Placement and Distributed Deployment of Sensor Teams for Triangulation Based Localization

Volkan Isler
Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180
Email: isler@cs.rpi.edu

***Abstract*— We address the problem of placing a sensor network so as to minimize the uncertainty in estimating the position of targets. The novelty of our formulation is in the sensing model: we focus on stereo sensors where the measurements from two sensors must be combined for an estimation.**

We study two versions of this problem. In the first version, which we call the placement problem, we are given a workspace and an error threshold. The objective is to place a minimum number of cameras so that no matter where the target is located in the workspace, the uncertainty in localizing it is less than the threshold. For this problem, we present an approximation algorithm and prove that the deviation of its performence from the optimal value is bounded by a constant. In the second version, called the deployment problem, we study the problem of relocating a mobile sensor team to minimize the uncertainty in localizing possibly moving targets. We present a distributed, discrete-time algorithm which explicity addresses communication and motion constraints and show how to compute the optimal move within the time-step for a given target/sensor-pair assignment. The utility of the algorithm is demonstrated with simulations.

I. Introduction

Concurrent advances in robotics, embedded sensing, computation and communication technologies opened the way for distributed and mobile sensing networks. Such networks are becoming increasingly popular in automation applications such as surveillance, inventory control and traffic management.

Sensor network technology has important implications for mobile robotics. As an example, consider the localization problem whose solution is a prerequisite for many robotics applications. Sensor network technology offers a scalable solution for localization of heterogeneous, independent robot teams operating in a large and complex environment: We can deploy a calibrated camera-network in such an environment and the robots can query these sensors for localization instead of relying on on-board sensors and customized applications.

In this paper, we address the sensor placement problem which is a fundamental issue affecting the quality of localization. In the first part of the paper, we address the placement problem for camera networks. As is well known, a robot cannot localize itself with a single measurement from a single camera. At least two different camera measurements are required for triangulation. However, the quality of the localization is a function of the target-camera geometry. We consider a scenario where the location of the cameras are known apriori to the robot. To localize itself, the robot queries two cameras and merges their measurements. The problem we address is: *given the workspace and an error threshold, what is the minimum number, and placement of cameras so that the error in localization is less than the threshold at every point in the workspace?* For this problem, we present an approximation algorithm that deviates from the optimal solution only by a constant factor both in the number of cameras used and the error in localization. To the best of our knowledge, this is the first approximation algorithm with provable performance for this problem.

In the second part of the paper, we study the placement problem for mobile, ad-hoc sensor networks. We consider a tracking scenario where a number of targets are moving independently in the workspace and the goal is to relocate the sensor-network nodes to improve the localization of targets. For ease of reference, we will refer to the second problem as the *deployment problem* whereas the first problem will be called the *placement problem*. Deployment differs from placement in two aspects: communication constraints and motion constraints.

As in most ad-hoc networks, the underlying communication structure of the network may change as the network nodes move. Therefore, not every pair of nodes may be able to communicate with each other. To address this issue, we include a communication graph (which may change over time) in our formulation and restrict pairwise measurements to the edges of this graph. The idea here is that, merging information from other pairs require multi-hop communication which is very costly in terms of energy – especially for cameras.

The second challenge is to address motion constraints. For mobile sensors, the set of locations reachable from a given location in a single time step is relatively constrained. This constraint must be incorporated (and exploited) in the problem formulation.

In order to address these challenges, we propose a discrete-time solution to the deployment problem. We focus on the case where the underlying communication structure is a tree. We present a distributed algorithm that computes the optimal next-location for each robot for given sensor-target assignments. Note that, due to the fact that the estimations involve pairs of sensors, the sensors can not compute their next move independently and some collaboration is needed. We present a protocol that addresses this issue explicitly.

A. Related work

The placement problem can be viewed as a clustering problem. For example, in the NP-complete k-center problem, we are given a set of locations for centers and a set of targets. The objective is to minimize the maximum distance between any target and the center closest to it [1]. In our problem, there are two centers associated with each target and the cost is much more involved than the Euclidean distance. We were unable to find any literature that addresses this type of clustering problem. Therefore, the placement algorithm presented in the next section may also be of independent interest.

Perhaps, the most well-known placement problem that involves cameras is the Art Gallery Problem [2] where a minimum number of omnidirectional cameras is sought to guard every point in a gallery represented by a polygon. Art gallery problems emphasize visibility/occlusion issues and there is no explicit representation of the quality of guarding – which is the focus of this paper.

Coverage and placement problems received a lot of attention recently. The problem of relocating sensors to improve coverage has been studied in [3]. In this formulation, the sensors can individually estimate the positions of the targets. However, the quality of coverage decreases with increasing distance. The deployment algorithm presented in the present paper studies a similar problem but for sensors which can not estimate the targets' positions by themselves.

The problem of controlling the configuration of a sensor team which employs triangulation for estimation has been studied in [4] where the authors present a numerical, particle-filter based framework. In their approach, the optimal move at each time step is estimated by computing an n dimensional gradient numerically where n is the size of the joint configuration space of the team. Further, the issue of multiple targets is not explicitly addressed. In this paper, we present a dynamic programming based approach to compute the best move in a distributed and optimal fashion.

The problem of choosing the best subset of cameras for a given placement has been studied recently in [5]. In this work, the focus is on selecting a small subset of cameras to minimize a joint uncertainty measure. In the present work, we restrict ourselves to stereo-pairs but focus on placement issues while addressing communication and motion constraints. A recent related result was presented in [6] where the problem of relocating a sensor team whose members are restricted to lie on a circle and charged with jointly estimating the location of the targets was studied.

II. An approximation algorithm for the placement problem

In this section, we study the placement problem and present an approximation algorithm. Before we formalize the placement problem, let us present the error model we use for triangulation.

A. Error model

The term triangulation refers to inferring the state $\vec{x}$ of a target by solving a system of simultaneous equations $\vec{z} = h(\vec{x})$ where $\vec{z}$ denotes the observation vector. In this section, we will study the process of estimating the position $\vec{x} = [x \;\; y]$ of a target (or a robot) from measurements by two cameras. We assume calibrated cameras, hence their location are known with respect to a common reference frame and their measurements can be interpreted as angles with respect to the horizontal axis (see Figure 1).

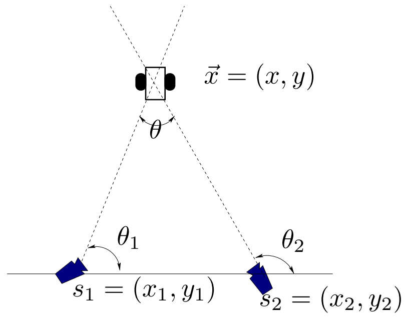

Fig. 1. The uncertainty in estimating the position of the target at x is given by: $U(s_1, s_2, x) = \frac{d(s_1,x) \times d(s_2,x)}{\sin\theta}$

In this case, we have observables θ_1 and θ_2 and solve for the unknowns x and y in:

$$\tan\theta_1 = \frac{y_1 - y}{x_1 - x} \quad \tan\theta_2 = \frac{y_2 - y}{x_2 - x}$$

One way of establishing the accuracy of the estimation is to study the effect of small variations in the observables on the estimate. One way of formally establishing this effect is to study the determinant of the Jacobian $H = \frac{\delta h}{\delta \vec{x}}$ which is commonly referred to as the Geometric Dilution of Precision (GDOP). In case of cameras $GDOP$ is given by

$$U(s_1, s_2, x) = \frac{d(s_1, x) \times d(s_2, x)}{|\sin \angle s_1 x s_2|} \tag{1}$$

where $d(x, y)$ denotes the Euclidean distance between x and y and $\theta = \angle s_1 x s_2$ is the angle between the sensors and the target (Figure 1). The details of this derivation can be found in [7]. In general, Equation 1 suggests that better measurements are obtained when the sensors are closer to the target and the angle is as close to 90 degrees as possible.

B. Problem formulation

Let $\mathcal{W}$ be the workspace which consists of all possible locations of the robot. Let $U(s_i, s_j, w)$ denote the uncertainty in localization when the robot is at location $w \in \mathcal{W}$ and queries sensors s_i and s_j as defined in Equation 1. Let $S = \{s_1, \ldots, s_n\}$ be a set of sensors. When there is no danger of confusion, we will use s_i to denote the location of sensor i as well. For a given placement S and a location $w \in \mathcal{W}$, let $assign(w, S) = \arg\min_{s_i, s_j \in S} U(s_i, s_j, w)$ be the assignment function which chooses the best pair of sensors for location w.

The uncertainty of a placement is defined as $U(S, \mathcal{W}) = \max_{w \in \mathcal{W}} U(w, assign(w, S))$.

We can now define the sensor placement problem:

Given a workspace $\mathcal{W}$ and an uncertainty threshold U^*,

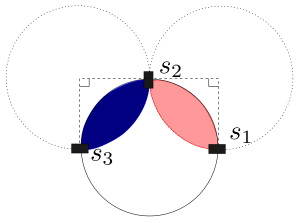

Fig. 2. The shaded areas show the set of points that see s_1s_2 and s_2s_3 with an angle greater than 135 degrees.

find a placement S with minimum cardinality such that $U(S,\mathcal{W}) \leq U^*$.

In the next section, we present an approximation algorithm for the placement problem.

C. Placement algorithm

Let OPT be an optimal solution for the placement problem. In this section, we present an algorithm to compute a placement S with $|S| < 3 \times |OPT|$ and $U(S,\mathcal{W}) \leq c \times U(OPT,\mathcal{W}) \leq c \times U^*$ where U^* is the uncertainty threshold and c is a constant. Let us call such a placement as a *competitive placement*. We assume that $\mathcal{W} \subseteq \mathbb{R}^2$ and cameras can be placed on the entire plane. However, as we will see shortly, no competitive placement can afford to place cameras too far from the workspace.

Let $R = \sqrt{U^*}$. The proposed placement algorithm consists of two phases. In the first phase, we choose a set of centers which will be used to determine the location of the sensors. In the second phase, we place sensors on circles whose centers coincide with the chosen centers and whose radii are at most $2R$. We will show that this placement is a competitive one.

The centers are chosen by the following algorithm:

Algorithm selectCenters(workspace $\mathcal{W}$):

```
• $C = \emptyset$
• while $W \neq \emptyset$
    – $w \leftarrow$ an arbitrary point in $\mathcal{W}$
    – $C \leftarrow C \cup \{w\}$
    – $W \leftarrow W \setminus \{x : d(x,w) < 2R\}$
```

The following lemma shows that the number of centers is small with respect to $|OPT|$.

Lemma 1: Let C be the set of centers chosen by selectCenters and OPT be an optimal placement. $|OPT| \geq |C|$.

Proof: For each center $c \in C$, let us define D_c to be a disk centered at c with radius R. Since the distance between the centers is at least $2R$, the disks are pairwise disjoint. We claim that each disk D_c contains at least one sensor in OPT, which proves the lemma.

Suppose the claim is not true and let c be a center such that OPT has no sensors in D_c. But then, for any $s_i, s_j \in OPT$, the error in observing the center of D_c will be:

$$U(s_i,s_j,c) = \frac{d(s_i,c)d(s_j,c)}{|\sin\angle s_i c s_j|} \geq d(s_i,c)d(s_j,c) \geq R^2 = U^*$$

However, this means that OPT exceeds the error threshold on c. A contradiction! ■

In the second phase, we use the set of centers to determine the placement of sensors.

Algorithm placeSensors(centers C):

```
• for each $c_i \in C$
    – $W_i \leftarrow \{w : d(c_i,w) < 2R, w \in W\}$
    – $r_i = \max_{w \in W_i} d(c_i,w)$
    – Place three sensors on $circle(c_i,r_i)$ at angles 0, $\frac{\pi}{2}$
      and $\pi$.
```

Here $circle(c,r)$ denotes the circle centered at c with radius r.

Clearly, Algorithm placeSensors places at most $3 \cdot |OPT|$ sensors (Lemma 1). All we need to show is that for any point w in the workspace, we can find two sensors s_i and s_j such that $U(w,s_i,s_j)$ is not too far from U^*.

Lemma 2: For each center $c_i \in C$, let W_i be the set of points defined in algorithm placeSensors. Let s_1, s_2, s_3 be the three sensors placed on $circle(c_i,r_i)$ at angles 0, $\frac{\pi}{2}$ and π respectively. For any point $w \in W_i$, either $\frac{\pi}{4} \leq \angle s_1 w s_2 \leq \frac{3\pi}{4}$ or $\frac{\pi}{4} \leq \angle s_2 w s_3 \leq \frac{3\pi}{4}$.

Proof: Since the angle $\angle s_1 c s_2$ is $\frac{\pi}{2}$, any point inside the circle sees s_1s_2 with angle at least $\frac{\pi}{4}$. (Similarly for s_2s_3). Let S_1 be the set of points x inside $circle(c_i,r_i)$ with $\angle s_1 x s_2 > \frac{3\pi}{4}$ and S_2 be the set of points x inside $circle(c_i,r_i)$ with $\angle s_2 x s_3 > \frac{3\pi}{4}$. These two sets are disjoint as shown in Figure 2. ■

Finally, we show that the uncertainty on any point of the workspace is a constant factor away from U^*.

Lemma 3: For any point $w \in W$, $U(w,S) \leq 23U^*$.

Proof: First we observe that the sets W_i defined in placeSensors cover W. Let $w \in W$ be a point that lies in W_i. As shown in the previous lemma, there are two sensors on $circle(c_i,r_i)$ that see this point with an angle between $\frac{\pi}{4}$ and $\frac{\pi}{2}$. Further, the distance between w and these two sensors is at most $4R = 4\sqrt{U^*}$. Hence, the error is at most $\frac{16U^*}{|\sin 45|} \leq 23U^*$. ■

We conclude the section with a remark on the trade-off between the number of sensors and the uncertainty guarantee. Suppose instead of using $3|OPT|$, we were allowed to use $3 \times k \times |OPT|$ sensors. One possible approach is to decrease the threshold U^*. However, in this case, the increase in the number of sensors is a function of the workspace and a more direct estimate would be desirable. Alternatively, one can cover the disks of radius $2R$ (used by placeSensors) by k disks with smaller radius and guarantee a smaller deviation from U^*. This brings up the following disk-covering problem:

Given a disk of radius $2R$, find the smallest radius $r(k) < 1$ required for k equal disks to completely cover the original disk. Clearly, this would guarantee a reduction of $r(k)^2$ in the performance guarantee of our algorithm. The interested reader can find different values of $r(k)$ in [8].

III. Deployment of a mobile sensor-network

Imagine a scenario where a team of n robots is charged with monitoring m possibly mobile targets. Let $X(t) = [x_1(t), \ldots, x_n(t)] \in \mathcal{C}^n$ denote the state of the team at time t where $\mathcal{C}$ denotes the configuration space of a single robot [1]. Let $Y(t) = [y_1(t), \ldots, y_m(t)]$ be the state (typically location) of the targets. As in the previous section, we consider the case where individual robots are incapable of estimating the position of the targets. Instead, a pair of robots are required to obtain a good estimate of a target's position. A typical and common example of this case is robots equipped with cameras where two cameras are needed to estimate the position of a target via triangulation. We will use a generic uncertainty function $U(x_i, x_j, y_k)$ which returns the uncertainty in estimating the position of target k using robots i and j. If the robots are equipped with cameras, this is the uncertainty defined in Section II-A. However, the algorithm presented in this section is rather general and works with other uncertainty functions as well.

More precisely, the estimation process is defined as follows: at any given time t, we are given a communication graph $G(t) = (V(t), E(t))$ whose set of vertices correspond to the robots. There is an edge between two robots if they can communicate at time t. Throughout the paper, we assume that only robots that can directly communicate can estimate the position of a target. We call such pairs of robots *admissible*. This is warranted because communication between other pairs involve multi-hop transfer of possibly large amounts of data (e.g. images) which is not feasible for mobile sensor-networks due to energy and communication constraints.

At any given time, we must solve two sets of problems:

Assignment: which robot pairs are assigned to which targets. As in the previous section, this amounts to computing the function

$assign(y_k(t), X(t)) = \arg\min_{x_i, x_j \in E(t)} U(x_i, x_j, y_k)$

Placement: we must find good locations for the robots so that the overall error in estimating the position of the targets is minimized. In other words, we must find a mapping $x_i(t) \rightarrow x_i(t^+)$ for each robot so that the maximum uncertainty in tracking any target is minimized.

A main challenge here is to solve these two problems simultaneously and in a distributed fashion. It is also required to address the two dynamic aspects of the problem: moving targets as well as the changes in the communication tree in an ad-hoc fashion. In the next section, we present the outline of a discrete-time solution where we treat the communication graph and the targets as static within a time-step. Afterwards, in Section III-C, we show how the optimal moves for each time step can be computed for the case where the communication graph is acyclic, i.e. a tree.

[1]For ease of notation, let us assume that the robots are identical.

A. A discrete-time algorithm

To capture the dynamic nature of the problem, we propose a discrete time algorithm. The discrete time-steps are chosen small enough so that each target can be treated as static and the communication tree is fixed within the time-step.

At the beginning of each time-step the robots compute the following in a distributed fashion:

- Step 1: Assign each target to the best possible admissible pair of robots
- Step 2: Given target-sensor assignments, compute an optimal destination for each robot to minimize the overall error.

Once the computation is performed, each robot moves to its new location.

Since we treat the targets as static within a time-step, for the rest of the paper we drop the dependency on time and denote their state by $Y = [y_1, \ldots, y_m]$. We assume that the robots have a fixed communication graph within a time-step and the graph is a tree. Further, we assume that one of the robots is arbitrarily selected as a leader. This assumption is required to have a well-defined child/parent relationship between the nodes. We root the communication tree at the leader who becomes the parent of its immediate neighbors. These children in turn recursively compute their own children. We can now use the following convention for edges: whenever we refer to edge $(u, v) \in E$, the node v will be the parent of u. Further, for the acyclic graph $G = (V, E)$ and each vertex $u \in V$, the graph $G(u) = (V(u), E(u))$ denotes the subtree rooted at u.

B. Computing the optimal assignment

At the beginning of each time-step, the robots compute an optimal target-robot pair assignment as follows:

Each node computes two tables whose size is equal to the number of targets. At node v, the table $U_v[t_i]$ is equal to the best uncertainty that can be achieved by tracking target t_i using the edges in $G(u)$. The table $A_v[t_i]$ holds the index of the best edge (sensor pair) for target t_i.

The computation starts at the leaves, who pass empty tables to their parents. Let v be an internal node with children $u_1, \ldots, u_k$. Upon receiving the tables from all children, node v computes its tables as follows: for each target t and child u_i, it compares $U(v, u_i, t)$ with $U_{u_j}(t)$ for all children u_j and picks the best edge to track target t.

Once the process propagates up to the root, the best assignment for the current state has been computed. The root passes this table back to its children to notify them of the assignment who then recursively pass it back to their children.

Therefore, at the end of the first phase, all nodes have access to a function $assign(u, v)$ which returns the set of targets assigned to the edge (u, v).

C. Computing the optimal move

In the second phase, the nodes compute an optimal move to relocate for their given target assignments.

Recall that from the previous step, the targets are assigned to pairs of robots (equivalently to the edges of the tree).

We will use the following notation in this section:

The notation x_i denotes the state of the robot i at the beginning of timestep. Let C_i be the set of states (configurations) accessible from x_i within a single time-step. We assume that a discrete representation of C_i is available. Let N_i be the number of samples in this representation and let $N = \max_i N_i$.

For a graph G, the set of all possible configurations of the nodes in G is denoted by $C(G) = \prod_{v \in V(G)} C_v$. Given $X(V(G)) = [x_1, \ldots, x_{|V(G)|}]$, the states of nodes in G, the uncertainty of the configuration X is given by

$U(G, X) = \max_{(u,v) \in E(G)} \max_{t \in assign(u,v)} U(t, x_u, x_v)$. That is, the maximum uncertainty on any target.

The computation starts at the leaf nodes. Starting from the leaves, the nodes will compute a cost table of size N and pass it onto their parents. Each non-leaf node waits until it receives all tables from its children, then updates its own table and passes onto its parent. The process continues until it reaches the root who then computes the optimal placement for itself and its children and informs them about their new location for this time-step. The children do the same recursively.

For each node v, the table $T_v[x]$ will hold the value of the best possible uncertainty by relocating the nodes in $G(v)$ subject to the constraint that v is at the configuration $x \in C_v$.

Let v be an internal node, whose children are $u_1, \ldots, u_k$. The entries on the table T_v is computed as follows.

$$T_v[x] = \max_{u_i} \min_{x' \in C_{u_i}} \max_{t \in assign(u_i,v)} \max\{T_{u_i}[x'], U(x', x, t)\} \tag{2}$$

As in the previous section, this process propagates up to the root; who then computes the best possible location for itself. This location is given by $\arg\min_{x \in C_r} T_r[x]$ where r is the root node. The root then notifies each child u_i that they should move to $\arg\min_{x' \in C_{u_i}} \max_{t \in assign(u_i,v)} \max\{T_{u_i}[x'], U(x', x, t)\}$. The children u_i notify their children recursively.

For a given target-sensor pair assignment, the optimality of this computation follows from the dynamic-programming principle. To see this, inductively assume that the computation is optimal for all trees whose height is at most k. Consider a tree of height $k+1$ rooted at node v. Then the optimality of the move $\arg\min_{x \in C_v} T_v[x]$ for node v follows from the definition of Equation 2 and the inductive hypothesis.

We emphasize that this computation is optimal only for given target assignments. Hence, it may not be the globally optimal move for the end of the time-step (as the best assignments may change after relocation).

D. Simulations

We demonstrate the utility of the deployment algorithm in two simulations.

In the first simulation (Figure 3), the targets (represented by squares) are stationary. The communication between the robots (represented by circles) is restricted to the edges (colored, bold lines). The color of the target matches the color of the assigned edge and the dashed lines connect the targets to their assigned sensors. The numbers on the targets are the estimation uncertainties for the time-step. The title of each subfigure shows the total error within the time-step. Further, since the targets are static and the communication structure does not change, the global computation converges to a local minimum (i.e. the robots do not move any further after a while).

The second simulation is similar to the first one with the exception of stationary targets. As shown in Figure 4, in this simulation the targets move along a line whose direction is chosen randomly. The targets move at a rate slower than the robots and therefore the robots eventually catch up with targets again forming spatially coherent clusters.

IV. Conclusion

In this paper, we studied two novel problems for sensor networks. In the first part of the paper, we studied the problem of placing a minimum number of (stationary) cameras to minimize the error in localizing targets in a given workspace. For this problem, we presented an approximation algorithm and showed that its performance is within a constant factor of the optimal performance.

Next, we studied the deployment problem where a mobile sensor team is charged with tracking possibly mobile targets. We focused on a version where the communication between the nodes is restricted to a tree and formulated a discrete-time version of this problem where the targets and the communication tree is treated as static within a single time-step. We showed how the optimal move can be computed for a given target/sensor-pair assignment and demonstrated the utility of this algorithm in simulations.

The paper raises interesting questions for future research directions: for the placement problem, establishing its complexity and extending the algorithm for other types of sensors are important problems. For the deployment problem, incorporating visibility issues in the target assignment is a significant challenge which we will address in our future work.

Acknowledgment

This work is supported in part by a grant from RPI.

References

[1] T. F. Gonzales, "Clustering to minimize the maximum intercluster distance," *Theoretical Comput. Sci.*, no. 38, pp. 293–306, 1985.

[2] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[3] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[4] J. Spletzer and C. Taylor, "Dynamic sensor planning and control for optimally tracking targets," *International Journal of Robotics Research*, vol. 22, no. 1, pp. 7–20, 2003.

[5] V. Isler and R. Bajcsy, "The sensor selection problem for bounded uncertainty sensing models," *IEEE Tran. Automation Science and Engineering*, 2006, to appear.

[6] S. Aranda, S. Martínez, and F. Bullo, "On optimal sensor placement and motion coordination for target tracking," in *International Conference on Robotics and Automation*, Barcelona, Spain, Apr. 2005.

[7] A. Kelly, "Precision dilution in mobile robot position estimation," in *Intelligent Autonomous Systems*, Amsterdam, Holland, 2003.

[8] E. W. Weisstein, "Disk covering problem," MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/DiskCoveringProblem.html.

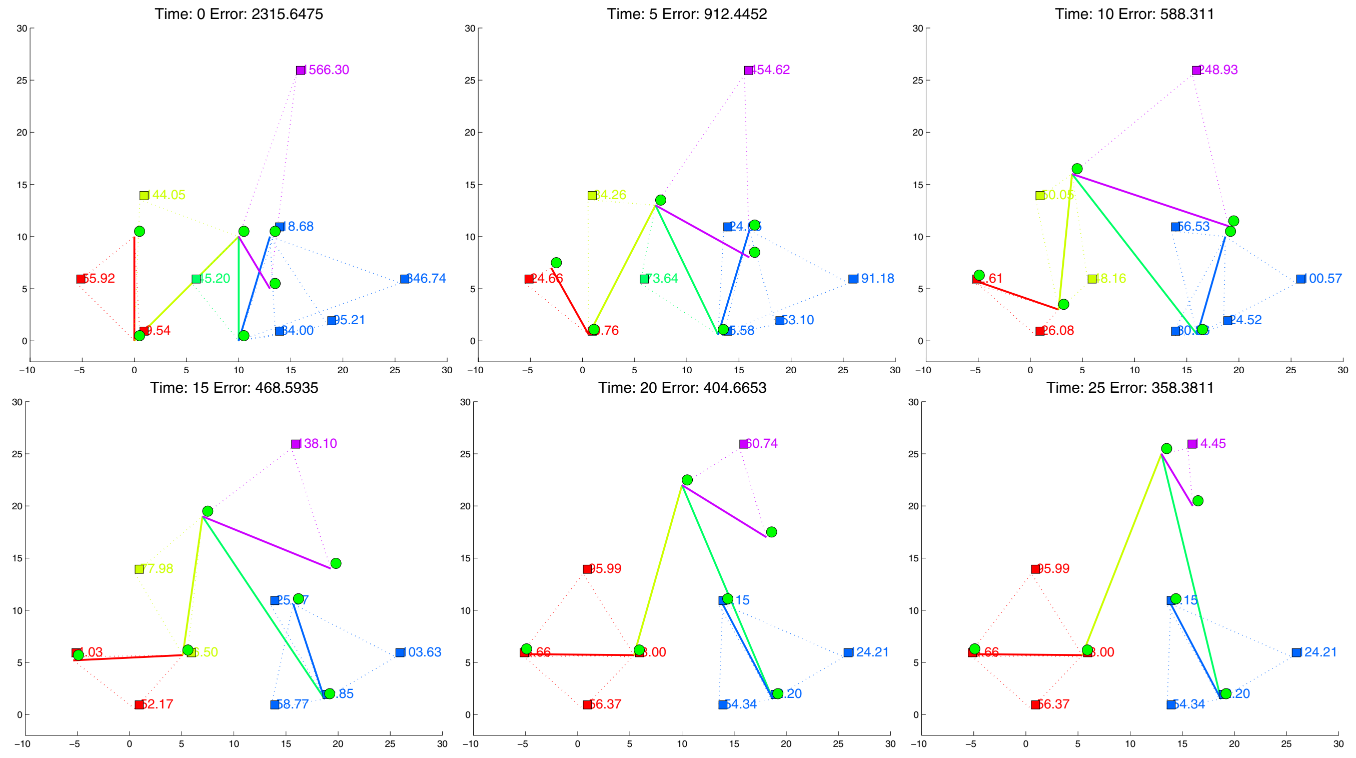

Fig. 3. Snapshots from the simulation of an instance of the deployment problem. The communication between the robots (represented by circles) is restricted to the edges (colored, bold lines). The color of the target matches the color of the assigned edge and the dashed lines connect the targets to their assigned sensors. The numbers on the targets are the estimation uncertainties for the time-step. The title of each subfigure shows the total error within the time-step.

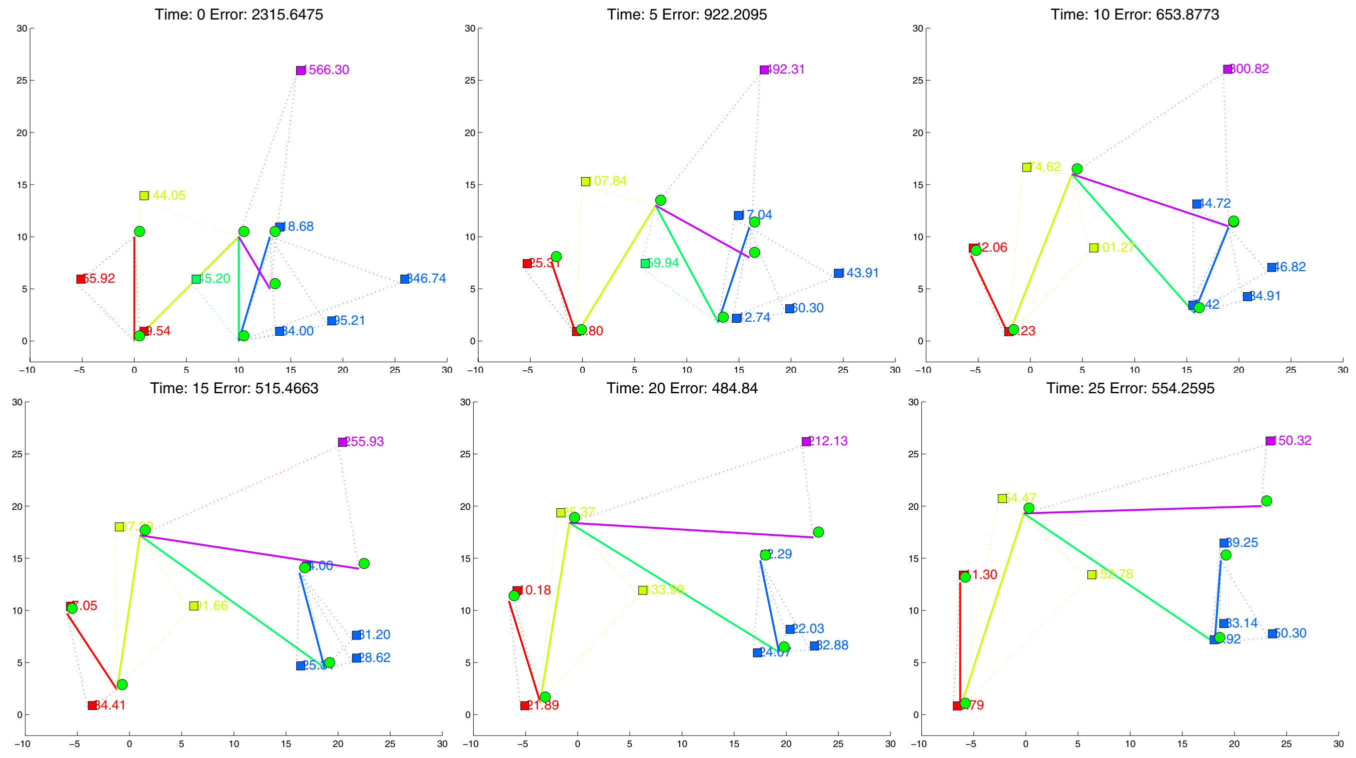

Fig. 4. Tracking targets moving at a constant velocity. See the caption of Figure 3 for details.