes the 2-step procedure once again. Note that, when the network is initialized, no normal node knows its current location. To get the localization process started, we can assign some random coordinates to normal nodes. After a short period of time, the normal nodes will start to locate themselves precisely.

It is worth noting that, in this paper, the distance matrix $D$ and the coordinate matrix $X$ are used to denote the matrices that store the precise distance and coordinate information (i.e. the theoretical distances and coordinates) respectively. A key component of the 2-step procedure is to use the collected connectivity/distance/coordinate data to approximate $D$ and $X$. Once the approximated matrices are available, each normal node uses them to determine the object $Y$. To differentiate the priority of the collected data, each normal node uses four $N_c \times N_c$ matrices, $D^{md}$, $D^{ot}$, $Q^{md}$, and $Q^{ot}$, to store the connectivity/distance information gathered at $t_c$. Furthermore, every normal node employs four matrices (whose size is the same size as $X$), $X^{nc}$, $X^{ac}$, $P^{nc}$, and $P^{ac}$, to store the coordinate information collected during $t_{cw}$.

To construct $D^{md}$, $D^{ot}$, $Q^{md}$, and $Q^{ot}$, each normal node $i$ needs to form a weighted undirected graph using the nodes in its current local subnet and the connectivity/distance information gathered at $t_c$. In detail, the connectivity information is first used to create a link between each pair of 1-hop neighbors. Secondly, if the distance between a pair of 1-hop neighbors can be measured, then the distance is used as the weight for the corresponding link. If the distance between them is not available, then the transmission range of the mobile nodes is used as the weight. Based on the undirected graph, $D^{md}$ is a matrix that stores the information about the directly measured distance. Specifically, if an entry in $D^{md}$ corresponds to a node pair who are 1-hop neighbors and whose distance can be measured directly, then the entry contains the square of the measured distance. Otherwise, the entry in $D^{md}$ is filled with a placeholder zero. In addition, $D^{ot}$ is a matrix that stores the information about other types of distances. Specifically, if $D^{md}(m,n)$=0, then $D^{ot}(m,n)$ is equal to the length of the shortest path between vertex $m$ and $n$ in the undirected graph. Note that the length of the shortest path is an approximated distance. If $D^{md}(m,n) \neq$0, then $D^{ot}(m,n)$=0. The reason why we use two matrices to store these distances separately is that the directly measured distances are more accurate than the approximated distances. Therefore, the measured distances should be given a higher weight in the optimization problem presented in the following sections. Furthermore, $Q^{md}$ and $Q^{ot}$ are two matrices used to indicate the non-zero entries in $D^{md}$ and $D^{ot}$. In detail, if $D^{md}(m,n)$=0, then $Q^{md}(m,n)$=0. If $D^{md}(m,n) \neq$0, then $Q^{md}(m,n)$=1. $Q^{ot}$ is defined in a similar manner.

With these definitions, we can use the non-zero entries in $D^{md}$ and $D^{ot}$ to approximate $D$. The resulting $D$ is denoted as $\widetilde{D}$. Formally, $\widetilde{D}$ can be generated using Eq. (5):

$$\widetilde{D}(m,n) = \begin{cases} D^{md}(m,n) & \text{if } D^{md}(m,n) \neq 0 \\ D^{ot}(m,n) & \text{otherwise} \end{cases} \quad (5)$$

Finally, we use $\widetilde{D}^{md}$ and $\widetilde{D}^{ot}$ to denote $\widetilde{D}.*Q^{md}$ and $\widetilde{D}.*Q^{ot}$ respectively, where $.*$ denotes the element-wise product.

To construct $X^{nc}$, $X^{ac}$, $P^{nc}$, and $P^{ac}$, each normal node $i$ needs to collect the coordinate information from its local subnet during $t_{cw}$. The size of these matrices is the same as the coordinate matrix $X$. Specifically, $X^{nc}$ stores the received coordinates from the normal nodes in node $i$'s local subnet during $t_{cw}$. Since the neighbors of node $i$ could change from one time slot to another during $t_{cw}$, the coordinates of a node in $N_{i,T}$ might not be available for some time slots within $t_{cw}$. When the coordinates are not available, the corresponding entries are filled with zero. In addition, $X^{ac}$ stores the received coordinates from the anchor nodes in node $i$'s local subnet during $t_{cw}$. Due to a similar reason, $X^{ac}$ could also contain some zeros that correspond to the unavailable coordinates. Furthermore, $P^{nc}$ and $P^{ac}$ are used to indicate the non-zero entries in $X^{nc}$ and $X^{ac}$ respectively. Specifically, if $X^{nc}(m,n)$=0, then $P^{nc}(m,n)$=0. If $X^{nc}(m,n) \neq$0, then $P^{nc}(m,n)$=1. $P^{ac}$ is defined in a similar fashion. Based on these matrices, the details of the 2-step procedure are presented in the following sections.

## 3.1 Approximating X(:,1:T-1)

Since the nodes in mobile networks keep moving around, the nodes in a normal node's local subnet could change over time. Namely, a node that is not part of the local subnet during one time slot could be included in the local subnet during another slot. Therefore, $X$ tends to include many unknown entries. As mentioned previously, $X(:, 1 : T-1)$ is a submatrix of $X$, which consists of the first $T-1$ columns of $X$. Thus $X(:, 1:T-1)$ is likely to be incomplete too. Step 1 of MALL uses $X^{nc}$ and $X^{ac}$ to approximate the unknown entries in $X(:, 1 : T-1)$. Technically, it takes advantage of the the low rank and temporal stability features of $X(:, 1 : T-1)$ and employs a novel matrix completion method

that only involves convex optimization to accomplish the approximation. The approximated $X(:, 1:T-1)$ will be utilized by Step 2 of MALL.

As mentioned previously, $X$ is a low-rank matrix that exhibits the temporal stability features when the whole network is considered the local subnet (i.e. all nodes in the network are included in the local subnet). Thus, for node $i$, $(G(k,i,t+1)-G(k,i,t))-(G(k,i,t)-G(k,i,t-1))$ approaches zero. When the local subnet only includes the anchor nodes within two hops and normal nodes within one hop, $X$ still has the low-rank and temporal stability features because the $X$ in this scenario is a submatrix of the $X$ in the previous case. As a submatrix of $X$, $X(:, 1:T-1)$ also exhibits the low rank and temporal stability features in this scenario.

Considering that the low rank and temporal stability features of $X(:, 1:T-1)$, we devised a novel matrix completion method that only involves convex optimization to approximate $X(:, 1:T-1)$. In detail, we assume that the rank of $X(:, 1:T-1)$ is $r_1$ and $X(:, 1:T-1)$ can be decomposed into two matrices, $L_X$ and $R_X$. We further assume that $L_X$ has $N \cdot d$ rows and $r_1$ columns; $R_X$ has $T$-1 rows and $r_1$ columns. In addition, $X(:, 1:T-1) = L_X R_X^T$. As mentioned previously, $X(:, 1:T-1)$ has the low rank and temporal stability features. In addition, $X(:, 1:T-1)$ should be consistent with $X^{ac}(:, 1:T-1)$ and $X^{nc}(:, 1:T-1)$. Note that $X^{ac}(:, 1:T-1)$ and $X^{nc}(:, 1:T-1)$ include the received coordinates from the anchor and normal nodes respectively. Thus, approximating $X(:, 1:T-1)$ can be converted to the following minimization problem that takes both the low rank and temporal stability features into consideration:

$$\begin{aligned} \text{minimize} \quad & \left\| (L_X R_X^T) .* P^{ac}(:,1:T-1) - X^{ac}(:,1:T-1) \right\|_F^2 \\ & + \alpha \cdot \left\| (L_X R_X^T) .* P^{nc}(:,1:T-1) - X^{nc}(:,1:T-1) \right\|_F^2 \\ & + \beta \cdot \left( \|L_X\|_F^2 + \|R_X\|_F^2 \right) + \left\| (L_X R_X^T) S_X^T \right\|_F^2 \end{aligned} \tag{6}$$

where $.*$ denotes the element-wise product of two matrices; $\|\cdot\|_F$ denotes the Frobenius norm of a matrix; $\alpha$ and $\beta$ are two tunable parameters; $S_X$ is the temporal transformation matrix used to enforce the temporal stability constraint.

Note that the first two terms of Notation (6) guarantee that the resulting approximation is consistent with the location of the normal and anchor nodes respectively. Since the coordinates of the anchors are more reliable than those of the normal nodes, we set $\alpha$ to 0.1. The third term of Notation (6) corresponds to the low rank property of $X(:, 1:T-1)$. We use $\beta = 0.1$ as the weight for the low rank portion in our research. The last term ensures that the approximation satisfies the temporal stability feature. In our research, $S_X = Toeplitz(0, 1, -2, 1)$, which denotes the Toeplitz matrix with central diagonal given by ones, the first upper diagonal given by negative twos, and the second upper diagonal given by ones. In detail, $S_X$ can be defined using Eq. (7):

$$S_X = \begin{bmatrix} 1 & -2 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & -2 & \cdots & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & -2 & 1 \end{bmatrix} \tag{7}$$

With $S_X$, the last term of Notation (6) guarantees that, for node $i$, $(G(k,i,t+1)-G(k,i,t))-(G(k,i,t)-G(k,i,t-1)) = G(k,i,t-1)+G(k,i,t+1)-2 \cdot G(k,i,t)$ approaches zero.

There are many different methods to solve the minimization problem [12][13]. In our research, the alternating least squares procedure is adopted. Furthermore, we use the operation $f(\cdot)$ to define a special matrix $Z$, which is a submatrix of $X(:, 1:T-1)$ and only includes the rows corresponding to the nodes that appear in the local subnet of node $i$ during $t_{rw}$. Formally, $Z$ can be calculated using Eq. (8):

$$Z = f(X(:, 1:T-1)) \tag{8}$$

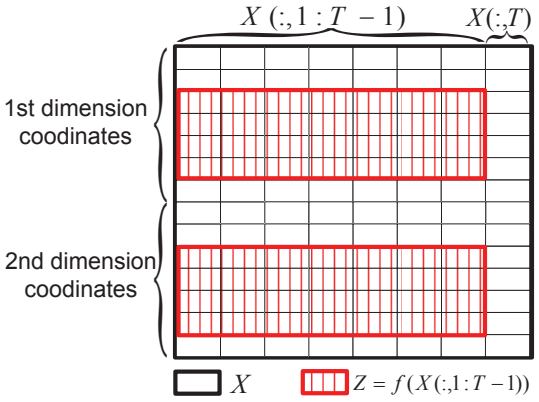



**Figure 2: Relationship between X and Z**

Fig. 2 illustrates the relationship between $X$ and $Z$ using a normal node in the example 2-dimension mobile network mentioned in Section 2. Note that $X(:, 1:T-1)$, a matrix consisting of the leftmost $T-1$ columns of $X$, is also marked in Fig. 2.

Once the approximation of $X(:, 1:T-1)$ is available, we can naturally approximate $Z$ by selecting the corresponding rows in the approximated $X(:, 1:T-1)$. In this paper, we use $\widetilde{Z}$ to denote this approximation of $Z$.

## 3.2 Calculating the Unknown Entries in Y

$Y$ contains the coordinates of the nodes that have appeared in a normal node's local subnet over $t_{rw}$. Most entries in the last column of $Y$ are empty because the coordinates of the normal nodes at $t_c$ are to be determined (i.e. not available yet). Only the entries in the last column corresponding to the anchor nodes are available. Step 2 of MALL uses $\widetilde{D}$, $\widetilde{Z}$, and $X^{ac}$ to fill the unknown entries in $Y$. Technically, a carefully-designed matrix completion method that involves low-complexity optimization is used to estimate $Y$. In this paper, we use $\widetilde{Y}$ to denote the resulting estimated $Y$.

According to the operation $f(\cdot)$ defined previously, $Y$ can be constructed using Eq. (9):

$$Y = f\big(X(:, T - T_r + 1 : T)\big) \tag{9}$$

Using Eq. (8) and (9), we can arrive at the following relationship:

$$f(X) = \big[ Z(:, 1 : T - T_r) \;\; Y \big] \tag{10}$$

where $[\cdot \;\; \cdot]$ denotes the operation of combining two submatrices into one matrix.

Step 2 of MALL attempts to estimate $Y$ by solving a carefully-formulated minimization problem under a series of important constraints. The details of these constraints are presented as follows.

**Connectivity/Distance:** We first formulate the constraints with regard to the connectivity/distance information. To reduce computation cost, we only use the connectivity/distance

information that is collected at $t_c$ and stored in $\widetilde{D}$. Namely, only the entries in $Y(:,T)$ are constrained. The constraints can be divided into two categories: equality constraints and bound constraints. Formally, we use $g(Y)$ to denote the total violation of these constraints:

$$\begin{aligned} g(Y) &= \|D.*Q^{md} - \widetilde{D}^{md}\|_F^2 \\ &+ \sum_{1\le m,n\le N_c} min\big\{0, D(m,n) - D^{lb}(m,n)\big\}^2 \\ &+ \sum_{1\le m,n\le N_c} max\big\{0, (D(m,n) - D^{ub}(m,n)\big\}^2 \end{aligned} \tag{11}$$

where $D$ is the squared distance matrix calculated using $Y(:,T)$ according to Eq. (1). $D^{lb}$ and $D^{ub}$ are two $N_c \times N_c$ matrices containing the lower and upper bound of the distances. For a pair of nodes $m$ and $n$, $D^{lb}(m,n)$ and $D^{ub}(m,n)$ represent the lower bound and upper bound of the distance between them respectively.

Since $\widetilde{D}^{md}$ include the measured distances that are highly precise, we use it to generate equality constraints: $D.*Q^{md} = \widetilde{D}^{md}$. The first term of $g(Y)$ capture the violation of these equality constraints. $D^{lb}$ and $D^{ub}$ are used to generate two bound constraints: $D(m,n) \ge D^{lb}(m,n)$ and $D(m,n) \le D^{ub}(m,n)$. The second and third term of $g(Y)$ represent the violation of these bound constraints. In our research, when two nodes cannot hear each other, $d^2_{max}$ is used as the lower bound. With regard to upper bound, when two nodes, $m$ and $n$, can hear each other, $d^2_{max}$ is used as the upper bound; otherwise, $\widetilde{D}^{ot}(m,n)$ is used as the upper bound.

**Low Rank:** Being a submatrix of $X$, $f(X)$ exhibits the low rank feature. As mentioned previously, $f(X)$ is equal to $[Z(:,1:T-T_r)\ \ Y]$ and Step 1 of MALL results in $\widetilde{Z}$, a satisfactory approximation of $Z$. The low rank constraint used in Step 2 is based on the observation that adding $Y$ to the approximated $\widetilde{Z}(:,1:T-T_r)$ should keep the low rank feature of $f(X)$.

To retain the low rank of $f(X)$, we let:

$$f(X) = \big[Z(:,1:T-T_r)\ \ Y\big] = Z \cdot B_X^T \tag{12}$$

where $B_X$ is an arbitrary $T \times (T-1)$ matrix. If $B_X$ exists, we can arrive at:

$$rank\big(f(X)\big) \le min\big\{rank(Z), rank(B_X)\big\} \tag{13}$$

This is based on the theorem that if a matrix is a product of two other matrices, the rank of the matrix does not exceed the rank of the other two matrices [14].

Note that the first $T-T_r$ columns of $Z$ should be the same as the corresponding part of $f(X)$. Therefore, $B_X(1:T-T_r, 1:T-T_r)$ should be a $(T-T_r)\times(T-T_r)$ identity matrix while each entry in $B_X(1:T-T_r, T-T_r+1:T-1)$ should be equal to zero. Namely, only the bottom $T_r$ rows in $B_X$ is unknown. In this paper, we use a $T_r \times (T-1)$ matrix, $B$, to denote this part of $B_X$. Then we use $h(Y,B)$ to capture the violation of the low rank constraint (i.e. the difference between $Y$ and $\widetilde{Z}\cdot B^T$):

$$h(Y,B) = \big\|Y - \widetilde{Z}B^T\big\|_F^2 \tag{14}$$

Note that $B$ is actually an unknown matrix to be determined in the minimization problem. An approximation of $B$ and $Y$ will be available once the minimization problem is solved. By keeping $h(Y,B)$ small, we can ensure that $[\widetilde{Z}(:,1:T-T_r)\ \ Y]$ is a low rank matrix.

**Temporal Stability:** As a submatrix of $f(X)$, $Y$ has the temporal stability property. Namely, for any node $j$ whose coordinates are included in $Y$, $G(k,j,t-1)+G(k,j,t+1)-2\cdot G(k,j,t)$ approaches zero. To ensure that this property holds well in $Y$, we include the temporal stability constraint in the minimization problem. Formally, the violation of the temporal stability constraint can be captured by $p(Y)$:

$$p(Y) = \big\|Y S_Y^T\big\|_F^2 \tag{15}$$

In our research, we set $T_r = 3$. Therefore, $S_Y = [1\ \ -2\ \ 1]$.

**Matching Received Coordinates:** As mentioned previously, each normal node collects the coordinates of its neighbors in the local subnet over $t_{cw}$. Since these coordinates are either calculated by the neighboring normal nodes or generated by anchors, $Y$ should be consistent with them. In addition, since the coordinates from anchors are more reliable than those from normal nodes, we should assign a higher weight to the violations related to anchors. Formally, we use $q(Y)$ to calculate the total violations of the coordinate matching constraint:

$$\begin{aligned} q(Y) = &\big\|Y.*f\big(P^{ac}(:,T-T_r+1:T)\big) - f\big(X^{ac}(:,T-T_r+1:T)\big)\big\|_F^2 \\ &+\gamma\cdot\big\|Y.*f\big(P^{nc}(:,T-T_r+1:T)\big) - f\big(X^{nc}(:,T-T_r+1:T)\big)\big\|_F^2 \end{aligned} \tag{16}$$

where $\gamma$ is a tunable parameter to adjust the weight of the terms in Eq. (16). In our research, we set $\gamma = 0.1$ to assign a lower weight to the violation term related to normal nodes.

**Complete Formulation:** Formally, filling the unknown entries in $Y$ can be converted into the following minimization problem incorporating the constraints mentioned previously (note that $Y$ and $B$ are the unknown matrices to be determined):

$$\text{minimize}\ \ c(Y,B) = g(Y) + \delta\cdot h(Y,B) + p(Y) + q(Y) \tag{17}$$

where $\delta$ is a tunable parameter ($0 < \delta \le 1$) used to assign a proper weight to the low rank constraint. Since real coordinate matrices are not completely low-rank, the weight of the low rank constraint should be kept small. Generally speaking, $\delta$ could be set to 0.1. In our research, we use a Quasi-Newton optimization algorithm, BFGS [15], to find the optimal solution to the minimization problem, ultimately arriving at $\widetilde{Y}$.

As mentioned previously, once $\widetilde{Y}$ is available, the corresponding node can localize itself easily by retrieving its current coordinates from $\widetilde{Y}$.

## 3.3 Time Complexity

The two steps of MALL correspond to two minimization problems: Notation (6) and Notation (17), by which the time complexity of MALL is dominated. As mentioned previously, Notation (6) is solved using the least squares method while Notation (17) is tackled using BFGS. Hence, the complexity of them are $O(r_1\cdot\overline{N}\cdot T\cdot I_{LS})$ and $O(d\cdot\overline{N_c}^2\cdot I_{BFGS})$, respectively. Here $\overline{N}$ and $\overline{N_c}$ are the average number of neighbors appeared during $t_{cw}$ and the average number of neighbors at $t_c$, respectively. $I_{LS}$ and $I_{BFGS}$ are the iteration number of the least squares method for Notation (6) and the iteration number of BFGS for Notation (17), respec-

tively. Overall, the time complexity of MALL is:

$$O\Big(r_1 \cdot \overline{N} \cdot T \cdot I_{LS} + d \cdot \overline{N_c}^2 \cdot I_{BFGS}\Big) \qquad (18)$$

Despite of the non-convex optimization nature of Notation (17), the complexity of Step 2 is on the same order as that of Step 1, which only involves convex optimization. Consequently, MALL tends to be very competitive compared to the existing mobile localization schemes. Our experimental results are consistent with the theoretical analysis.

# 4. SIMULATION

We carried out intensive simulations to compare MALL with the state-of-the-art mobile localization schemes: MCL [6], MSL* [7], IMCL [8], and TSLRL [10]. MCL, MSL*, IMCL, and MALL are decentralized algorithms while TSLRL is a centralized method. In our simulation, we implemented MCL, MSL*, IMCL and MALL with MATLAB ourselves and adopted the TSLRL program provided by the authors [10]. The performance comparison of these schemes in terms of localization error, running time, and communication cost is presented in this section.

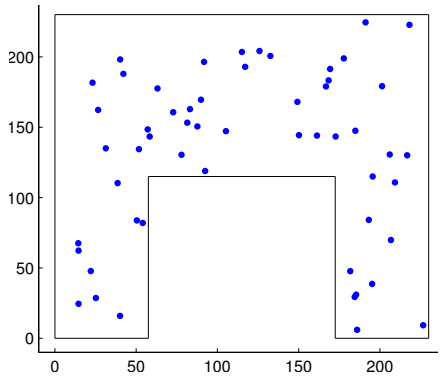



**Figure 3: A $C$-shaped area**

Our simulations were conducted with both synthetic and real mobility traces. The synthetic mobility traces were generated using the standard random waypoint model [16] and modified random waypoint model [6]. Each of the traces involves 50 nodes over 300 time slots, which allows us to complete 10 random runs and calculate the average localization error. In our research, $10m/slot$ and $30m/slot$ are considered the typical low-mobility and high-mobility velocity respectively, which is consistent with the simulations carried out in [7][10]. With synthetic traces, we studied the performance of MALL using two different experimental fields. The first field was a $200m \times 200m$ square-shaped region. The second field is an irregular $C$-shaped area that was used in [10]. Specifically, it is a $230m \times 230m$ square with a $115m \times 115m$ block taken off. The details of the $C$-shaped field is shown in Fig. 3. To ensure every node's position falls within the fields, whenever a node is about to traverse outside, a new randomly-selected speed and direction are re-selected for the node. Unless otherwise specified, 50 nodes, including 5 anchor nodes, are placed in the fields randomly using standard random waypoint model with a maximum speed of $10m/slot$.

As defined in previous studies [7][10], node density is the ratio of the average number of nodes to the area corresponding to $d_{max}$. Formally, we use $\pi d_{max}^2 M/A_{area}$ to denote this parameter, where $M$ is the number of nodes in the network and $A_{area}$ denotes the experimental area. Similarly, anchor density is defined as $\pi d_{max}^2 J/A_{area}$, where $J$ is the number of anchors in the network. The configuration of the experimental fields for synthetic traces leads to a node density of 10 and an anchor density of 1.

To investigate the practical performance of MALL, we also used a few real mobility traces, including ZebraNet traces [17], Seattle Bus traces [18], and Human Mobility traces [19]. Specifically, 100 vehicles in the Seattle Bus traces were randomly selected to the mobile nodes while all the nodes in the ZebraNet and Human Mobility traces were used in our research. In order to have 10 random runs over 300 slots, we extracted 300 consecutive slots from the Human Mobility traces. The Seattle Bus traces only involve 300 slots, thus all of them were used in our simulation. Since the ZebraNet traces only involve 90 slots, we carried out 3 random runs with these traces. For our simulations with the real traces, all the distances in the traces were scaled down in order to arrive at an node density of 10 and anchor density of 1 approximately.

In our simulations, we quantified the localization error using the mean absolute error (MAE) [6] [7][8][10]. Formally, MAE is defined using Eq. (19):

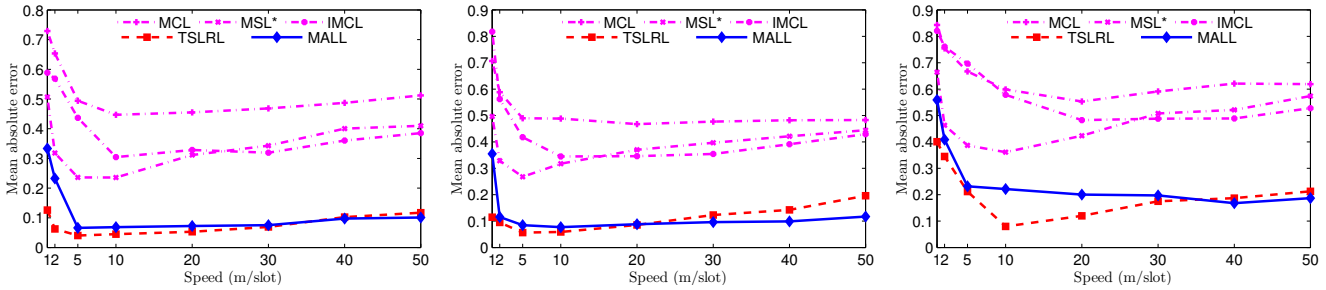
$$MAE = \frac{1}{M \cdot T_{last}} \sum_{i,t} \|G(:,i,t) - \widehat{G}(:,i,t)\| \qquad (19)$$

where $G(:,i,t)$ and $\widehat{G}(:,i,t)$ denote the real and approximated coordinates for node $i$ at $t$ respectively; $T_{last}$ is a time window covering a number of past time slots. MAE indicates the average difference between real and estimated coordinates. In practice, the ratio of MAE to $d_{max}$ is often used to quantify the localization error. In our simulation, we also use this ratio to compare the performance of different localization schemes.

In our experiments, the iteration numbers used by MALL were $I_{LS} = 20$ and $I_{BFGS} = 100$, respectively. These iteration numbers guarantee that MALL can generate satisfactory results. For TSLRL, the time window $t_{cw}$ consists of 15 slots. For MALL, $t_{cw}$ and $t_{rw}$ are set to 8 and 3 slots respectively. In addition, $r_1$ is set to 3. For all the localization schemes under investigation, our simulations keep running for 30 slots in order to ensure convergence. When we compare their performance, we only consider their localization error during the last 10 slots (i.e. $T_{last} = 10$).

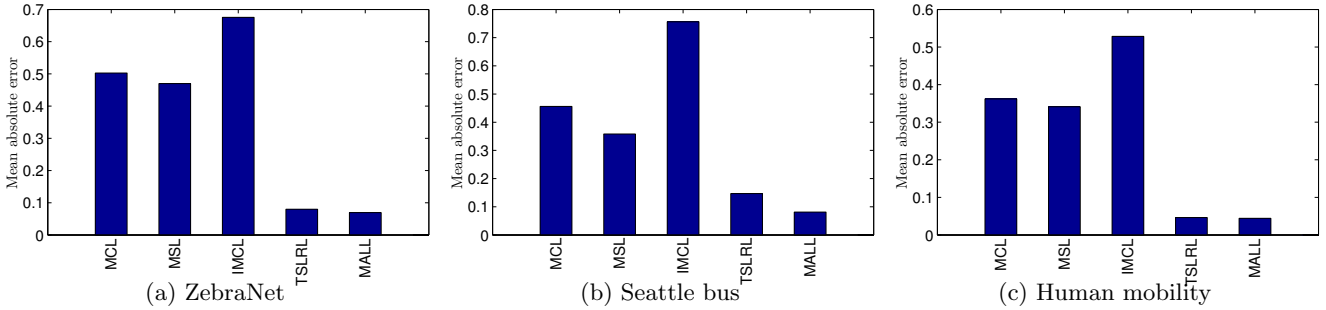
## 4.1 Localization Error

In this section, we compare the localization error of MALL with that of MCL, MSL*, IMCL, and TSLRL using both synthetic and real mobility traces. Fig. 4(a) and Fig. 4(b) illustrate the performance of the localization schemes in the case of the square-shaped region. Specifically, these two figures correspond to the standard and modified random waypoint model respectively. In both of these cases, we calculated the localization error when the maximum speed, $S_{max}$, varied in the range of $1m/slot$ to $50m/slot$. Our experimental results show that the performance of MALL has the following features. First of all, MALL significantly outperforms all of the decentralized schemes, MCL, MSL* and IMCL, in all scenarios. For example, MALL reduces the MAE by a factor of 1.9-6.5 over MCL, 1.4-4.5 over MSL*, and 1.8-5.6 over IMCL. Secondly, the performance of MALL is compa-




(a) Square: standard random waypoint (b) Square: modified random waypoint (c) $C$-shaped: standard random waypoint

**Figure 4: Comparison of different localization schemes using synthetic traces**




(a) ZebraNet (b) Seattle bus (c) Human mobility

**Figure 5: Comparison of different localization schemes using real traces**

rable to that of TSLRL, but a subtle difference does exist. When $S_{max}$ is low, MALL leads to slightly higher localization error. As $S_{max}$ increases, their performance difference narrows till it disappears when $S_{max}$ reaches a critical speed. The critical speed in the case of standard and modified random waypoint model are 30 $m/slot$ and 20 $m/slot$ respectively. Once $S_{max}$ exceeds the critical speed, MALL will start to outperform TSLRL and their performance difference will increase with $S_{max}$. Finally, the MAE of all localization schemes decreases with the maximum speed when the maximum speed is small (i.e. 5 $m/slot$ or less). When the maximum speed exceeds 5 $m/slot$, the MAE starts to increase with the maximum speed.

The localization error performance of MALL in the case of the $C$-shaped region and standard waypoint model was illustrated in Fig. 4(c). In this case, $S_{max}$ was also in the range of $1m/slot$ to $50m/slot$. Our experimental results show that MALL results in a better performance than the decentralized schemes. Compared with TSLRL, MALL leads to a similar localization error level.

We also studied the performance of MALL using real mobility traces. Fig. 5 includes our experimental results. The results indicate that MALL significantly outperforms the distributed schemes, MCL, MSL*, and IMCL. In addition, the performance of MALL is comparable to that of TSLRL, which is a centralized scheme. This is consistent with our experimental results based on synthetic traces.

## 4.2 Running Time

In this section, we compare the running time performance of the localization schemes. Specifically, we vary the size of the networks by increasing the number of nodes from 25 to 400 and check whether the schemes scale well. When the number of nodes increases, the experimental area is enlarged accordingly in order to keep the node density at 10 and anchor density at 1. In our research, we assume that each mobile node attempts to localize itself during every time slot. For MALL, MCL, MSL*, and IMCL we calculate the average running time consumed by each node per time slot. For TSLRL, a centralized approach, we assume that the base station generates the coordinates of all nodes during each slot. Thus we calculate the average running time consumed by the base station per time slot.

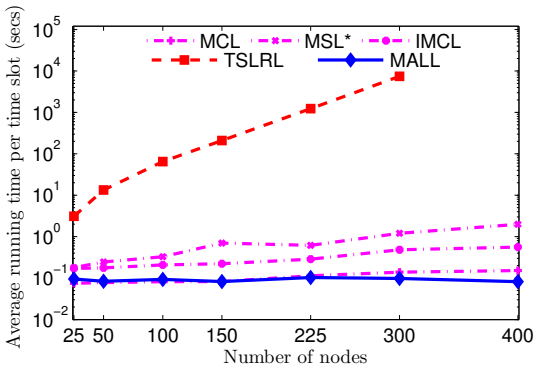



**Figure 6: Running time**

Our simulations were executed on a 64-bit Windows 7 machine with Intel Core i7-3770 CPU, 16GB memory, and 8MB cache. Since TSLRL requires a large volume of memory when the number of nodes is relatively large, we set the virtual memory to 64GB when the TSLRL simulations

were carried out. However, we still cannot run TSLRL when the number of nodes exceeds 300 due to memory overflow. Thus the data point corresponding to the 400-node scenario is missing in the case of TSLRL. However, the data for other schemes are collected successfully.

Fig. 6 includes the details of the running time results. Our results indicate that, in terms of running time, MALL is similar to MCL and outperforms all other schemes. Furthermore, we found that the running time of MALL, MCL, MSL*, and IMCL do not increase very much as the network size goes up while the running time of TSLRL does not scale well. Actually, the running time of TSLRL increases with the network size at a skyrocketing pace.

## 4.3 Communication Cost

In this section, we compare the communication cost of the localization schemes. Technically, we use the average number of packets that are transmitted during a time slot to quantify the communication cost. In our research, we considered the default synthetic scenario. Table 1 includes the the detailed experimental results. With MCL, only anchor nodes forward their location-related information to the neighbors within two hops. As a result, it leads to the lowest communication cost: 68.3 packets/slot. With MSL, all mobile nodes send their data to their neighbors within two hops. Therefore, it results in the highest cost: 746.4 packets/slot. With MALL and IMCL, anchor nodes communicate with their neighbors within two hops while normal nodes only send the information to their 1-hop neighbors. Thus, their communication cost is slightly higher than that of MCL. Considering that TSLRL is a centralized scheme, we assume that a randomly-selected node plays the role of base station and collects all of the location-related information from other nodes. Finally, TSLRL's communication cost is similar to that of MALL and IMCL.

**Table 1: Communication Cost**

| Schemes | MCL | MSL* | IMCL | TSLRL | MALL |
|---|---|---|---|---|---|
| Average number of packets per slot | 68.3 | 746.4 | 113.3 | 121.9 | 113.3 |

# 5. TESTBED EXPERIMENTS

To study the performance of MALL in practice, we tested the schemes under investigation using a sensor testbed in our research lab. Specifically, we deployed 25 TelosB motes on the floor of our lab. Each TelosB mote has a microcontroller with 10kB of RAM and 24KB of flash memory. In addition, it is equipped with a CC2420 RF transceiver that communicates via 2.4GHz radio. The CC2420 chip is tailored for IEEE 802.15.4 and ZigBee protocol stacks. The TelesB mote has the built-in capability to measure the received signal strength (RSS), which enables the RSS-based distance measurement between two nearby motes.

In our experiment, we adjusted the transmission power of the mote to the lowest level: $-25dBm$. However, it still led to an overlong transmission range, which resulted in a fully connected network. To generate a realistic network that is not fully connected, we used a carefully-selected power level threshold, $-73dBm$ in our research. Furthermore, we assumed that two motes cannot communicate with each other if the RSS measurement is less than the threshold. With the threshold of $-73dBm$, the transmission range of the motes is limited to $2.2m$ (i.e. $d_{max}$=2.2m).The experimental region used in our research is a $5m \times 5m$ square. The transmission range and the experimental region lead to a node density of around 15. Among the 25 motes in the testbed, 5 motes are used as anchor nodes.

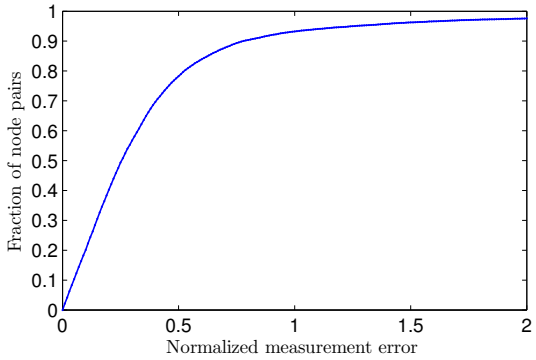



**Figure 7: The CDF of distance measurement error**

In our research, we tested the localization schemes using two different mobility scenarios: low mobility ($0.2D_{max}$/slot) and high mobility ($0.6D_{max}$/slot). For each of these scenarios, the coordinates of motes were generated using the standard random waypoint model [16] over 30 time slots. For each time slot, the motes were placed in the testbed manually according to the generated coordinates. Then each mote broadcast 30 packets so that the neighbors of the mote could acquire multiple RSS measurements. Finally, the average of the RSS readings was used as the RSS value to estimate the distance between each pair of neighboring motes.

We noticed that the RSS-based distance measurement is not very precise. To quantify its performance, we calculated the normalized measurement error, which can be obtained through dividing the difference between the measured (calculated using RSS readings) and actual distance by the actual distance. The cumulative distribution function (CDF) of the normalized measurement error is summarized in Fig. 7. Specifically, for around 80% of the distance measurements, the normalized error is less than 0.5; for around 6% of the distance measurements, the normalized error is greater than 1. Namely, the measured distances in the testbed contain some noise. The testbed can be used to study the practical performance in a relatively realistic environment.

Fig. 8 and 9 illustrates the performance of the schemes under investigation in the testbed. Our experimental results indicate that MALL significantly outperforms the state-of-the-art decentralized schemes, MCL, MSL* and IMCL. In addition, the performance of MALL is similar to TSLRL in both the low and high mobility cases. Overall, the experimental results from our testbed are consistent with those from our simulations.

# 6. CONCLUSIONS

In this paper, we propose MALL, a 2-step algorithm for mobile network localization. The first step uses a novel matrix completion technique to generate an intermediate matrix. The second step makes use of the intermediate matrix and the received location-related information to localize mobile nodes. Through intensive simulation and tested

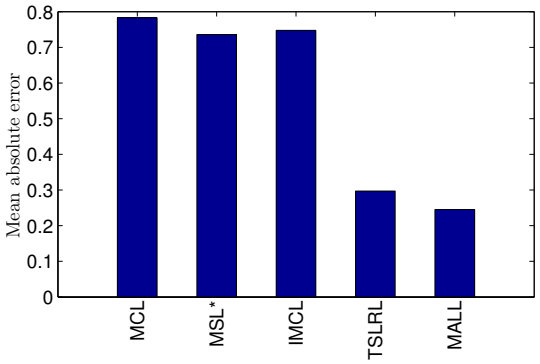



**Figure 8: Comparison of various schemes using the testbed with low mobility**

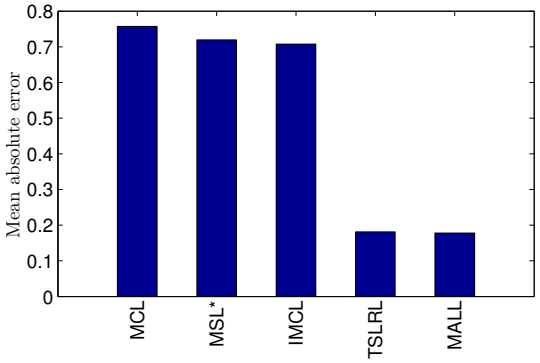



**Figure 9: Comparison of various schemes using the testbed with high mobility**

experiments, we found that MALL outperforms the state-of-the-art decentralized localization schemes in terms of localization error. Compared with the existing centralized schemes, MALL achieves the same high level of precision at a much faster pace because it only uses convex optimization and low-complexity non-convex optimization. In addition, MALL scales much better than centralized schemes due to its decentralized nature. Finally, MALL leads to low communication cost since it only collects the information from the anchor nodes that are at most two hops away and the normal nodes that are one-hop neighbors.

## 7. ACKNOWLEDGMENTS

This research was supported by the National Natural Science Foundation of China under Grant No. 61100191 and 61370216. This research was also supported by Shenzhen Strategic Emerging Industries Program under Grant No. ZDSY20120613125016389.

## 8. REFERENCES

[1] E.D. Kaplan and C.J. Hegarty. *Understanding GPS: principles and applications*. Artech House Publishers, 2006.

[2] Y.T. Chan, W.Y. Tsui, H.C. So, and P. Ching. Time-of-arrival based localization under nlos conditions. *IEEE Transactions on Vehicular Technology*, 55(1):17–24, 2006.

[3] P. Bahl and V.N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Porc. of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 775–784. IEEE, 2000.

[4] Y. Shang, W. Ruml, Y. Zhang, and M.P.J. Fromherz. Localization from mere connectivity. In *Proc. of the 4th ACM international symposium on Mobile ad hoc networking and computing*, pages 201–212. ACM, 2003.

[5] Y. Shang and W. Ruml. Improved mds-based localization. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2640–2651. IEEE, 2004.

[6] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proc. of the 10th annual international conference on Mobile computing and networking*, pages 45–57. ACM, 2004.

[7] M. Rudafshani and S. Datta. Localization in wireless sensor networks. In *Proc. of the 6th international conference on Information processing in sensor networks*, pages 51–60. ACM, 2007.

[8] J. Sheu, W. Hu, and J. Lin. Distributed localization scheme for mobile sensor networks. *IEEE Transactions on Mobile Computing*, 9(4):516–526, April 2010.

[9] A. Baggio and K. Langendoen. Monte carlo localization for mobile wireless sensor networks. *Ad Hoc Networks*, 6(5):718–733, July 2008.

[10] S. Rallapalli, L. Qiu, Y. Zhang, and Y.C. Chen. Exploiting temporal stability and low-rank structure for localization in mobile networks. In *Proc. of the sixteenth annual international conference on Mobile computing and networking*, pages 161–172. ACM, 2010.

[11] H. Chenji and R. Stoleru. Toward accurate mobile sensor network localization in noisy environments. *IEEE Transactions on Mobile Computing*, 12(6):1094–1106, June 2013.

[12] B. Recht, M. Fazel, and P.A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *Society for Industrial and Applied Mathematics Review*, 52(3):471–501, 2010.

[13] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and internet traffic matrices. *ACM SIGCOMM Computer Communication Review*, 39(4):267–278, 2009.

[14] R. Horn. *Matrix Analysis*. Cambridge University Press, February 1990.

[15] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.

[16] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97. ACM, 1998.

[17] http://crawdad.cs.dartmouth.edu/meta.php?name=princeton/zebranet.

[18] http://crawdad.cs.dartmouth.edu/meta.php?name=rice/ad_hoc_city.

[19] http://crawdad.cs.dartmouth.edu/meta.php?name=ncsu/mobilitymodels.