Probability Based Optimal Algorithms for Multi-sensor Multi-target Detection

T. R. Rahul
Robotics Dept
International Institute of Information Technology,
Hyderabad – 500032, INDIA
trahul@students.iiit.ac.in

K. Madhava Krishna
Robotics Dept
International Institute of Information Technology,
Hyderabad – 500032, INDIA
mkrishna@iiit.ac.in

Henry Hexmoor
CS Dept
Southern Illionis University,
Carbondale, IL, 62901, USA
hexmoor@uark.edu

Abstract– *The algorithm presented in this paper is designed to be used in automated multi-sensor surveillance systems which require observation of targets in a bounded area to optimize the performance of the system. There have been many approaches which deal with multi-sensor tracking and observation, but there haven't been many which deal purely with targets detections i.e. each target needs to only be detected once. The metric used to gauge the performance of the system is percentage of targets detected among those that enter the area. Targets enter the area through source points on the side of the area according to Poisson distribution, the rate of entry is constant for all sources. The algorithm presented here uses target arrival information, sensor positions to generate an optimal motion strategy for the multi-sensor system every T time-steps i.e. every T time-steps, the probability of finding undetected targets is estimated, the optimal sensor paths for the next T time-steps are calculated. The algorithm performs robustly and optimally detecting around 80% of the targets that enter the area.*

1. Introduction

Many security, surveillance and reconnaissance systems require distributed autonomous observation of movement of targets navigating in a bounded area of interest. This is done using a mobile sensor system in which the sensors either work autonomously or in collaboration with other sensors. Multi sensor surveillance finds applications such as in border patrol, guarding of secured areas, search and rescue and warehouse surveillance. There have been many approaches which deal with target detection and tracking. Some of these approaches are behavior based where apriori knowledge of target behavior are used. There are also many approaches which try to optimize performance by using formations which maximise coverage. In most cases, not much is known about target arrival and in some cases even the dimensions of the surveillance area isn't known.

In this paper, we take leads from Krishna's [2] approach (IROS 05). This algorithm is designed to be used to monitor a large rectangular surveillance area with a limited number of sensors. On the boundary of the surveillance area lie sources from which targets emanate into the surveillance zone. The dimensions of the surveillance area and the poisson rate of entry from the sources are known beforehand. A T-step ahead algorithm is presented, which works by calculating optimal sensor paths every T time-steps for the next T time-steps. This is done by using target arrival information, sensor positions,information of detected targets. This approach will be very useful in guarding large open areas which are crisscrossed by moving targets. Since the number of sensors at our disposal are limited the sensors have to be mobile to try and cover as much of the region as possible.

2. Framework

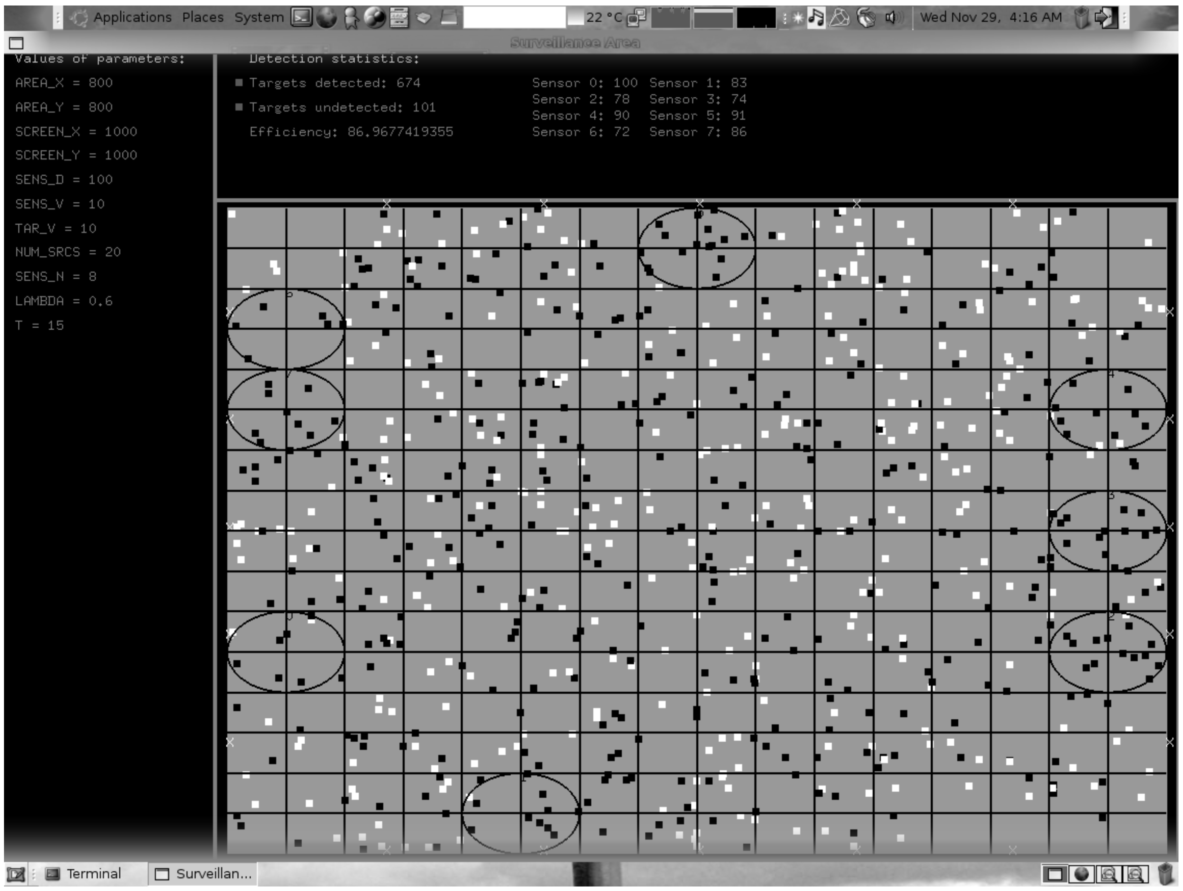
The framework consists of a large rectangular area and a limited number of mobile sensors whose coverage area is much smaller than the total area to be covered. All along the boundary lie discrete points from which targets emanate into the surveillance area. The angle at which the target emanates from the source is a random variable taking values in [0, Π] and the targets enter according to Poisson distribution. The targets move across the area with constant velocity in linear trajectories. We divide our surveillance zone into smaller rectangles and further divide these into smaller cells. This is done to help in computation, to make it less complicated. Also certain assumptions are made in our framework.

- We assume that the sensors have an FOV of 360 degrees and can detect all targets within their FOV.
- All targets move with the same uniform velocity within the surveillance zone along linear trajectories, which can be ascertained by the sensor.
- The takeoff angle for a target from its source is uniformly distributed between [Π/9, 8Π/9]. (We have used these values instead of [0,Π] to see to it that targets don't enter and leave from the corners.)

Apart from these, we have apriori knowledge of the environment and of the statistics of target entry and we use this information to try and come up with an optimal algorithm. This situation is not new in robotic and multi-robotic literature where optimal path planning and scheduling algorithms require prior knowledge of the workspace in which they operate in terms of their static and dynamic contents vis-à-vis behavior based approaches that do not guarantee optimality or completeness but require no prior knowledge.

2.1 Description of Surveillance Zone, Sensors, Targets.

Consider the surveillance system depicted in Figure 1. The sides of the rectangle on the right forms the boundary of the surveillance zone – the area enclosed by it is the area of interest where sensors attempt to optimize their rates of detection. The circles are used to represent the FOVs of the sensors and the radius of the circle is the effective sensor range. The field of vision (FOV) of a



```
Values of parameters:        Detection statistics:
AREA_X = 800                 ■ Targets detected: 674        Sensor 0: 100  Sensor 1: 83
AREA_Y = 800                 ■ Targets undetected: 101      Sensor 2: 78   Sensor 3: 74
SCREEN_X = 1000                                             Sensor 4: 90   Sensor 5: 91
SCREEN_Y = 1000                Efficiency: 86.9677419355    Sensor 6: 72   Sensor 7: 86
SENS_D = 100
SENS_V = 10
TAR_V = 10
NUM_SRCS = 20
SENS_N = 8
LAMBDA = 0.6
T = 15
```

Figure 1 – Screenshot of a simulation in progress.

sensor is 360 degrees. The square dots in the surveillance area are the targets which move across the area. The white dots denote the targets that haven't been detected by a sensor yet, the black dots represent the detected targets.

The entire surveillance region is discretized into a lattice of cells. The points in the figure in the surveillance area at which the lines intersect represent the center of each such cell. At each of the cell locations various aspects of target statistics are computed that are described later. The crosses outside the surveillance zone are the source points from where targets emanate as per Poisson statistics. Targets percolate from each of those sources into that horizontal or vertical half-plane that contains the surveillance zone. Therefore, all targets coming from a particular source will be contained with an angular span of 7π/9 radians.

2.2 Problem Statement

- **S:** The set of all sensors in the system. S ={ s_0, s_1,..., s_{ns}}, where s_i denotes the sensor with label i , ordered in a sequence. Hence ns the label of the last enumerated sensor is also the number of sensors in the system, which is a constant.
- **TT :** The set of all targets that entered the area since the start of the simulation. TT = { tt_0, tt_1,..., tt_{ntt}} and ntt is the number of targets in the system until now.
- **DT :** The set of all detected targets in the system. D = { dt_0, dt_1,..., dt_{nd}} and nd is the number of targets in the system in current time step.
- **F :** The set of variables which represent whether a target has been detected or not. F = { f_0, f_1, . . f_{ntt} }, where f_i is a binary variable which is set to one once the target with id i has been detected.
- Our goal is to maximise:

$$\mathbf{R = 1 / ntt^{*}\sum_{i=1}^{ntt} f_i}$$

2.3 Objectives

Lot of related work has been done on optimal target tracking and observation in the last few decades, but this algorithm focuses on target detection and not tracking i.e. targets can't be detected more than once. The goal is to maximize the percentage of target detections. Our primary objectives are:

- To try and minimize detections of targets by more than one sensor.
- To maximize total percentage of target detections.
- To reduce overlapping of sensor paths.
- To make sure there aren't many overlooked areas.
- To see to it that the search space doesn't become too large.

At every T time-steps in the process, the probability of finding undetected targets at each cell in the surveillance area is estimated for the next T time-steps, where T is a parameter which depends on the area of the surveillance zone. To find the probabilities, a number of factors are considered:

- The target's velocity, source location and the statistics of target entry and exit are used.
- Information of targets that have already been detected is used as these targets need not be detected again.
- The sensor's position information is also used while finding the probability of undetected targets at these areas.

At every time-step in the process, the probability of finding undetected targets at each cell in the surveillance area is estimated for the next T time-steps, where T is a parameter which depends on the area of the surveillance zone and number of sensors. After identifying the area's which have a high probability, an allocation algorithm is used to allocate sensors to these areas, while considering the speed of the sensors and the area to be covered.

3. Approach

3.1 Algorithm A

Once the simulation starts, every T time-steps, the following steps run to estimate the probabilities at each position on the grid for the next T time-steps and generate sensor paths:

1. At each cell, Pi , in the lattice the following are computed.

   - $\lambda_{Pi,t}$ **:** The expected rate of target entry into the FOV of a sensor centered at Pi at time t. If there are Q target sources each emanating targets at rate λ then

$$\lambda_{Pi,t} = \lambda * \sum_{j=1}^{Q} \theta_j /(7\pi/9), \quad \text{if } avgmin_{Pi,j} < t < avgmin_{Pi,j} + avgstay_{Pi,j},$$

where $avgmin_{Pi,Q}$ denotes the average time taken by a target from source Q to enter the FOV of a sensor placed at Pi, $avgstay_{Pi,Q}$ denotes the average time for which a target emanating from Q stays in FOV of a sensor positioned at Pi. The ratio $\theta j /(7\pi/9)$ denotes the expected fraction of the total number of targets that would enter the FOV of the sensor at Pi from the jth of the Q target sources. It is also denoted by

$$\lambda_{Pi,j} = \lambda * \theta_j /(7\pi/9),$$

the rate at which the sensor at Pi sees targets from source j . In other words, among the targets taking off at any angle within [$\pi/9, 8\pi/9$] , from source j only those that takeoff within the angular span [$\delta_j , \delta_j + \theta_j$] would enter the FOV of the sensor, where δ_j is the smallest angle formed by the segment connecting a vertex of the FOV square and source j and $\delta_j + \theta_j$ is the largest angle.

   - $b\lambda_{Pi,j,k,t}$ **:** The expected rate of targets at Pi from source j at time t blocked by the sensor positions T-k time-steps ago i.e. out of the total angle of entry into any position Pi on the grid, a certain amount $b\lambda_{Pi,j,k,t}$ is blocked by sensors, so, the targets which enter from there are already detected before they reach Pi. This is calculated as:

$$b\lambda_{Pi,j,k,t} = \lambda * \sum_{n=1}^{ns} b\theta_{i,j,n,k} /(7\pi/9), \quad \text{if } avgmin_{Pi,j} < t + T < avgmin_{Pi,j} + avgstay_{Pi,j},$$

where $b\theta_{i,j,n,k}$ is the angle blocked out at position Pi by sensor n from source j T-k time-steps ago. An example situation is given in Figure 2 to make things clear.

2. For every sensor sj in the system located at a particular cell location Pj the path it takes for the next T time steps is computed such that the number of target detections is maximized based on steps 3 and 4.

3. Using the values we calculated in step 1, the probability of finding undetected targets in the next T secs is estimated at each position on the grid. Two different methods of generating the probabilities have been implemented and tested. The second method will be described later on. In the first method, this probability is calculated by using the following factors:

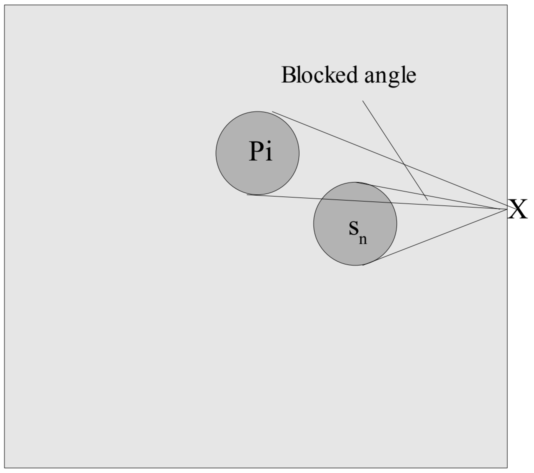


Figure 2 - An example situation illustrating the blocked angle calculation.

- Information about targets already detected.
- Information about target arrival statistics and positions of sources.
- Positioning of sensors in the previous T time-steps in the surveillance zone.

The formula used to calculate this value is as follows:

$$\mathbf{P_{i,t} = \sum_{j=1}^{Q} \left(\lambda_{P\,i,t} - \sum_{k=1}^{T} b\lambda_{i,j,k,t} \right) - det_{t\,i}/avgstay_{Pi,j} \,),}$$

where $det_{t\ i}$ is the number of detected targets which are expected to lie in the FOV of sensor positioned at Pi at time t.

4. Once the probabilities are calculated at each position on the grid, the allocation algorithm takes over which consists of the following steps:

    a) For each sensor, all the positions that can be accessed by the sensor in t (number of time-steps left for sensor) time-steps are found. This value is T for every sensor before it moves and is recalculated based on how much the sensor has moved.

    b) Now, each of these positions are checked for conflict and resolved. Two things are checked for, the distance between chosen positions and distance from a stationary sensor (To see to it that paths don't overlap). This is done using priorities, each sensor is randomly allocated a priority every T time-steps. In the event that two sensors choose conflicting positions, the sensor with the higher priority is allowed to move to the position and the next best position is checked for conflict and resolved.

    c) This goes on until all the sensors are allocated positions. If any of the sensors still has enough time to move to another position, then step 4 runs again for those sensors with the number of time-steps left as t referred to above.

5. Steps 1 through 4 are repeated throughout the simulation every T time-steps.

3.2 Other Methods

Our methodology needed to be compared to an existing algorithm, but since there haven't been many algorithms which deal with target detections and not tracking. The problem which arises is that our algorithm detects a target just once i.e. once a target has been detected by a sensor, it won't be counted as a detection if the target is detected again. Most other algorithms don't take care of this. So, in addition to the method presented above, we have implemented two other methods which we will compare later on with our current method.

3.2.1 Algorithm B – This algorithm is the same as algorithm A except for one difference, which is that the probabilities at each place is the same for all T time-steps. For each point the entry probability and blocked angle are constant for all T time-steps.

3.2.2 Algorithm C – This method is similar to Algorithm B. It differs from Algorithm B in the probability calculation. Here, blocked angle is not considered i.e. the probabilities are calculated using just the poisson entry rate and information of detected targets.

3.3 Code

The code structure is divided into two main modules. The first being the simulation modules and the other, the sensor module. The simulation module is responsible for running the simulation, rendering the area, the targets,calculating all the important statistics etc.. The sensor module on the other hand contains the sensor class, all the methods which the sensors use such as the sensor algorithm, motion strategy, calculating probabilities, sensor movement, rendering etc. By dividing it in such a way, it is very easy to separate the sensor algorithm from the actual simulation thus making it easy to make changes.

The simulation module has all the methods required to run the simulation, it is responsible for creating, managing targets, checking detections and rendering the whole area. It is also responsible for communicating with the sensor objects, signaling them to update their position, algorithm and data. The simulation also has count of targets at each location at each time-step. This can be very useful in generating heuristics by studying the counts and looking for patterns.

The sensor module represents the sensor system, which takes care of the sensor algorithm, movement, rendering etc. Every T time-steps (we define T), the simulation calls the sensorai() function which is responsible for finding out the best positions and allocating the sensors to these positions. The allocate() function then finds the best paths and updates the sensors' path array for the next T time-steps. The sensors move according to their path array every time-step. The sensor algorithm has two main parts, calculating the probabilities, calculating the motion strategy. The probabilities at calculated at points throughout the area and the values are updated in the position array every time-step. The motion strategy is then calculated based on these values once in T time-steps.

4. Simulation/Results

Some of the important parameters used in this simulation:

- AREA_X - Length of surveillance area.
- AREA_Y - Breadth of surveillance area.
- SENS_D = 100 - Diameter of sensor FOV.
- SENS_V = 10 - Sensor velocity.
- TAR_V = 10 - Target velocity.
- NUM_SRCS(Q) - Number of sources.
- LAMBDA(λ) - Poisson rate of entry for each source.
- SENS_N(ns)=(AREA_X+AREA_Y)/(2*SENS_D) - Number of sensors.
- T = 15 - The parameter which decides when the algorithm runs.

The results presented here give the details of the performance of all the methods which we had mentioned earlier under different conditions. The metric used to measure the efficiency is percentage of targets detected among all the targets that entered the area. The whole simulation has been coded using Python and OpenGL. All the simulations were run on an desktop with a 2.0 Ghz AMD Athlon 2400+ processor and 512 MB RAM.

Area	*LAMBDA(λ)*	*Number of Sources*	*Efficiency(A)*	*Efficiency(B)*	*Efficiency(C)*
500x500	0.3	16	80.31	79.72	76.44
500x500	0.3	20	79.77	79.43	74.14
500x500	0.6	16	79.92	79.39	76.29
500x500	0.6	20	79.61	78.76	73.98
600x600	0.3	16	80.03	79.07	75.61
600x600	0.3	20	79.42	78.61	73.86
600x600	0.6	16	79.81	78.84	74.77
600x600	0.6	20	79.48	78.22	73.18
700x700	0.3	16	79.91	78.66	76.15
700x700	0.3	20	79.14	78.32	75.62
700x700	0.6	16	79.58	78.24	75.71
700x700	0.6	20	78.90	77.86	74.09
800x800	0.3	16	80.41	80.03	75.46
800x800	0.3	20	80.07	79.81	73.99
800x800	0.6	16	79.88	79.11	75.11
800x800	0.6	20	79.34	78.60	73.67

Table 1. Results of simulations of all three methods

From the results it can been seen that algorithm A performs consistently under various circumstances. It can also be observed that the algorithm is robust and can scale well with rate of entry, number of sources and size of surveillance area. Algorithm B is also efficient but it doesn't perform as consistently. But it scales pretty well with number of sources. Algorithm C doesn't perform as well as the other two and it has problems dealing with high number of sources. Also, the performance is not very consistent as the efficiencies vary a lot.

5. Conclusions

A method to optimize the performance of a multi-sensor based surveillance system is presented. The algorithm makes use of a-priori known target arrival statistics and sensor positioning to compute an optimal motion strategy every T time-steps. The algorithm tries to minimise the overlapping of sensor paths and provide maximum coverage. The algorithm presented is very robust and performs consistently in different situations. It detects approximately 4 of every 5 targets that enter the area. The algorithm can be improved further by reducing the complexity of blocked angle calculation by using a more general calculation.

6. References

[1] K Madhava Krishna and Henry Hexmoor, A framework for measuring tracking performance of a sensor network, *Integrated Computer Aided Engineering Journal,* 2005.

[2] K. Madhava Krishna and Henry Hexmoor, A T-step Ahead Constrained Optimal Target Detection Algorithm for a Multi Sensor Surveillance System, submitted to *IEEE/RSJ IROS* 2005.

[3] Yi Zou and Krishnendu Chakrabarty, Sensor Deployment and Target Localization Based on Virtual Forces, *The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, 2003.