# Real-time Tracking for Sensor Networks via SDP and Gradient Method

[Extended Abstract]

Zizhuo Wang
Department of Management Science and Engineering, Stanford University
zzwang@stanford.edu

Yichuan Ding
Department of Management Science and Engineering, Stanford University
y7ding@stanford.edu

## ABSTRACT

The sensor tracking problem is an important problem studied in many different fields. But many of those studies use analysis or machine learning method rather than optimization method. Recently, several approaches have been proposed to solve the static version of the tracking problem, the sensor network localization problem, via Semi-definite Programming(SDP). In this paper, we analyze a new real-time sensor tracking scheme by combining the SDP approach and the gradient method. We show that this approach provides fast and accurate tracking for network sensors. We also discuss the problem of extracting information from the moving sensors, which could be used to predict their movements.

## Categories and Subject Descriptors

G.1 [**Numerical Analysis**]: Optimization

## General Terms

Performance

## Keywords

Semidefinite Programming, Sensor Network Localization, Tracking

## 1. INTRODUCTION

Recent years have witnessed a fast increasing trend of using sensor networks to track targets across certain geometric regions, e.g., fire fighting commanders want to track each fireman in a fire fighting action; chemists want to track the movement of the molecules in a chemical reaction. Typically, networks of this type consist of a large number of densely deployed sensors which gather local data and communicate with other nearby sensors and anchors. The Global Positioning System(GPS) could be used, but is usually too expensive to be practical or otherwise impossible to implement in some situation. Hence, alternative approach to track these sensors is desirable.

In the sensor tracking problems, we are usually given dynamic data concerning the Euclidean distance between the sensors. Usually the distance is known between two sensors if they are close enough. During the tracking process, the set of the pairs of sensors whose distance are known may change due to the movement of the sensors. In the real-time tracking, our task is to find a way to update the locations of the sensors instantaneously, and predict their future trajectories based on the dynamic data we have had.

The tracking problem usually involves two stages. At first stage, we get a rough localization of the sensors, while in the second stage, we observe and analyze the dynamic information. We will use a new proposed further relaxation of SDP approach[7] to solve the first stage localization problem, and use gradient method to track the movements. We will also discuss how to extract useful data and make prediction for the future movement of the sensors.

The paper is organized as follows. Chapter 2 formalizes the sensor tracking problem and introduces the SSDP approach proposed by [7] for solving the first stage problem. Chapter 3 discusses the tracking procedure. Chapter 4 discusses the problem of how to extract data and make prediction for the movement of sensors. Numerical results will be presented in Chapter 5. We will conclude in Chapter 6.

## 2. FORMULATION AND BACKGROUND

The mathematical model of the sensor tracking problem can be described as follows. There are $n$ distinct moving sensors in $R^d$ that we want to track, and $m$ other fixed points (called the anchor points) whose locations, $a_1, a_2, \dots, a_m$, are known and fixed during the whole process. The Euclidean distance $d_{ij}(t)$ between the $i$th and $j$th sensor points at time $t$ is known if they are close enough, namely, $(i,j) \in N_x(t)$, where $N_x(t) = \{(i,j) : \|x_i(t) - x_j(t)\| = d_{ij}(t) \le r_d\}$, $r_d$ is a fixed parameter called radio range. Similarly, the distance $\bar{d}_{ik}(t)$ between the $i$th sensor and $k$th anchor points at time $t$ is known if $(i,k) \in N_a(t)$ where $N_a = \{(i,k) : \|x_i(t) - a_k\| = \bar{d}_{ik}(t) \le r_d\}$. The problem is to find $x_i(t) \in R^d$, $i = 1, 2, \dots, n$ for which

$$\begin{array}{ll} \|x_i(t) - x_j(t)\|^2 = d_{ij}^2(t) & \forall\ (i,j) \in N_x(t) \\ \|x_i(t) - a_k\|^2 = \bar{d}_{ik}^2(t) & \forall\ (i,k) \in N_a(t). \end{array}$$

for all $t$. Unfortunately, this problem is hard to solve in general even for the static version and for $d = 2$; see, e.g., [8].

For simplicity, we will restrict ourselves to the case of $d = 2$. Started from this decade, semi-definite programming(SDP) approach has been widely studied and used for the static version of the problem, the sensor network localization problems. Many theoretical results are given for this problem as well as numerical results, see, e.g., [3, 1, 2, 6, 4, 5, 7]. However, the low-efficiency of the SDP for large problems has prevented the practical usage of this approach. Fortunately, in recent two years, two main approaches have been proposed to greatly improve the efficiency of the SDP approach for the problem. One is the Sub-SDP approach proposed in [7] that introduces a further relaxation of the original SDP model. The other is proposed by [6] by using region decomposition method. We will mainly adopt the first approach in this paper since it is aimed to solve one single SDP and can be incorporated in the decomposition techinique.

To start with, we will first give the general SDP relaxation formulation for the sensor localization problem, for detail of this, see [4],[1],[2]:

$$\begin{array}{ll} \text{minimize} & \mathbf{0} \bullet Z \\ \text{subject to} & Z_{(1,2)} = I, \\ & (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^T \bullet Z = d_{ij}^2, \\ & \quad \forall\ (i,j) \in N_x \\ & (-a_k; e_i)(-a_k; e_i)^T \bullet Z = \bar{d}_{ik}^2, \\ & \quad \forall\ (i,k) \in N_a \\ & Z \succeq 0 \end{array} \tag{1}$$

where $N_i = \{j : (i,j) \in N_x\}$ is the sensor-$i$-connected point set in a static setting, and

$$Z = \begin{pmatrix} I & X \\ X^T & Y \end{pmatrix}$$

is the variable consisting of the first and second order information of the localization. In [7], two further relaxed models are given. The first is a node-based relaxation model called the NSDP relaxation:

$$\begin{array}{ll} \text{minimize} & \mathbf{0} \bullet Z \\ \text{subject to} & Z_{(1,2)} = I, \\ & (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^T \bullet Z = d_{ij}^2, \\ & \quad \forall\ (i,j) \in N_x \\ & (-a_k; e_i)(-a_k; e_i)^T \bullet Z = \bar{d}_{ik}^2, \\ & \quad \forall\ (i,k) \in N_a \\ & Z^i = Z_{(1,2,i,N_i)} \succeq 0, \quad \forall i \end{array} \tag{2}$$

Here, $Z_{(i_1,\ldots,i_k)}$ denotes the principal submatrix consistituted of the rows and columns indexed by $i_1, \ldots, i_k$. Compared with the original SDP approach, the single $(2+n)$-dimensional matrix cone is replaced by $n$ smaller $3 + |N_i|$-dimensional matrix cones, each of which is a principal submatrix of $Z$.

The second relaxation is an edge-based relaxation which they called the ESDP relaxation:

$$\begin{array}{ll} \text{minimize} & \mathbf{0} \bullet Z \\ \text{subject to} & Z_{(1,2)} = I, \\ & (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^T \bullet Z = d_{ij}^2, \\ & \quad \forall\ (i,j) \in N_x \\ & (-a_k; e_i)(-a_k; e_i)^T \bullet Z = \bar{d}_{ik}^2, \\ & \quad \forall\ (i,k) \in N_a \\ & Z_{(1,2,i,j)} \succeq 0, \quad \forall (i,j) \in N_x. \end{array} \tag{3}$$

In this case, the single $(2+n)$-dimensional matrix cone is replaced by $|N_x|$ smaller 4-dimensional matrix cones, each of which is a principal submatrix of $Z$.

They showed in [7] that ESDP is weaker than SDP but is far more efficient. In practical computation, they use a dual ESDP method which computes from the dual problem of (3) rather than the primal. Their experimental tests on PC showed that dual ESDP can solve problems up to 5000 sensors in about 200 seconds with high accuracy.

The above results enable us to track the sensors using the ESDP approach as the first stage localization scheme. In the next chapter, we will fully discuss the method to track the sensors in a real-time setting.

## 3. SENSOR TRACKING SCHEME

In the above chapter, we review the NSDP and ESDP method for solving the sensor network localization problem. Although ESDP is shown to be quite efficient in locating each sensor given a static distance data, it is still not fast enough to perform instantaneous location updating during the movement of the objects. However, once we have an initial data of the location of each sensor, we can apply the simple gradient method to accomplish the real-time update process. We will formalize the above discussion as follows.

First we discretize the time space to $0 = t_0 < t_1 < \ldots < t_n \ldots$. The time interval is determined by the computation capability which we will discuss in chapter 5. At time $t_n$, the data we have is an $n+m$ by $n+m$ partial matrix $D_n$ where

$$D_n(i,j) = \begin{cases} d_{i-m,j-m}(t_n) & \text{if } i > m, j > m \\ \bar{d}_{i-m,j}(t_n) & \text{if } i > m, j \le m \\ \bar{d}_{i,j-m}(t_n) & \text{if } j > m, i \le m \\ 0 & \mathit{otherwise} \end{cases}$$

For each time $t_n$, we try to minimize the error of the location matrix $X_n$, which we define as follows:

$$\begin{array}{l} \epsilon_{t_n}(X_n) \\ = \sum_{(i,j)\in N_x(t_n)}(||x_n(i) - x_n(j)|| - d_{ij}(t_n))^2 \\ + \sum_{(i,k)\in N_a(t_n)}(||x_n(i) - a_k(t_n)|| - \bar{d}_{ik}(t_n))^2 \end{array} \tag{4}$$

where $x_n(i)$ is the $i$th column of $X_n$. Note that $\epsilon_{t_n}(X_n) = 0$ means that $X_n$ satisfies all the distance information. If we directly apply the gradient method to find the minimum of (4) taking 0 as the initial point, as [1] shows, the results will not make sense. This is because $\epsilon_{t_n}$ is not a convex function, thus the gradient method will be likely to stop in local minimums other than the global minimum. However, if we have a close enough initial point, namely $X_{n-1}$ in this case, the gradient method may give accurate localization for the next time slot. This gradient method was also used in [1] for refining the SDP solution for static sensor network localization problems.

Now we summarize our sensor tracking scheme as follow:

**Sensor Tracking Scheme:**

1. Step 1

   Initialize: Input initial data $D(0)$, set accuracy $\delta$ and maximum iteration number *miter*;

2. Step 2

Use Dual ESDP to solve for an initial localization for all the sensors, which we denote by $X_0$;

3. Step 3

   At each time $t_n$, input the distance data $D_n$, Apply gradient method to solve the minimum of $\epsilon_{t_n}$, taking $X_{n-1}$ as the initial point. Stop when $\epsilon_{t_n} < \delta$ or the iteration number exceeds *miter*.

To implement the gradient method, we refer to the discussion in [1]. Note that $\bigtriangledown_x ||x-y|| = \frac{x-y}{||x-y||}$, we have

$$\begin{array}{l} \bigtriangledown_{x_n(i)} \epsilon_{t_n}(X_n) = \\ 2\sum_{(i,j)\in N_x(t_n)}(1-\frac{d_{ij}(t_n)}{||x_n(i)-x_n(j)||})(x_n(i)-x_n(j)) \\ +2\sum_{(i,k)\in N_a(t_n)}(1-\frac{d_{ik}(t_n)}{||x_n(i)-a_k||})(x_n(i)-a_k) \end{array} \quad (5)$$

And we update the location according to the formula

$$x_n^{k+1}(i) = x_n^k(i) + \alpha \bigtriangledown_{x_n(i)} \epsilon_{t_n}(X_n^k)$$

in each iteration until we reach the given accuracy $\delta$ or reach the maximum iteration number. Notice that the gradient with respect to $x_n(i)$ only involves the sensor and anchors that are connected to sensor $i$. The descent direction can be computed through a distributed fashion. To determine the step size $\alpha$, we use the back-tracking line search method. The detail of this method is referred to [9].

## 4. RETRIEVING MOVING FUNCTION AND PREDICTION

In real-time tracking, it is often desirable to extract certain dynaimic information. These information can be used to analyze the motion and predict the future movements. For example, we might fail to note the distance matrix for a certain time slot, then the information can be used to approximate the data at that time; in another case, the movement of the object is too fast such that the gradient method fails to give an accurate solution if we start from the position of the last time slot, then using the predicted localization as the starting point for the gradient method may work. The above discussion motivates us to the analysis below.

Suppose the movement of the objects is subject to a certain class of function. This class of function should be predetermined by the property of the objects. A general setting will be to assume that the objects are moving according to the differential equation:

$$\frac{dX(t)}{dt} = F(X(t), t)$$

Here, $X(t)$ is a column vector consisting of the positions of all sensors. To make the problem tractable, we further assume that the objects are moving according to a linear differential equation:

$$\frac{dX(t)}{dt} = AX(t) + Zt + C$$

where $A, C, Z$ are the coefficient matrices to be determined. If we have already acquired the data of the movement at time $t_0, t_1, ..., t_n$, then by using the difference $\frac{X(t_n)-X(t_{n-1})}{t_n - t_{n-1}}$ as the approximate derivative at time $t_n$, we will get a system of approximate linear equations, namely,

| # | $n$ | $m$ | $rd$ | DualESDP time |
|---|---|---|---|---|
| 1 | 50 | 5 | 0.35 | 1.22 sec |
| 2 | 100 | 5 | 0.3 | 1.91 sec |
| 3 | 200 | 5 | 0.25 | 4.19 sec |
| 4 | 400 | 10 | 0.2 | 8.98 sec |
| 5 | 800 | 20 | 0.12 | 18.58 sec |
| 6 | 1600 | 40 | 0.07 | 43.91 sec |
| 7 | 3200 | 80 | 0.04 | 104.36 sec |
| 8 | 5000 | 100 | 0.03 | 192.08 sec |
| 9 | 6400 | 160 | 0.025 | 250.97 sec |

**Table 1: The First Stage Problem**

$$\begin{array}{l} \frac{X(t_i)-X(t_{i-1})}{t_i - t_{i-1}} \simeq AX(t_i) + Zt_i + C \\ \qquad \forall i = 1, 2, ..., n \end{array}$$

To determine $A, C$ and $Z$, we minimize the error term. In order to predict the future, the current information is usually more important than the previous ones, so one could put an increasing sequence of weights $\{\eta_i\}$ before each error. Then at time $t_n$ ,we will have the following problem:

$$\begin{array}{ll} \min & \sum_{i=1}^n \eta_i \gamma_i^2 \\ \text{s.t.} & \gamma_i \geq \frac{X(t_i)-X(t_{i-1})}{t_i-t_{i-1}} - AX(t_i) - Zt_i - C \\ & \qquad \forall i = 1, 2, ..., n \\ & \gamma_i \geq -\frac{X(t_i)-X(t_{i-1})}{t_i-t_{i-1}} + AX(t_i) + Zt_i + C \\ & \qquad \forall i = 1, 2, ...n \\ & A, C, Z \in \Lambda \end{array} \quad (6)$$

This is essentially a least square problem. To make this problem easier to solve, one can restrict the number of nonzero elements of $\eta_i$. As soon as we get the information of $A, C$ and $Z$ we can make predictions about the future movement of the objects.

## 5. NUMERICAL RESULTS

In this chapter, we present two parts of numerical results. First, we show how fast the first stage problem can be performed. This part of data is referred to the numerical results presented in [7], see Table 1.

In Table 1, $n$ denotes the total number of points, including both sensors and anchors, $m$ denotes the amount of anchors which are deployed uniformly in the region. $rd$ is the radio range. DualESDP time is the time used to compute the localization using dual ESDP. The computation was carried out in a Laptop with 1.99GB Memory and 1.06GHz CPU.

Table 1 shows that the first stage problem can be implemented efficiently for small problems. Even for problems with several thousand sensors, it requires only several minutes. Next we show how fast the second stage problem can be done. In the experiments, we assume that the sensors are moving in a one by one squared region, at a maximum speed of $v$ units per time interval. That is, in the experiments, each movement is a vector of random direction and of norm chosen uniformly between 0 and $v$. These experiments are done on a laptop with 1.99GB Memory and 1.66GHz CPU. The MATLAB codes can be found in www.stanford.edu/ yyye/ESDPDTracking.p

In Table 2, the problem setting is exactly the same with the one in the corresponding row in Table 1. The last column

| # | $n$ | $m$ | $rd$ | Gradient Methods |
|---|---|---|---|---|
| 1 | 50 | 5 | 0.35 | 0.01 sec |
| 2 | 100 | 5 | 0.3 | 0.02 sec |
| 3 | 200 | 5 | 0.25 | 0.05 sec |
| 4 | 400 | 10 | 0.2 | 0.15 sec |
| 5 | 800 | 20 | 0.12 | 0.35 sec |
| 6 | 1600 | 40 | 0.07 | 1.05 sec |
| 7 | 3200 | 80 | 0.04 | 2.97 sec |

**Table 2: The Second Stage Problem**

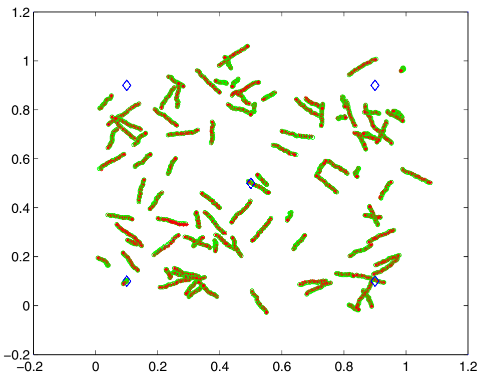

**Figure 1: Tracking Result at 20 movements**

shows the time required to perform one gradient procedure. In this table, we set the moving speed $v$ to be 0.01, and the accuracy threshold to stop the gradient method is 1e-7. The localization quality remains high during the tracking process.

The above data shows that for 1600 sensors moving at a speed not exceeding 0.01 unit per second, we can track them with no problem. Even for more sensors or faster moving speed, we can lower the accuracy in each gradient step to expediate the updating process. We test a problem with 3200 sensors moving at a speed of 0.01 unit per second. We found that by setting the accuracy in the gradient step to be 1e-5, we can update the data with an average 0.45 sec per step and the RMSD(Root Mean Square Deviance) of the localization is 8.3e-3 after 20 steps compared to 7e-3 at the first step.

Next we show some graphical results. We show a tracking trajectory for a graph of 100 sensors, 5 anchors with radius range 0.3 and a movingspeed of 0.01 unit per second. In figure 1, the green trajectory is the true trajectory of the sensors and the red one is the computed trajectory. The blue diamonds are the anchor points. We can see from the graphs that the tracking quality is high. In fact, the RMSD is around 0.01 through the whole process.

## 6. CONCLUSION

In this paper, we discuss how to track moving sensors in a real-time fashion. We combine the SDP approach for sensor network localization problem and the gradient method to get a practical way to track the moving objects. We also present methods to extract information from the movement and to make relative predictions. Numerical result shows that this method is quite efficient even when the problem is fairly large.

There are still several problems to be solved. One important issue is the initialization of the tracking process. In this paper, we assume that we can compute the initial location of the sensors before they start to move. This is practical when people can control the start of the movement. But sometimes, it is not the case. Then we need to incorporate the solving SDP process into the tracking process, i.e., when we solve the SDP, the data is updating according to the new information. Certain decomposition method may be appealing in this setting.

## 7. REFERENCES

[1] P. Biswas, T.-C. Liang, K.-C. Toh, T.-C. Wang and Y. Ye. Semidefinite Programming Approaches for Sensor Network Localization with Noisy Distance Measurements. IEEE Transactions on Automation Science and Engineering, 2006, pp. 360-371.

[2] P. Biswas, T.-C. Liang, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. ACM Trans. Sen. Netw., 2(2006), pp. 188-220.

[3] P. Biswas, Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on, 2004, pp. 46-54.

[4] A. M.-C. So and Y. Ye. Theory of semidefinite programming relaxation for sensor network localization. SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, 2005, pp. 405-414.

[5] A. M.-C. So and Y. Ye. A semidefinite programming approach to tensegrity theory and realizability of graphs. SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, 2006, pp. 766-775.

[6] M. Carter and H. H. Jin and M. A. Saunders and Y. Ye. Spaseloc: An Adaptable Subproblem Algorithm for Scalable Wireless Network Localization. SIAM J. on Optimization, 2006, pp. 1102-1128.

[7] Z. Wang, S. Zheng, S. Boyd and Y. Ye. Further Relaxation of SDP approach to Sensor Network Localization To appear in SIAM J.on Optimization.

[8] J. Aspnes, D. Goldenberg, R. Yang. On the Computational Complexity of Sensor Network Localization. *ALGOSENSORS 2004*, in *LNCS* 3121:32–44, 2004.

[9] S. Boyd. Convex Optimization Cambridge University Press.