Sensors **2008**, *8*, 1797-1818

sensors

ISSN 1424-8220

www.mdpi.org/sensors

Full Research Paper

Localization Algorithm Based on a Spring Model (LASM) for Large Scale Wireless Sensor Networks

Wanming Chen [1,2], Tao Mei [1,*], Max Q.-H. Meng [1,3], Huawei Liang [1], Yumei Liu [2], Yangming Li [1,2] and Shuai Li [1,2]

[1] Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, 230031, P. R. China; E-mail: cvchen@mail.ustc.edu.cn

[2] Department of Automation, University of Science and Technology of China, Hefei, 230027, P.R. China

[3] Department of Electronic Engineering, The Chinese University of Hong Kong, Sha Tian, Hong Kong; E-mail: max@ee.cuhk.edu.hk

* Author to whom correspondence should be addressed; E-mail: tmei@iim.ac.cn.

Received: 6 December 2007 / Accepted: 12 March 2008 / Published: 15 March 2008

Abstract: A navigation method for a lunar rover based on large scale wireless sensor networks is proposed. To obtain high navigation accuracy and large exploration area, high node localization accuracy and large network scale are required. However, the computational and communication complexity and time consumption are greatly increased with the increase of the network scales. A localization algorithm based on a spring model (LASM) method is proposed to reduce the computational complexity, while maintaining the localization accuracy in large scale sensor networks. The algorithm simulates the dynamics of physical spring system to estimate the positions of nodes. The sensor nodes are set as particles with masses and connected with neighbor nodes by virtual springs. The virtual springs will force the particles move to the original positions, the node positions correspondingly, from the randomly set positions. Therefore, a blind node position can be determined from the LASM algorithm by calculating the related forces with the neighbor nodes. The computational and communication complexity are O(1) for each node, since the number of the neighbor nodes does not increase proportionally with the network scale size. Three patches are proposed to avoid local optimization, kick out bad nodes and deal with node variation. Simulation results show that the computational and communication complexity are almost constant despite of the increase of the network scale size. The time

consumption has also been proven to remain almost constant since the calculation steps are almost unrelated with the network scale size.

Keywords: Localization algorithm, spring model, large scale wireless sensor networks, robot navigation

1. Introduction

An unmanned lunar rover will be part of the essential equipment for moon exploration in the second phase of the Chinese "Chang'e" mission [1,2]. The long-distance localization and navigation of a lunar rover cannot depend on lasers, which can not be used with obstacles. It cannot depend on a Global Positioning System (GPS) either, because there is no GPS on the moon. Other localization and navigation methods, such as inertia navigation, have large error accumulations, which cannot be used in long-distance navigation. Wireless sensor network technologies [3] can be used for the lunar rover's localization and navigation. Wireless sensor nodes are slung around with the movement of lunar rover. With the increase of detection area on the moon surface, a large number of sensor nodes need to be deployed, which constitute a large scale wireless sensor network. Therefore, there is an urgent need to develop a low complexity and high energy efficient localization algorithm for large scale sensor networks on the moon.

Besides this, localization is also important and challenging in wireless sensor networks. A wireless sensor network comprises a mass of geographically separated sensor nodes [4]. Each sensor node consists of a microcontroller, a wireless transceiver battery and many different sensors. Locations of sensor nodes are necessary in a variety of applications, including environment monitoring, object detection, target tracking [5-7], security surveillance [8]. In addition, location information is also necessary for geographic routing protocols [9,10], clustering algorithm [11,12] and geographic data fusing algorithms [13-15].

From the point of view of localization accuracy, localization algorithms can be classified into two different types: range-based algorithms and range-free algorithms [16]. In range-free algorithms, the hop-count based scheme is simple and scalable. In this scheme, the distance between two nodes can be represented by the hop number between them. The classic hop-based algorithms are DV-Hop [17] and Hop-TERRAIN [18] algorithms. The location estimation accuracy is influenced largely by the distribution and density of nodes in the network. E.g., in the network, if the nodes in one region are dense, in another region are sparse, the same hop count in these different regions will represent different distances; if the whole network is sparse, the distance will be represented by a small hop count, which will be resulted in a large quantizing error. DHL [19] uses different hop distances for different nodes densities and uses neighbor numbers of nodes for their densities. In [20], intelligent robots are adopted to improve the node density. Those robots move to the static nodes with few neighbor nodes to enhance their density. The APIT algorithm [16] is another range-free area-based algorithm. The blind node is covered in many triangles formed by a set of anchor nodes. The center of these triangles is defined as the position of the blind node. The localization accuracy is improved by

this method. The COCB-based algorithm [21] is also an area-based distributed scheme. It has lighter computational load and a lower energy consumption comparing with APIT. The CAB algorithm [22] does not require communication between neighbor nodes. The anchor nodes can transmit information at different power levels, dividing the radio range into a circle and rings. The blind node located in these rings uses their center as its position. These range-free algorithms can not determine the exact distance between neighbor nodes; therefore they can not obtain a high accurate localization.

Range-based algorithms utilize relative accurate distance information between neighbor nodes for their localization. The distance information can be obtained (or calculated) by signal time of arrival (ToA), time difference of arrival (TDoA), angle of arrival (AoA), received signal strength indicator (RSSI) and so on [23]. The RSSI scheme is not very accurate due to non-uniform signal propagation. The ToA and TDoA schemes have better accuracy, but require additional hardware such as a highly accurate timer or extra equipment for sending and receiving signals that has a lower propagation speed than radio [24]. When accurate distances between neighbor nodes are obtained, the MDS-based[23] localization algorithm can have accurate position estimation. Multidimensional scaling (MDS) is an efficient data analysis technique originating in psychometrics and psychophysics. MDS-MAP(C) [25] is a centralized algorithm. It requires the distances between all pairs of nodes to form the distance matrix for MDS, and then apply MDS to the distance matrix to calculate the relative positions of these nodes. The relative positions can be converted into global positions if there are absolute positions for more than three or four anchor nodes. The computational complexity is $O(n^3)$. When the number of nodes in the network grows larger, the computational complexity is much larger. MDS-MAP(P) [26] is an improved algorithm for MDS-MAP(C). It is a distributed algorithm. Each node in the network builds a local map based on MDS-MAP(C). Then these maps are patched together to form a global map. The computational complexity is $O(n)$ and the total communication cost is $O(n \log n)$ in this step [26].

In a large scale sensor network containing thousands of sensor nodes, it is very sensitive to the complexity of network, such as computational complexity, time complexity and communication complexity.

In this paper, we propose a low complexity localization algorithm based on spring model which is suitable for large scale networks. It has a complexity of $O(1)$ for each sensor nodes. In the spring model, the sensor nodes are represented by particles with masses. The connection information between each pair of neighbor nodes is represented by a spring connecting this pair of nodes. The distance between this pair of neighbor nodes is represented by the original length of the spring. To localize the positions of blind particles, we imagine that these blind particles are first drawn to random virtual positions by extra forces, and then these particles will go back to their stable positions virtually by the forces of springs. Three patches are also used to avoid local optimization, kick out bad nodes and deal with node variation.

The proposed LASM localization algorithm has the following advantages: First of all, LASM has a low computational, time, and communication complexity of $O(1)$ for each sensor node in distributed scheme, which is suitable for the large scale sensor networks. Second, LASM is optimal under the aspect of minimizing error square sum in the network. This will be proven in Section 2. In addition, LASM is simple to implement. Each sensor node just needs to use some simple addition, subtraction, multiplication and division to calculate its acceleration, velocity, position and so on.

This paper is organized as follows: Both the basic LASM localization algorithm and the patches of LASM are described in Section 2. Following which, the simulation results are shown in Section 3. Conclusions are given in Section 4.

2. Localization Algorithm based on Spring Model (LASM)

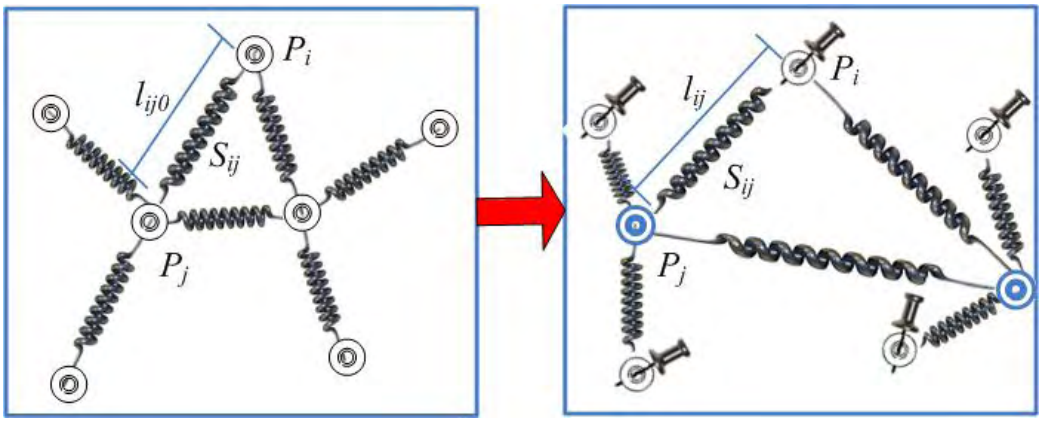
There are some assumptions in this paper:

1) The sensor nodes can be deployed in a two or three-dimensional space. To simplify the explanation, we assume that they are deployed in a two-dimensional space in the rest of the paper.
2) The wireless sensor network is a dynamic network, which means that the sensor nodes can add in or leave from the network at any time. In addition, the sensor nodes can fail at any moment.
3) There are at least 3 anchor nodes that know their accurate positions in the wireless sensor network. Other nodes are blind nodes that do not know their positions. The anchor nodes can be realized by GPS or by manual placing in specific positions.
4) The sensor nodes are able to communicate with neighbor nodes that are in a range of radio range *R*. They can also distinguish their neighbor nodes by their IDs.
5) The distance estimates of neighbor nodes can be obtained. It can be realized by RSSI, ToA, TDoA etc as discussed before.

2.1. System model

A particle-spring system $N = (P, S)$ is proposed to model a wireless sensor network, where P is a particle set and S is a spring set (Figure 1).

Figure 1. Spring model for wireless sensor networks.



Each particle $p_i \in P$ is associated with a sensor node i in the network, and each spring $s_{ij} \in S$ is corresponded to a sensor node pair i, j where the distance d_{ij} between i, j can be obtained. Each particle has four attributes, *i.e.* m, r, v, a, where m denotes the mass of the particle, r denotes the position, v

denotes the velocity and *a* denotes the acceleration of the particle. Each spring has two attributes, *i.e. k*, l_0, where *k* denotes the spring constant, l_0 denotes the original length of the spring.

2.1.1 Static and dynamic equations

For each spring s_{ij}, the force of the spring is defined as:

$$F_{i,j} = -k_{ij}(l_{ij} - l_{ij0}) \tag{1}$$

For each particle p_i, the total force exerted on it is defined as:

$$\vec{F}_i = f(\vec{F}_{i,j}, \vec{v}_i)$$

In this paper, we set $\vec{F}_i$ be:

$$\vec{F}_i = \sum_{s_{ij} \in S} \vec{F}_{i,j} - \eta \vec{v}_i \tag{2}$$

where the η is a constant.

The acceleration of the particle is:

$$\vec{a}_i = \frac{\vec{F}_i}{m_i} \tag{3}$$

The state function for position and velocity is:

$$\begin{aligned} \dot{\vec{r}}_i &= \vec{v}_i \\ \dot{\vec{v}}_i &= \vec{a}_i \end{aligned} \tag{4}$$

The discrete function is:

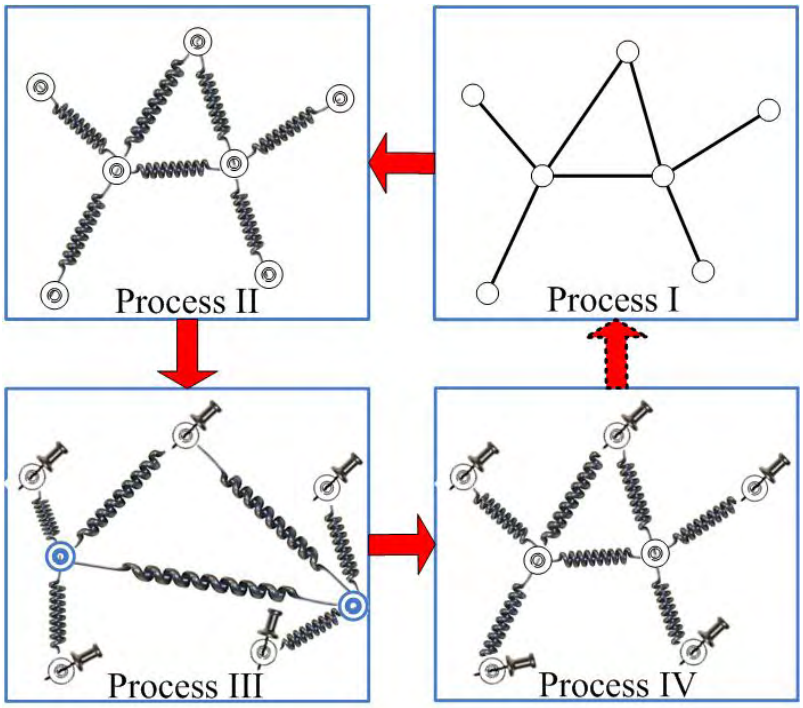
$$\begin{aligned} \vec{r}_i(t + \Delta T) &= \vec{r}_i(t) + \vec{v}_i(t) \cdot \Delta T \\ \vec{v}_i(t + \Delta T) &= \vec{v}_i(t) + \vec{a}_i(t) \cdot \Delta T \end{aligned} \tag{5}$$

2.1.2 Localization process

Figure 2 shows the explanation of the localization process based on the spring model. In process I, sensor nodes are deployed in an area. The circles denote the nodes and the solid lines denote that these two nodes are within the radio range. In process II, the wireless sensor network is represented by the spring system. The sensor nodes are represented by the particles. The distances between neighbor nodes are represented by the springs. As shown in process III, the anchor nodes have their actual absolute positions, so the anchor particles (which represent anchor nodes) are fixed in their absolute positions. The blind particles (which represent blind nodes) don't know their positions, so they are drawn to random positions. After this, the springs between these blind particles are stretched or compressed, which makes the total forces of the blind particles not equal to zero. Therefore, these blind particles will move according to (4). In process IV, these blind particles go back to the stable positions where they are before. The positions can be obtained using (4). These positions are what we need to localized.

Here something needs to be declared: each spring is a virtual one. The position and velocity of each particle is also changed virtually. In fact, the position of each blind node is not change.

Figure 2. Explanation of the proposed localization algorithm based on spring model.
I. wireless sensor network
II. Spring model.
III. Drawing blind particles in random positions
IV. Blind particles go back to their stable positions



2.2. The basic LASM (LASM(B))

The simple process for localization based on the spring model is explained above. We describe the LASM(B) in detail in this part. This localization algorithm can be executed in both distributed and centralized schemes. In a distributed scheme, each sensor node calculates its position through communicating with its neighbors. It's a good method when the sensor network is very large. However, each node needs to communicate with its neighbors, which uses a lot of energy. In a centralized scheme, the sensor nodes send information to the head node. The head node executes the localization algorithm for these nodes and calculates their positions. This method doesn't need rapid communication, but the head node needs a relatively much larger memory and more powerful computing capability.

The distributed scheme for LASM(B) is described as follows:

For each anchor particle, it just sends its actual absolute position to its neighbors if needed. It does nothing else.

For each blind particle *i* in the network:

Step 1: Initialization

Assign its virtual position at (x_i, y_i) randomly and its velocity as zero.

Step 2: Communication

Communicate with its neighbor particles, obtaining the distances between its neighbor particles and itself (the distances can be calculated by RSSI, TDoA, AoA etc.) and the virtual positions of its neighbors.

Step 3: Calculation

Calculate the total force exerted by its neighbors, the acceleration *a*, and then the next virtual position and velocity using (2)(3)(5). It can be noticed that the next step of position is related with ΔT . The larger the ΔT is, the more rapidly the position changes. Therefore, in the beginning, we set the ΔT larger to let the particle run to the right position more rapidly. After that, we set the ΔT smaller to let the particle adjust its position lightly. Here, we set ΔT be:

$$\Delta T = c(1 - l / lstep) \tag{6}$$

where *c* is a constant, *l* is the number of calculation steps it has run (or the number the particle communicates with its neighbors), and *lstep* is the upper limited number of steps we set for each particle.

Step 4: Iteration

Repeat step 2,3 until the total force exerted by neighbors is smaller than a threshold *Tforce* or *l* is larger than *lstep*.

The centralized scheme for LASM(B) is described as follows:

Here, without loss of generality, we assume that the centralized algorithm is executed in the sink node.

Step 1: each sensor node sends the distances between itself and its neighbors to the sink node. The anchor nodes also send their absolute positions to the sink node.

Step 2: the sink node calculates the positions and velocities for all nodes, using the formula similar with the step3 in the distributed scheme described before.

Step 3: repeat step 2 until the total forces of all particles exerted by neighbors are smaller than a threshold *Tforce* or *l* is larger than *lstep*.

In the distributed scheme of LASM(B), each node just communicates with its immediate neighbor nodes and the calculation steps is less than a threshold. Therefore, the computational complexity for each node is *O*(1) and the communication cost is also *O*(1). The total communication cost will be *O*(*n*), where *n* is the number of blind nodes. Each sensor node executes the localization algorithm in parallel model, and has a computational complexity of *O*(1), therefore the time complexity is *O*(1). The computational complexity, time complexity and communication complexity are very appropriate for the large scale sensor networks. In the centralized scheme, the total computational complexity is *O*(*n*), which is also much smaller than that of other localization algorithms such as MDS-MAP ($O(n^3)$).

2.2.1 Convergence analysis

Firstly, the Lyapunov stability of the spring model is analyzed.

A Lyapunov type functional V is defined as:

$$V = \frac{1}{4}\sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot(\vec{r}_i - \vec{r}_j - \vec{r}_{ij}) + \frac{1}{2}\sum_{\{i|p_i\in P\}} m_i\dot{\vec{r}}_i\cdot\dot{\vec{r}}_i \tag{7}$$

where $\vec{r}_{ij}$ is the vector of spring ij's initial length,

$$\vec{r}_{ij} = r_{ij0}\frac{\vec{r}_i - \vec{r}_j}{\left\|\vec{r}_i - \vec{r}_j\right\|}$$

It can be found that $V \geq 0$. $V = 0$ if and only if each spring's length is equal to its initial length and each particle's velocity is equal to 0.

dV/dt is calculated as follows:

$$\frac{dV}{dt} = \frac{1}{2}\sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot(\dot{\vec{r}}_i - \dot{\vec{r}}_j) + \sum_{\{i|p_i\in P\}} m_i\dot{\vec{r}}_i\cdot\ddot{\vec{r}}_i \tag{8}$$

The first item is:

$$\begin{aligned}
&\sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot(\dot{\vec{r}}_i - \dot{\vec{r}}_j)\\
&= \sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_i - \sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_j\\
&= \sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_i - \sum_{\{ij|s_{ij}\in S\}} k_{ji}(\vec{r}_j - \vec{r}_i - \vec{r}_{ji})\cdot\dot{\vec{r}}_i\\
&= \sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_i - \sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_j - \vec{r}_i + \vec{r}_{ij})\cdot\dot{\vec{r}}_i\\
&= 2\sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_i
\end{aligned}$$

The second item is:

$$\begin{aligned}
&\sum_{\{i|p_i\in P\}} m_i\dot{\vec{r}}_i\cdot\ddot{\vec{r}}_i\\
&= \sum_{\{i|p_i\in P\}} [-\sum_{\{j|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij}) - c_i\dot{\vec{r}}_i]\cdot\dot{\vec{r}}_i\\
&= -\sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_i - \sum_{\{i|p_i\in P\}} c_i\dot{\vec{r}}_i\cdot\dot{\vec{r}}_i
\end{aligned}$$

Therefore:

$$\begin{aligned}
\frac{dV}{dt} &= \frac{1}{2}\sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot(\dot{\vec{r}}_i - \dot{\vec{r}}_j) + \sum_{\{i|p_i\in P\}} m_i\dot{\vec{r}}_i\cdot\ddot{\vec{r}}_i\\
&= \sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_i - \sum_{\{ij|s_{ij}\in S\}} k_{ij}(\vec{r}_i - \vec{r}_j - \vec{r}_{ij})\cdot\dot{\vec{r}}_i\\
&\quad - \sum_{\{i|p_i\in P\}} c_i\dot{\vec{r}}_i\cdot\dot{\vec{r}}_i\\
&= -\sum_{\{i|p_i\in P\}} c_i\dot{\vec{r}}_i\cdot\dot{\vec{r}}_i
\end{aligned} \tag{9}$$

It can be found that $dV/dt \leq 0$. $dV/dt \equiv 0$ only when the total force and velocity for each particle are equal to zero.

Therefore, with arbitrary initialization, the system is stable at the points where the total force of each particle is equal to zero.

Because the LASM algorithm simulates the physical process of the system, the LASM algorithm is convergent.

2.2.2 Error analysis

First, in the 2-dimension system, the error square sum of the nodes can be defined as:

$$Err = \sum_{(i,j)} (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_{ij0})^2 \tag{10}$$

where (*i*,*j*) means node *i*, *j* are neighbors.

The minimum of the error square sum is:

$$\min(Err) = \min(\sum_{(i,j)} (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_{ij0})^2)$$

It is minimum when $\frac{\partial Err}{\partial x_i} = 0, \frac{\partial Err}{\partial y_i} = 0 \quad (i = 1, \cdots, n)$

i.e.:

$$\begin{aligned} \frac{\partial Err}{\partial x_i} &= \frac{\partial (\sum_{(i,j)} (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_{ij0})^2)}{\partial x_i} \\ &= 2 \sum_{j \in neighbor(i)} \frac{(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_{ij0})(x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \\ &= 0 \end{aligned} \tag{11}$$

Compared it with the total force in node *i*:

$$\begin{aligned} F_i &= \sum_{s_{ij} \in S} F_{i,j} - \eta v_i \\ &= \sum_{\{j | s_{ij} \in S\}} -k_{ij} (\vec{r}_i - \vec{r}_j - \vec{r}_{ij}) - \eta \vec{v}_i \end{aligned}$$

The component of force in the direction of *x* is:

$$\begin{aligned} F_{xi} &= \sum_{\{j | s_{ij} \in S\}} -k_{ij} (\vec{r}_i - \vec{r}_j - \vec{r}_{ij}) - \eta \vec{v}_i \\ &= \sum_{\{j | s_{ij} \in S\}} -k_{ij} (\frac{\left\| \vec{r}_i - \vec{r}_j \right\| - r_{ij0}}{\left\| \vec{r}_i - \vec{r}_j \right\|} (x_i - x_j)) - \eta \vec{v}_{xi} \end{aligned} \tag{12}$$

When the velocity of node *i* is zero,

$$\begin{aligned} F_{xi} &= \sum_{\{j | s_{ij} \in S\}} -k_{ij} (\frac{\left\| \vec{r}_i - \vec{r}_j \right\| - r_{ij0}}{\left\| \vec{r}_i - \vec{r}_j \right\|} (x_i - x_j)) \\ &= \sum_{\{j | s_{ij} \in S\}} \frac{-k_{ij} (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} - r_{ij0})(x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}} \end{aligned} \tag{13}$$

It can be found that when the spring is stable, i.e. the node is static, the total force equals to zero. Therefore, when $k_{ij} = 2$ and the system is stable, the total force:

$$F_{xi} = -\frac{\partial Err}{\partial x_i} = 0 \tag{14}$$

As described in Section 2.2.1, when the positions of particles are initialized around their true positions, the LASM algorithm will be convergent around their true positions. Using (14), we can conclude that the LASM algorithm minimizes the error square sum when it is convergent at the point

where 1)each force exerted by spring is near or equal to zero and 2)the total force of each particle is equal to zero.

2.3. Patches for the basic localization algorithm (LASM(P))

Three patches for the LASM(B) are lunched to conform to some specific situations. When some particles are not stable at their right positions, i.e. local optimization, patch A is lunched to identify these particles, reinitialize them and reuse the LASM(B) to draw them in their right positions. When there are some nodes which are not well localized (we call them bad nodes), patch B is lunched to seek them out and set different trust values for them. Then we can use the nodes with the higher trust values and delete the bad nodes for the lunar rover's navigation. When there are node variations, i.e., there are sensor nodes adding to or leaving the network, patch C is lunched to deal with these nodes. In this way, the proposed LASM algorithm will be more suitable and robust with different situations.

2.3.1 Patch A for particles at needless point

In some specific situations, some particles may be stable at a needless point. In this point, the total force exerted by its neighbors equals to 0 but at least one force exerted by its neighbor doesn't equal to 0, i.e., the length of one spring r_{ij} doesn't equal to the original length r_{ij0} of the spring. Compared to this point, we denote the point (that each force exerted by its neighbor equals to 0) the right point. A solution is needed to deal with this problem. However, we stress that the number of these needless points is small in the experiment shown in Section 3.

It is not a large problem to find out these particles stabilizing at needless points. For each sensor node, it can obtain the distances between it and its neighbors. It can compare these distances with the lengths of springs to find out whether the sensor node is stable at needless points.

After these sensor nodes (particles) have been found, the next step is to set these nodes out of the needless points.

One simple method is to reinitialize the positions of these particles at random points. After this, we use the LASM(B) algorithm again to run to new positions. The particle is stable at its right point (or the localization algorithm is convergent at its right point as proved in Section 2). Therefore, if the particle runs near the true point, it will be static in the right point finally. Therefore, this simple method can solve this needless point problem.

In addition, another method can solve this problem efficiently.

Because a majority of sensor nodes are stable in their right positions, this information can be used to improve the precision of those particles which are at needless points.

If there are at least 3 neighbor particles (in 2-D system) which are stable at right positions, the least squares algorithm can be used to derive the position of this particle which is at a needless point.

$$\begin{cases} (x_i - x_j)^2 + (y_i - y_j)^2 = {r_{ij0}}^2 \\ (x_i - x_k)^2 + (y_i - y_k)^2 = {r_{ik0}}^2 \\ \vdots \\ (x_i - x_n)^2 + (y_i - y_n)^2 = {r_{in0}}^2 \end{cases} \tag{15}$$

where (x_i, y_i) is the position to be solved, $(x_j, y_j), (x_k, y_k), \cdots, (x_n, y_n)$ is the positions of its n neighbors $(n \geq 3)$, and $r_{ij0}, r_{ik0}, \cdots r_{in0}$ is the distance between the particle and its neighbors.

This function can be linearized by subtracting the last row:

$$\text{AX=b}$$

where $A = -2\begin{bmatrix} x_j - x_n & y_j - y_n \\ x_k - x_n & y_k - y_n \\ \vdots & \vdots \\ x_{n-1} - x_n & y_{n-1} - y_n \end{bmatrix}$, $X = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$, $b = \begin{bmatrix} {r_{ij0}}^2 - {r_{in0}}^2 - {x_j}^2 + {x_n}^2 - {y_j}^2 + {x_n}^2 \\ {r_{ik0}}^2 - {r_{in0}}^2 - {x_k}^2 + {x_n}^2 - {y_k}^2 + {x_n}^2 \\ \vdots \\ {r_{i(n-1)0}}^2 - {r_{in0}}^2 - {x_{n-1}}^2 + {x_n}^2 - {y_{n-1}}^2 + {x_n}^2 \end{bmatrix}$.

This equation can be solved using a traditional least squares algorithm.

$$X = (A'A)^{-1}A'b \tag{16}$$

After the position of this particle is calculated, we consider that this position is also a right position, and also use this position to help its neighbors to calculate their positions using (16).

If there are no particles which can use the second method to reinitialize the positions, the residual particles can use the first simple method to reinitialize.

After these particles which are stable at needless points have been reinitialized, we can reuse the LASM(B) algorithm to obtain the positions of these nodes.

Even after the LASM(B) is reused once, there may also be a smaller number of particles which are not at their right positions. We can reinitialize and reuse the LASM(B) several times to decrease the number of these particles at needless positions. However, it was seen in the experiment that the localization error was very small after the reuse of LASM(B) once. Therefore, LASM(B) is just reused for once in the experiment in Section 3.

The Patch A described above is to reinitialize the positions of those particles at needless points, and then also uses the LASM(B) algorithm. Therefore, the good performance of the LASM(B) algorithm can also be maintained, such as the computational complexity, the communication cost, the convergence and the optimization under the aspect of minimizing error square sum.

2.3.2 Patch B for bad nodes

In order to obtain higher localization accuracy for lunar rover, the bad nodes which can't be well localized need to be found out. A trust value is set for each of these sensor nodes. If the trust value of one node is 1, it means that the estimated position can be well trusted. When using sensor nodes' position information for lunar rover's localization and navigation on the moon, it can just use the information of nodes with higher trusts to the best of our abilities. In this way, the accuracy of lunar rover's localization and navigation can be increased. Here, "not well localized" means that:

1. The node cannot be localized;

2. The node is localized at needless points.

Theorem 1. In a two dimensional system, if a particle is connected with at least 3 unaligned particles who know their positions, this particle can be localized. ('Unaligned' means that these particles are not in a line)

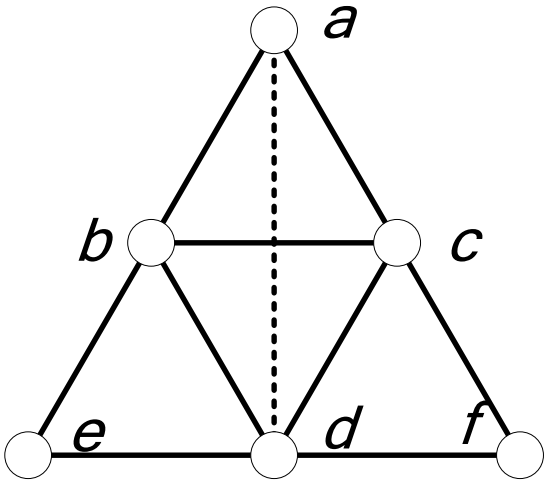
Proof. If two particles are connected, the distance between these two particles can be obtained. When there are three nodes *j*, *k*, *l* , which know their positions and the distances (between these nodes and the un-located node *i*), triangulation algorithm [18] can be used as follows:

$$\begin{cases} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = r_{ij0} \\ \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} = r_{ik0} \\ \sqrt{(x_i - x_l)^2 + (y_i - y_l)^2} = r_{il0} \end{cases} \tag{17}$$

Therefore, the coordinates (x_i, y_i) of the particle *i* can be calculated.

In some specific situations, a particle can also be localized even if there are only two particles connected with it, which is shown in Figure 3. Particle *a* is just connected with particle *b* and *c*. This particle can be localized at two positions (position *a* and *d*). However, if this particle were located at position *d*, it would have obtained connections with particles *e*, *f*. In face, it does not have connections with *e*, *f*. Therefore, this particle can just be localized at position *a*.

Figure 3. An specific example where node a can also be localized.



However, we don't pay much attention on these situations, because we need a fast and simple algorithm to find out these nodes for the lunar rover's localization and navigation on the moon, not a precise algorithm.

Two methods are used to solve this problem. The relatively complex algorithm is used to find out the nodes which can't be localized as follows:

First, 3 particles which are connected with each other are chosen and put in a set. Using these 3 particles, a relative coordinate system can be established.

Secondly, if there are particles which are connected with at least 3 particles in that set, these particles can also be localized. And we also put them into that set. The amount of particles in that set will increase.

In addition, if there are at least 3 anchor nodes in this set, all particles in this set can have absolute coordinates by coordinate transformation.

Finally, the particles which can't be put in the set may not be able to be localized to a great extent (These node may also be localized in some specific situations as described before).

This algorithm is a little complex, and do not take the information of LASM(B) into consideration. Therefore, a new algorithm can be used to find out the nodes which can't be well localized.

The main idea of this algorithm is as follows:

Step 1, for each particle, if it is at needless point, the trust of this particle is set to be 0.5; if it only has one or two neighbor particles, the trust of this particle is set to be 0. The trusts of other particles are set to be 1.

Step 2, for each particle, if its trust is not equal to 1, cut off all the springs connected with it. Therefore, it will not be the neighbor particles of others. And then tells its former neighbors to delete itself from the neighbors' neighbor list.

Step 3, repeat steps 1&2, until no particles need to change their trust.

Even if the trust of the particle is less than 1, it maybe not well localized with certainty. However, if the trust of the particle is 1, it may also not well localized in some specific situations. Here, we just need a fast and simple algorithm; therefore we don't pay more attention to the precision.

All the sensor nodes are divided into 3 parts with different trust values. The sensor nodes with higher trusts will be more likely used in the latter process of lunar rover's localization and navigation.

2.3.3 Patch C for node variation

With the increase of the exploration area and working time on the moon, there are more nodes adding to or leaving from the network. How to deal with these nodes after the network has been localized?

When a new blind node adds in the network, it communicates with its neighbors and then obtains their positions and distances between itself and its neighbors. First, the positions of its neighbors are fixed. Then, the new node calculates its position using the proposed LASM algorithm. After that, small movements are made for the other nodes in the network using LASM. When a new anchor node adds in the network, it communicates its position to its neighbor nodes. These neighbor nodes use the LASM to adjust their positions again.

When a new blind node adds in the network in the LASM process of its neighbor nodes, it first calculates its approximate position using (16). Then it broadcasts its position to its neighbors using LASM to estimate its position. Its neighbor nodes receive its position, and then add it in the calculation of total force.

When a node leaves from the network after the LASM process, if the network is immobile (which means that the nodes in the network don't move once being deployed), the positions of the other nodes need not be changed. Therefore, nothing needs to be done on the positions of the other nodes. If the network is mobile, the other nodes will be relocalized using the proposed LASM algorithm.

When a node leaves from the network suddenly during the LASM process, may its neighbor nodes run into a deadlock because they are waiting for the leaving node's position information? The answer is no. In the LASM process, a time threshold can be set for each sensor node. If it does not receive all the neighbors' position information in the time threshold, it will just use the positions of the remaining neighbor nodes to calculate its velocity, acceleration and position. Therefore, the LASM algorithm can perform well in dynamic networks with nodes adding and leaving rapidly.

3. Simulation Results

Three parts are presented here: the performance evaluation of LASM(B) and LASM(P), comparisons with MDS-MAP algorithm [23], and an initial experiment using sensor nodes designed by us. The simulation software is Matlab, which is the same as in [23]. Simulation environment: $10r \times 10r$ square area, with 200 sensor nodes randomly deployed in it, where r=1 is the unit length. The distance error is presented as follows: If the distance error is e and the true distance is d, the measured distance is a random value with normal distribution $d(1+N(0,e))$ [23]. The position estimation error err is normalized to R, where R is the transmitting range of each node

$$err = \frac{\sum_{i=1}^{n} \sqrt{(x_{ei} - x_i)^2 + (y_{ei} - y_i)^2}}{n}$$

where n is the number of blind nodes, (x_{ei}, y_{ei}) is the estimated position of sensor node i, (x_i, y_i) is the actual position of sensor node i.

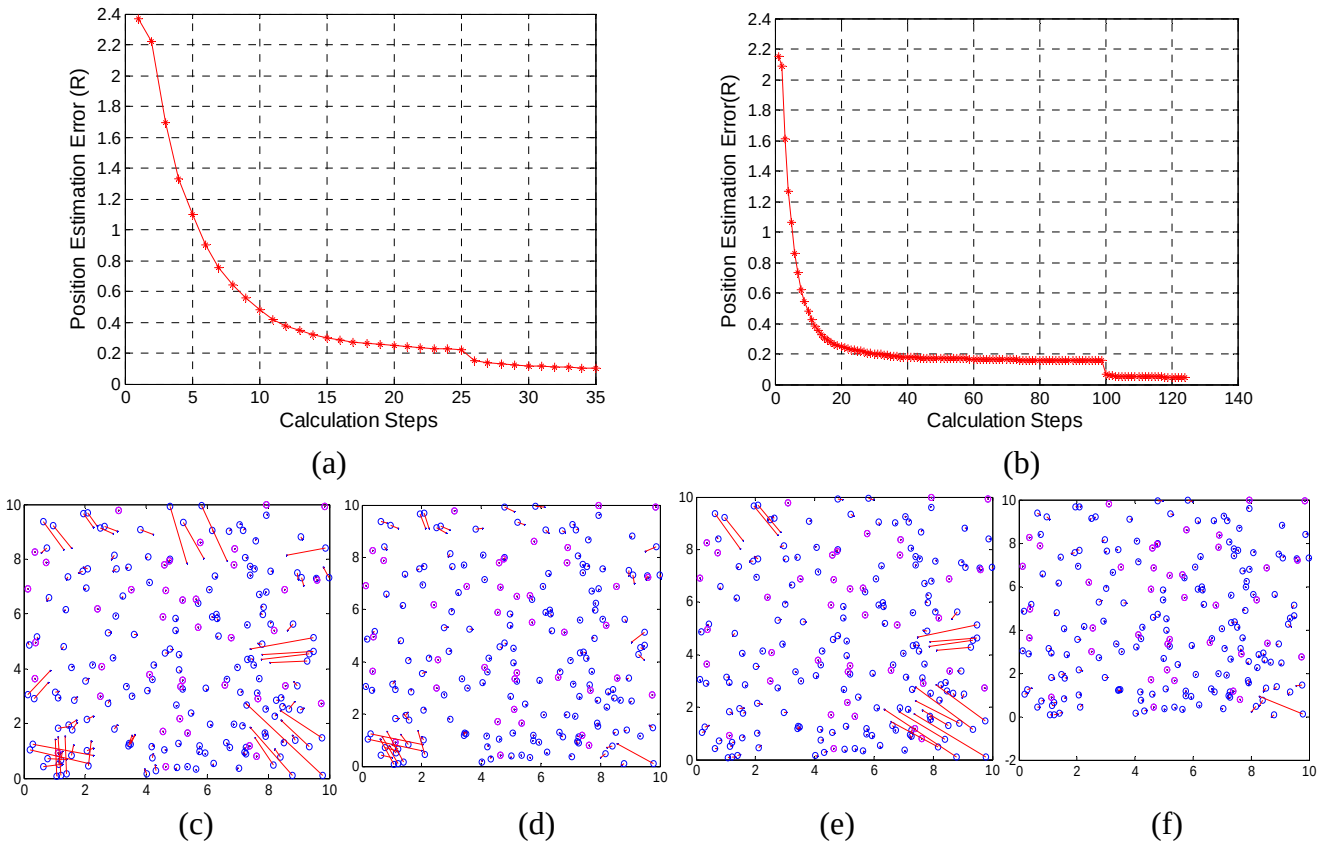
3.1. Performance of LASM(B) and LASM(P)

The simulation process of LASM algorithm is shown in Figure 4. In this experiment, there are 40 anchor nodes (20%) in the network, with radio range R equaling to 1.5r. The distance error is 5%. The mass for each particle m=1, each spring constant k=2, η=2. The thresholds for calculation steps and total force are *lstep*=700, *Tforce*=1 respectively. The constant c=0.2. After the virtual position of each particle is initialized, the position error is about 2.4R. Then these particles are attracted or repulsed by the springs between them. Their virtual positions change quickly. After 25 calculation steps, the total force for each particle is smaller than *Tforce* (=1) and the position error is 0.2238. The result of position estimation for each node is shown in Figure 4(c). The blue circles represent the true positions of blind nodes; the red circles represent the true positions of anchor nodes; the red lines represent the position errors of the nodes; the two endpoints of the line represent the true position and estimated position of the node. Most of these nodes have good position estimations. However, a small number of nodes (about 20 nodes) have large position errors. These nodes are located in needless points (described in Section 2). After we reinitialize the positions of these needless nodes, and use LASM(B) again for one time, which is the process of Patch A, The position error decreases to 0.1014. The result of the position estimation for each node is shown in Figure 4(d). Most of the needless nodes in Figure 4(c) are located in the right positions now. Just a few of them are also located in needless points. We just reinitialize these nodes for one time. We can conclude that if we reinitialize these nodes for several times, the LASM(P) can get a even smaller localization error.

In Figure 4(b) we set *Tforce*=0.1. The position error decreases slowly from step number 25 to 99 and then obtain a relative small position error 0.1562 in LASM(B). The result of position estimation for each node is shown in Figure 4(e). After one time reinitializing, the position error of LASM(P) decreases to 0.0476. Just one node is in needless position in Figure 4(f).

Though it can get a smaller position estimation error by reinitializing several times in LASM(P), the number of calculation steps increases correspondingly. This will increase communication cost and energy consumption. In order to decrease them, we just use one time reinitializing in LASM(P) in the follow experiments and set *Tforce*=0.1.

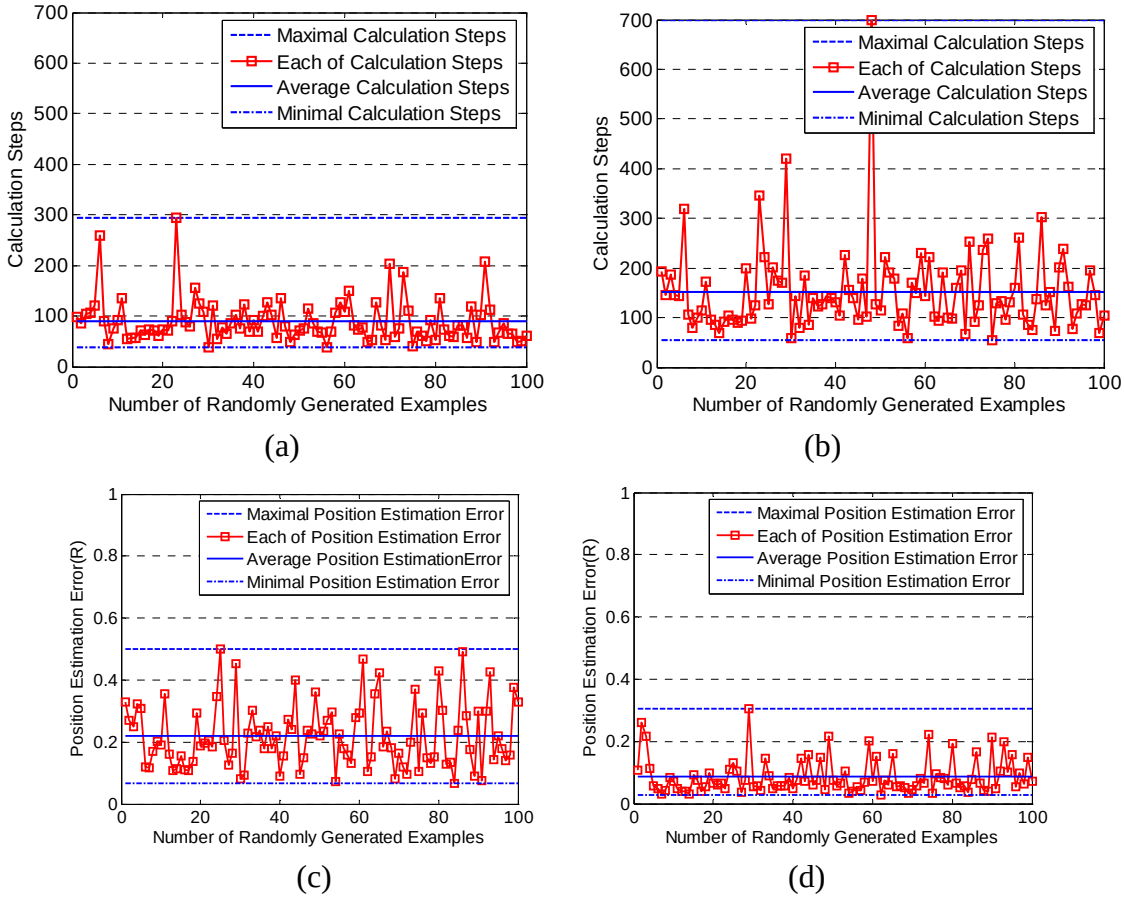
Figure 4. The simulation process and results of LASM. (a) the position estimation error changes in simulation process, *Tforce*=1. (b) *Tforce*=0.1. (c) The result of position estimation in LASM(B), *Tforce*=1. (d) LASM(P), *Tforce*=1. (e) LASM(B), *Tforce*=0.1. (e) LASM(P), *Tforce*=0.1.



The robustness of the LASM algorithm is tested in Figure 5. 100 random examples are generated. In each example the true and virtual positions of nodes are randomly deployed in the square. The other parameter values are the same as those in Figure 4(b). Figures 5(a,b) show general calculation steps the LASM(B and P) have used. In LASM(B), all calculation steps are smaller than 300, and the number of average calculation steps is smaller than 100. This means that the number of calculation steps is very stable, which doesn't change a lot in different examples. In LASM(P), the number of average steps is about 160. This means that in LASM(P), after reinitializing the needless nodes once, the number of average calculation steps to use LASM(B) again is about 60 (160-100). The

Convergence Speed increased after reinitializing again. Figures 5(c,d) show the position estimation errors the LASM(B and P) have obtained. In LASM(B), the largest position error is about 0.5, while the average position error is about 0.2. This also shows that the LASM(B) is very robust. In LASM(P), the average position error is smaller than 0.1. It shows that the LASM(P) can obtain a relatively accurate localization.

Figure 5. The robustness of the LASM algorithm in 100 randomly generated examples. (a) general calculation steps in LASM(B). (b) general calculation steps in LASM(P). (c) position error in LASM(B). (d) position error in LASM(P).



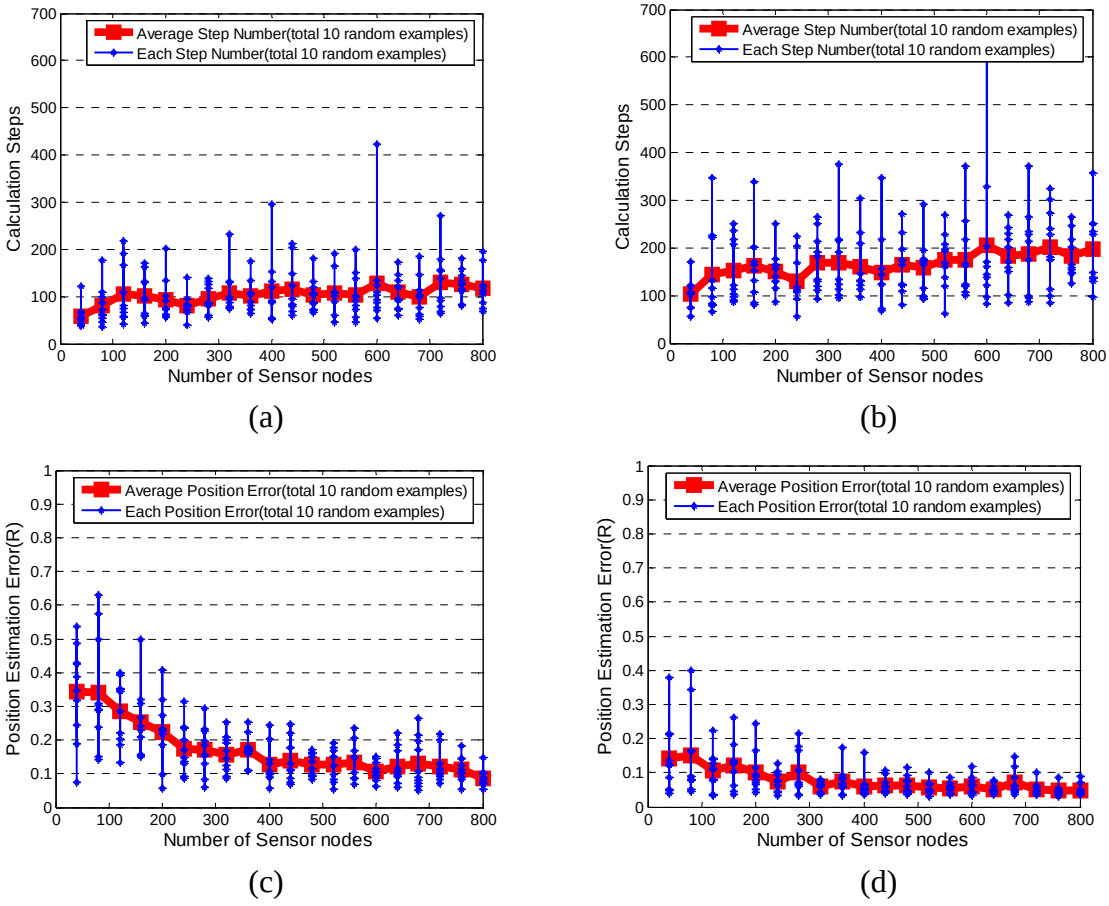
The communication cost and position error versus the number of nodes is tested in Figure 6. Each node just communicates with neighbors once in each calculation step of the LASM. Therefore, the number of calculation steps represents the communication cost. The ratio of anchor nodes is also 20%, with R=1.5. The number of sensor nodes goes from 40 to 800, in increments of 40, i.e., $n = 40 \times i$, and $n_{anc} = 8 \times i$, $L = (10/\sqrt{5}) \times \sqrt{i}$ $(i = 1, 2 \cdots 20)$, where n and n_{anc} are the number of total nodes and anchor nodes respectively, L is the edge length of the square where the nodes are deployed.

The average number of calculation steps doesn't change much versus the number of nodes. As shown in Figures 6(a,b), it is about 100 in LASM(B) and about 150 in LASM(P). It means that the communication cost is almost the same in different sizes of sensor networks. Therefore, we don't test more experiments with the number of nodes larger than 800. It is a good performance that the cost is nearly a constant with the increase of sensor nodes. This means that the LASM localization algorithm

is very appropriate in applications of large sensor networks with large amount of sensor nodes in it. The communication cost for each node is $O(1)$, and the total communication cost is $O(n)$, compared with the MDS-MAP algorithm with $O(n \log n)$ communication cost. Each sensor node executes the localization algorithm in parallel model, and has a computational complexity of $O(1)$, therefore the time consumption for each sensor node is also $O(1)$.

The position error has a small tendency to go smaller as shown in Figures 6(c,d). This shows that when the number of nodes increases, the blind nodes can use more information about other nodes, therefore decrease the position error.

Figure 6. The communication cost and position error versus the number of nodes. (a) average calculation steps versus the number of nodes in LASM(B). (b) average calculation steps in LASM(P). (c) the position error versus the number of nodes in LASM(B). (d) position error in LASM(P).



3.2. Comparisons with MDS-MAP

The MDS-MAP algorithm was chosen because it can also be used in range-based localization algorithms. In addition, thus far it has been used to obtain much higher localization accuracy.

Figure 7(c) shows the position error versus the connectivity in uniform networks. The connectivity of network means the average neighbors each node has. The numbers of anchor nodes and total nodes are 10 and 200 in a $10r \times 10r$ network. R goes from 1.25 to 2.5, in increments of 0.25 [23]. A randomly

generated example is shown in Figure 7(a). When the connectivity is small, the accuracy of LASM is worse than MDS-MAP. When the connectivity is large, e.g., it is 30, the accuracy of LASM and MDS-MAP is slightly similar. In the experiment, just a single reinitializing is used in LASM(P). The accuracy of LASM(P) will get better with more reinitializations. However, this will increase communication cost. The accuracy of localization is decreased in order to obtain a lower communication cost.

Figure 7(d) shows the position error versus the connectivity in irregular networks. The numbers of anchor nodes and total nodes are 10 and 160 in a $10r \times 10r$ C-shaped network. R goes from 1.25 to 2.5, in increments of 0.25 [23]. The average connectivity levels are 8.1975, 11.2931, 14.6406, 18.8562, 22.1138, and 27.6275. A randomly generated example is shown in Figure 7(b). The accuracy of LASM is a little similar with MDS-MAP(P) as shown in [23]. From Figure 7(d) it can be found that the accuracy of LASM is better than MDS-MAP(C) whenever the connectivity is small or large. This is because the MDS-MAP(C) uses the shortest paths between all pairs of nodes replacing their actual Euclidean distances. The complexity of LASM and MDS-MAP is shown in Table 1. It can be found that the computational complexity of LASM is smaller than that of MDS-MAP(C), and the communication complexity of LASM is smaller than that of MDS-MAP(P). The low complexity of the proposed LASM algorithm is more suitable for the large scale network on the moon.

Figure 7. Comparisons with MDS-MAP. (a) a random example in uniform networks. (b) a random example in C-shaped networks. (c) the position error versus connectivity in uniform networks. (d) the position error versus connectivity in C-shaped networks.

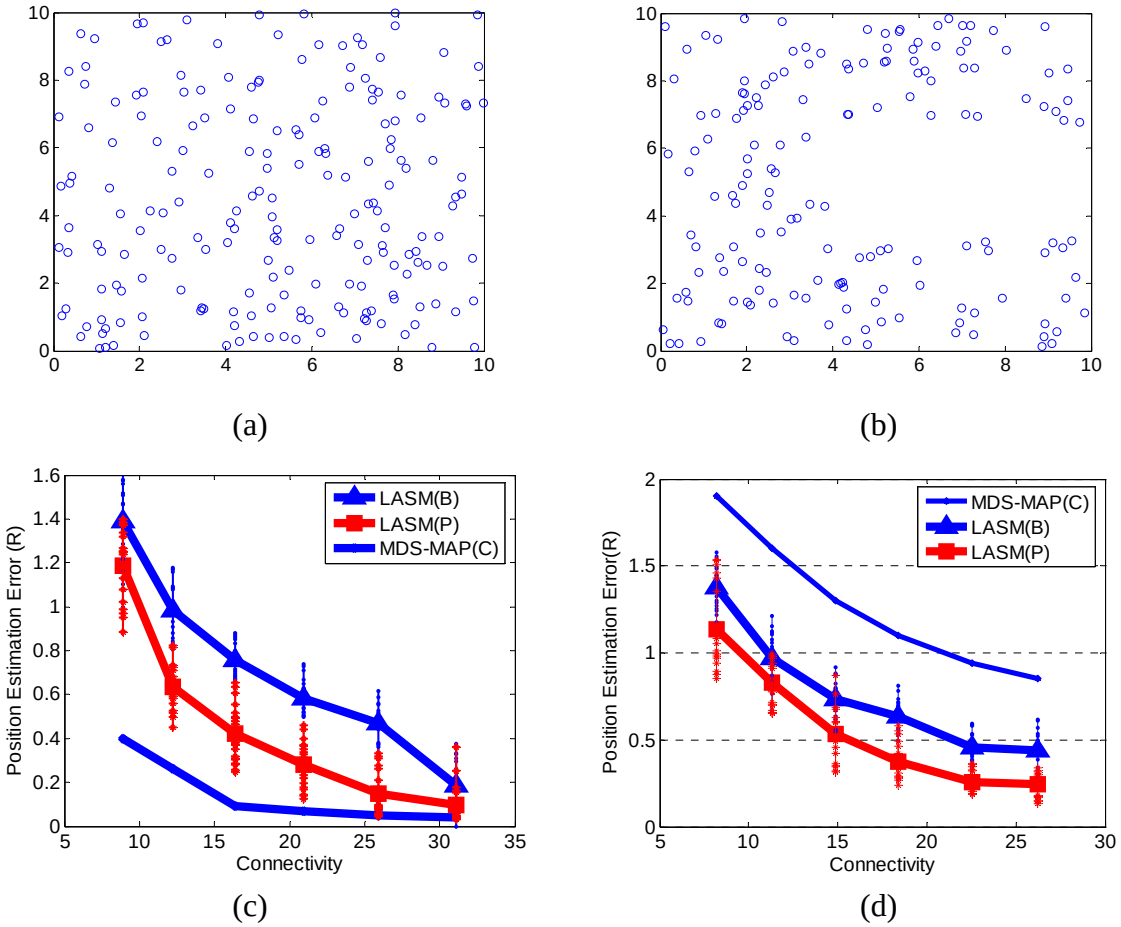


Table 1. The Complexity of LASM and MDS-MAP.

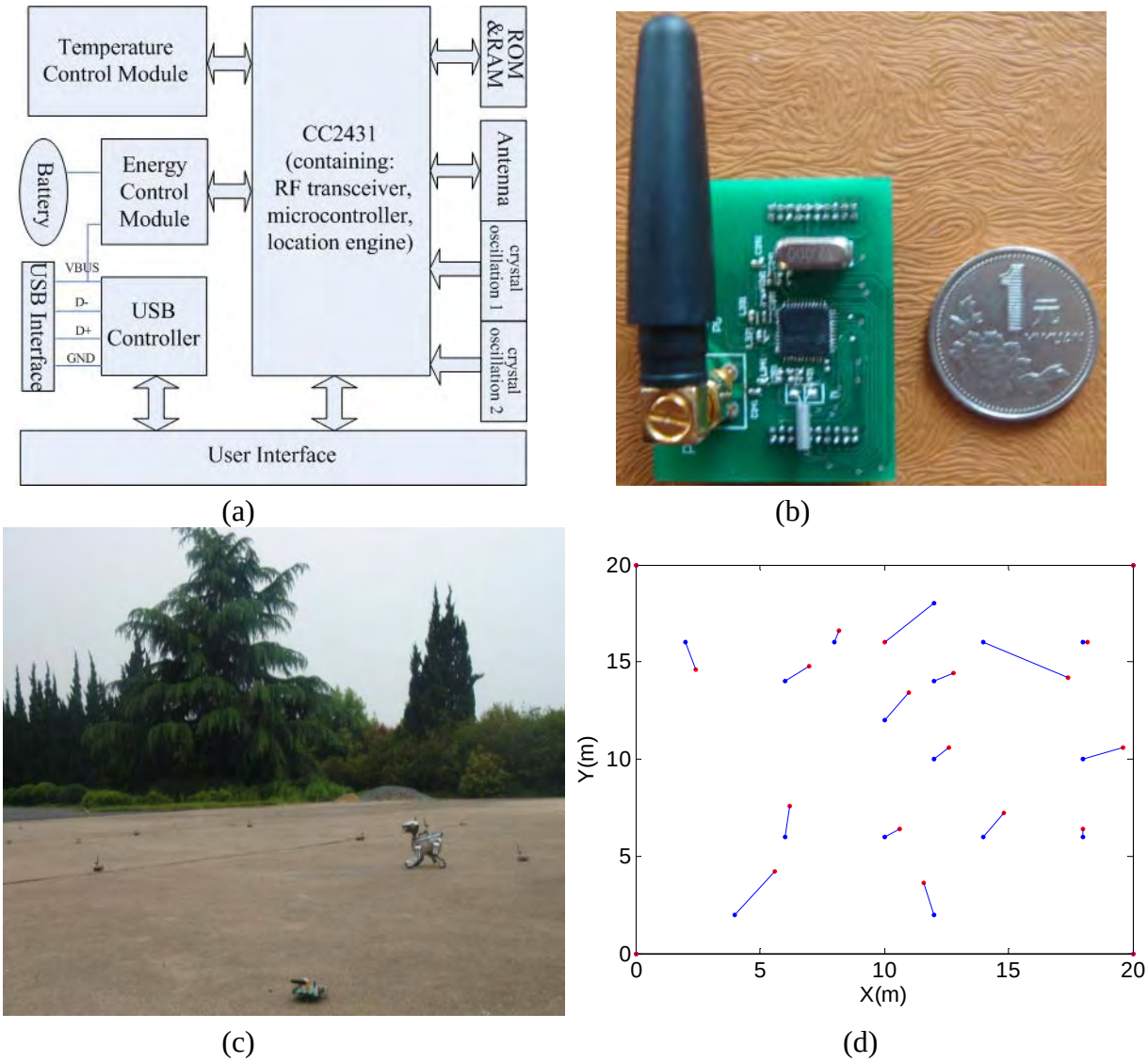
	Distributed LASM(B)	Distributed LASM(P)	MDS-MAP(P)
Communication Cost	$O(1)$*n	$O(1)$*n	$O(n\log n)$
	Centralized LASM(B)	Centralized LASM(P)	MDS-MAP(C)
Computational Complexity	$O(1)$*n	$O(1)$*n	$O(n^3)$

3.3. Initial experiment

An initial experiment was performed to testify the communication performance of sensor nodes designed by us and the LASM with 20 sensor nodes. The wireless sensor node designed by us is shown in Figures 8(a,b). This node mainly consists of a temperature control module, an energy control module, a CC2431, extra ROM&RAM, an RF circuit, a USB controller& interface and a user interface [3]. The temperature on the moon changes from about 102K to 384K [27]. A temperature control module is needed to protect the sensor node from the bad temperatures. A vacuum multiplayer insulation structure is used to passively control the temperature variation. Electric heaters and heat pipes are used to increase the temperature actively if possible [28]. The energy control module is used to make the voltage stable and select the power source. The USB interface is an optional part that is used to communicate with computer. The power can be supplied by either USB or battery. The user interface is an interface that is used for installing various sensors according to the needs of the application. It also gives some communication interfaces such as I2C and UART. CC2431 is the heart of the node, which contains an RF transceiver, a high performance and low-power consumption 8051-based microcontroller, and a location engine. It controls all the other components in the node.

The experiment field is shown in Figure 8(c). The experiment field is a square of 20 m by 20 m with 20 sensor nodes and a robot deployed in it. The anchor nodes are deployed at (0,0),(0,20),(20,0),(20,20). A sensor node is fixed on the robot for the robot's localization and navigation. The localization results are shown in Figure 8(d). The blue point of each line is the actual position for each sensor node, and the red point of each line is estimated position of each sensor node. The position error is about 2 m in experiment. This is a small scale network in initial experiment. It mainly tests the communication quality of wireless sensor nodes and the feasibility of using LASM for nodes localization and robot's localization. In the next phase, we will increase the network to a 1000 m by 1000 m area with about 1000 wireless sensor nodes deployed in it. A lunar rover will be placed in it for localization and navigation.

Figure 8. Experiment design. (a) The proposed sensor node architecture (b) the sensor node designed by us (c) experiment field to test the localization algorithm (d) the result of experiment.



4. Conclusions

After the theoretical study, computer simulation and initial experiments, the LASM algorithm has been proven efficient for large scale wireless sensor networks. The main results are listed as follows:

The localization algorithm based on spring model (LASM) has the advantage of low complexity in computation, communication and time consumption since the calculation steps are almost constant while the node size increases. Three patches are launched to improve the algorithm performances to avoid the local optimum, remove bad nodes and adapt to dynamic changes of the active node. The simulation results of a sensor network with 40 to 800 nodes show that the calculation steps (l) are 150±50 for different node numbers. The localization accuracy is about 10% with the variety of sensor nodes. Hardware of a small scale sensor network has been developed to test the algorithm and the positioning accuracy in initial experiments. Future work includes optimizing the positions of anchor

nodes to obtain the performance of no reinitializing in LASM(P), and increasing the network size for lunar rover localization and navigation.

Acknowledgements

This paper was sponsored by Key Program of the National Natural Science Foundation of China under Grant #60535010 to Max Q.-H. Meng, and the National Natural Science Foundation of China under Grant #60475027 to Max Q.-H. Meng.

References and Notes

1. Deng, Z.; Fang, H.; Dong, Y.; Tao, J. Research on wheel-walking motion control of lunar rover with six cylinder-conical wheels. *IEEE International Conference on Mechatronics and Automation* **2007**, Harbin, China, 388-392.
2. Friedman, L.; Huntress,W. Proposal for an International Lunar Decade. *36th COSPAR Scientific Assembly* 2006, Beijing, China, 2006.
3. Chen, W.M.; Liang, H.W., Mei, T. Design and implementation of wireless sensor network for robot navigation. *International Journal of Information Acquisition* **2007**, *4*, 77-89.
4. Khajehnouri, N.; Sayed, A. Distributed MMSE relay strategies for wireless sensor networks. *IEEE Transactions on Signal Processing* **2007**, *55*, 3336-3348.
5. Wang, X.; Bi, D. W.; Ding, L.; Wang, S. Agent collaborative target localization and classification in wireless sensor networks. *Sensors* **2007**, *7*, 1359-1386.
6. Wang, X.; Wang, S.; Ma, J. An improved particle filter for target tracking in sensor system. *Sensors* **2007**, *7*, 144-156.
7. Wang, X.; Wang, S. Collaborative signal processing for target tracking in distributed wireless sensor networks. *Journal of Parallel and Distributed Computing* **2007**, *67*, 501-515.
8. Wang, S.; Shih, K.; Chang, C. Distributed direction-based localization in wireless sensor networks. *Computer Communications* **2007**, *30*, 1424-1439.
9. Yu, Y; Govindan, R.; Estrin D. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. *UCLA Computer Science Department Technical Report* **2001**, UCLA-CSD-TR-01-0023.
10. Chen, W.M.; Mei, T.; Li, Y.M.; Liang, H.W.; Liu Y.M.; Meng, M. Q.-H. An auto-adaptive routing algorithm for wireless sensor networks. *IEEE International Conference on Information Acquisition* **2007,** Jeju City, Korea, July 2007, 574-578.
11. Liu, P.X.; Liu, Y. A two-hop energy-efficient mesh protocol for wireless sensor networks. *International Journal of Information Acquisition* **2004**, *1*, 237-247.
12. Wang, X.; Ma, J.; Wang, S.; Bi, D. Cluster-based dynamic energy management for collaborative target tracking in wireless sensor networks. *Sensors* **2007**, *7*, 1193-1215.
13. Liu, C.; Wu, K.; Pei, J. An energy-efficient data collection framework for wireless sensor networks by exploiting spatiotemporal correlation. *IEEE Transactions on Parallel and Distributed Systems* **2007**, *18*, 1010-10235.

14. Liu, P.X.; Ding, N. Data gathering and communication for wireless sensor networks-a centralized approach. *International Journal of Information Acquisition* **2005**, *2*, 267-278.
15. Szu, H. Geometric topology for information acquisition by means of smart sensor web. *International Journal of Information Acquisition* **2004**, *1*, 1-22.
16. He, T.; Huang, C.; Lum, B.; Stankovic, J.; delzaher, T. Range-free localization schemes for large scale sensor networks. *In Proc. ACM MobiCom* **2003**, San Diego, CA, Sep. 2003; pp. 81–95.
17. Niculescu, D.; Nath, B. DV based positioning in ad hoc networks. *Telecommunication Systems* **2003**, *22*, 267-280.
18. Savarese, C.; Rabaey, J.; Langendoen, K. Robust positioning algorithms for distributed ad hoc wireless sensor networks. *in: Proceedings of General Track:* **2002**. USENIX Annual Technical Conference, 2002, 317–327.
19. Wong, S.Y.; Lim, J.G.; Rao, S.V.; Seah, W.K.G. Hop-count localization with density and path length awareness in non-uniform wireless sensor networks. *in: Proceedings of IEEE Wireless Communications and Networking Conference* **2005**, 13–17 March 2005, New Orleans, LA, USA, 2005.
20. Sit, T.; Liu, Z.; Jr, M.H.; Seah, W. Multi-robot mobility enhanced hop-count based localization in ad hoc networks. *Robotics and Autonomous Systems* **2007**, *55*, 244–252.
21. Lai, X.Z.; Yang S.X.; Zeng G.X.; She, J.H.; Wu, M. New distributed positioning algorithm based on centroid of circular belt for wireless sensor networks. *International Journal of Automation and Computing* **2007**, *4*, 315-324.
22. Vivekanandan, V.; Wong, V. Concentric anchor beacon localization algorithm for wireless sensor networks. *IEEE Transactions on Vehicular Technology* **2007**, *56*, 2733-2744.
23. Savvides, A.; Han, C.C.; Srivastava, M. Dynamic fine-grained localization in ad hoc networks of sensors. *Proc. ACM/IEEE Int'l Conf. Mobile Computing and Networking (MOBICON)*, July 2001.
24. Shang, Y.; Ruml, W.; Zhang, Y.; Fromherz, M. Localization from connectivity in sensor networks. *IEEE Transactions on Parallel and Distributed Systems* **2004**, *15*, 961-974.
25. Shang, Y.; Ruml, W.; Zhang, K.; Fromherz, M. Localization from mere connectivity. *In ACM MobiHoc* **2003**, Annapolis, MD, June 2003, 201-212.
26. Shang, Y.; Ruml, W. Improved MDS-based localization. *In INFOCOM* **2004**. pp.2640-2651
27. Heiken,G.; Vaniman, D.; French, B. Lunar Sourcebook: A User's Guide to the Moon. Cambridge Univ Press, 1991.
28. Miao, S.F.; Liang, H.W.; Meng, M. Q.-H.; Chen, W.M.; Li, S. Design of wireless sensor networks nodes under lunar environment. *Transducer and Microsystem Technologies* **2007**, *26*, 117-120.