Sensor Optimization in a Virtual Environment

Tolga Can
Veysi Isler
Department of Computer Engineering
Middle East Technical University
06531 Ankara TURKEY
+90-312-210-5597, +90-312-210-5579
tcan@ceng.metu.edu.tr, isler@ceng.metu.edu.tr

Maj. Ziya Ipekkan
General Plans and Policy Division
Technology, Defense Research and Armament Department
06100 Bakanlıklar Ankara TURKEY
+90-312-402-1387
zipekkan@tsk.mil.tr

Keywords:
sensors, coverage, surveillance, virtual environment, optimization, genetic algorithms

ABSTRACT: *An algorithm for locating different types of sensors in a 3D virtual environment to achieve a desired coverage of that environment while satisfying some constraints is being developed. In this context, the keyword 'sensor' is used to identify a FLIR (Forward Looking Infra-Red) camera. It can be extended to include Day-TV cameras, night vision goggles and radars. In this problem, the sensor system is composed of two major types of sensor platforms, namely moving and static platforms. The main goal of this system is to maintain security of the overall or indicated regions of environment. While achieving this goal, the system should also satisfy constraints imposed on the system. Maintaining stealth (remaining hidden), constraining route and placement locations are examples of these constraints. As a result of the optimization of the sensor system, the optimal routes and viewing directions for the moving sensors, the optimal locations and viewing directions for the static sensors will be found.*

1. Introduction

Sensor optimization has been previously studied in different research areas. A similar problem called sensor planning has been studied by robotics and computer vision communities, to locate the sensors on a robot to enhance its navigation capabilities in a closed environment [1]. The environments used in these researches are mostly small, closed environments, like offices, laboratories, etc. The goal behind these optimizations is to collect as much information as possible about the surrounding environment. Thus working in an open, big virtual environment has more difficulties involved than working in a small environment [2].

Little research has been done considering the problem of optimally locating sensors in a wide and open synthetic environment. In the paper by Cook, Gmytrasiewicz, and Holder [3] a solution for the mission planning of unmanned ground vehicles (UGV) is discussed. The underlying goals in their study are similar to the ones discussed here: optimally covering the area around a unit of unmanned ground vehicles to maintain security, while remaining hidden from possible enemies.

The rest of this paper is organized as follows. In Section 2.1, the formulation of the optimization problem will be presented. In Section 2.2 and Section 2.3 the methods used for analyzing the coverage and stealth of the sensor platforms will be given. In Section 3, possible solutions for the problem will be discussed. In Section 4, the main steps of a genetic algorithm will be explained. Implementation details will be given in Section 5, and finally the paper will be concluded in Section 6.

2. Optimization

The optimal attribute sets of a set of sensors, *S*, located over a region *A,* is to be found. To be able to judge if a set of attributes are optimal or not, we have to define the goals of the optimization. We also have to define methods to find out how well the system of sensors satisfies these goals.

2.1 Optimization Goals

Establishing a desired coverage of the area of interest and remaining hidden from possible enemies are the two main goals of the sensor system.

To compute the coverage areas of the sensors, the area A, which is a 3D polygonal region, is partitioned into a set of sub-regions. These sub-regions are associated with importance values, which are used to give priorities to regions in the coverage process. The sub-regions of the area *A* have the following properties:

$\bigcup a_i = A$ and $\bigcap a_i = \varnothing$ ($\mathbf{0 \le i \le n}$, n: # of sub-regions).

The area importance function can be defined from the *set of sub-regions* to *real numbers* ($A_{sub} \to R$):

imp: $A_{sub} \to R$, imp(a_i)=c_i area(a_i)

c_i is the area importance constant entered by the user, and area(a_i) is the area of the polygonal *sub*-region a_i.

The *time* dimension should also be considered in the optimization process because of the moving platforms in the sensor set, *S*. Thus, the coverage performance function of a sensor can be defined from *set of sub-regions × set of sensors × time* to *real numbers* ($A_{sub} \times S \times T$):

cover: $A_{sub} \times S \times T$,
cover(a_i,s_j,t)=the area covered over a_i by sensor s_j at time *t*.

The coverage function is computed using computer graphics techniques. The area importance function is used in the computation of the coverage function.

The stealth function of a sensor can be defined similarly from *set of sub-regions × set of sensors × time* to *real numbers* ($A_{sub} \times S \times T$) as:

stealth: $A_{sub} \times S \times T$,
stealth(a_i,s_j,t)=the part of sensor platform a_i seen from the sub-region s_j at time *t*.

The stealth function is computed using line of sight calculations. A probability distribution function over area *A* is used to place enemies on the sub-regions and calculate the line of sights between enemy forces and the sensor platforms. The probability distribution function is defined again by the user.

Considering the time dimension, the function to be optimized in the time interval $T=t_1,t_2$ is the summation:

$$\Sigma\ w_1 \text{ cover}(a_i,s_j,t) + w_2 \text{ stealth}(a_i,s_j,t)$$

where,

$(0 \le i \le \ n,\ \ n : \#\text{ of sub-regions})$
$(0 \le j \le \ m,\ m : \#\text{ of sensors in the system})$
$(t_1 \le t \le \ t_2)$

In this summation the *cover()* and *stealth()* are computed in discrete time intervals, *Δt*.

2.2 Coverage Analysis

The coverage analysis function is used to compute the part of the region *A* that is seen (covered) by the sensors in the sensor set *S*. A quantitative and a visual result of this computation are provided. The quantitative result is used by the optimization algorithm, while the visual result is used for evaluation and verification of the optimization results.

A similar study by Silicon Graphics for visualizing the terrain parts seen by an unmanned air vehicle (UAV) has been done [4]. The projective texturing and shadow testing method presented in that study are used to find the coverage area of a sensor.

The 3D terrain model and the sensor information, such as the location, viewing direction, viewing range, and the field of view angle is used to compute the coverage area. This method uses the rendered sensor view. By defining it as a texture, it is mapped back on to the terrain after automatically generating the texture coordinates using backward projection.

The boundaries of the region, which the sensor senses is found with the projective texturing method. But using projective texturing alone cannot identify the regions, which are occluded by possible terrain features such as hills and valleys. Depth testing is used to identify those regions, which are actually not seen by the sensor. In depth testing the depth information in the rendered sensor view is compared with the computed depth coordinate, *r*, which is found after backward projecting that view as a texture image. If the computed *r* value is not less than or equal to the real depth value in the sensor view, then that part of the terrain is actually not seen by the sensor, and thus the texture is not mapped on it. The depth testing mechanism, which is provided in hardware by high end Silicon Graphics Workstations, is illustrated below in Figure 2.2.1.

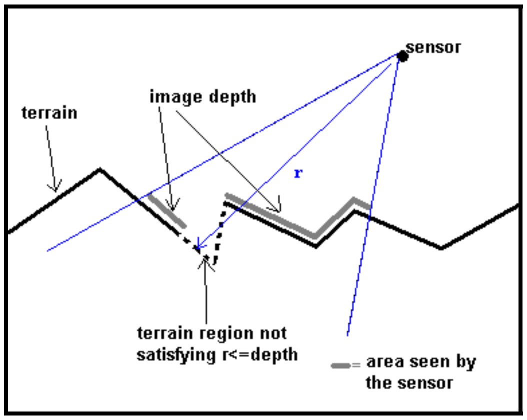
sensor
terrain
image depth
r
terrain region not satisfying r<=depth
▬ = area seen by the sensor

Figure 2.2.1 The Depth Test

The quality of the simulated sensor view depends on the distance, and the viewing angle between the sensor and the target. The idea behind the lighting calculations in computer graphics is very similar to this interaction between the sensor and the terrain. So, the quality of the data is simulated by putting a spot light source on the sensor platform pointing in the sensor viewing direction, which is shown in Figure 2.2.2.

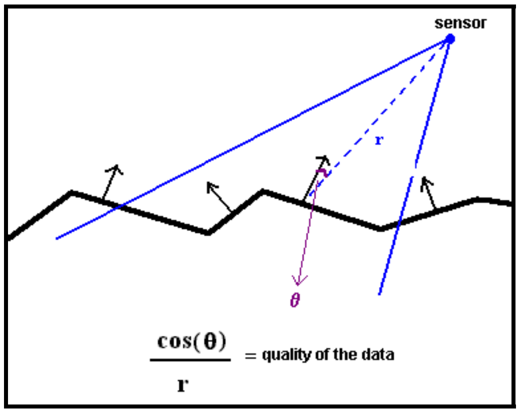

Figure 2.2.2 Simulation of the Data Quality

The result of the coverage computations can be presented on the 3D model of the terrain from any view location and direction. An orthographic view of the whole area is processed and a quantitative result is produced by processing that view. The visual result of the coverage analysis, and the sensor view is shown in Figure 2.2.3. The sensor view, which is shown on the lower right corner of Figure 2.2.3, is used to obtain the real depth information, which is used in the depth testing process. For the coverage analysis calculations the sensor view, with the sensor properties such as location and viewing direction of the sensor, are used.
In the visual result of the coverage analysis, the parts seen by the sensor are shown shaded with a gray tone. The black parts in the image are the regions which are inside the boundaries of the coverage area but not seen by the sensor.

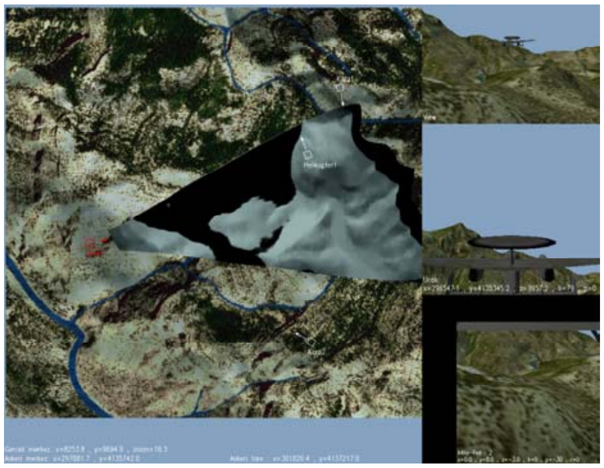

Figure 2.2.3 A Snapshot from the Coverage Analysis

For the quantitative results of the coverage analysis, the visual coverage results of the sensor platforms in the system are merged in the image space. The regions covered by sensors are identified after processing the resulting image, and a matrix data is produced. The matrix data has the same dimensions with the processed image, i.e. the number of cells in the matrix data is equal to the number of pixels of the image. The binary information in this coverage matrix, C_{matrix}, indicates whether a cell is covered by a sensor or not. The terrain importance values are converted to a matrix data of the same size, Imp_{matrix}. The coverage performance of a sensor system at time *t* is computed as shown:

$$\Sigma \text{ cover}(a_i, s_j, t) = \Sigma_x \Sigma_y \, Imp_{matrix,x,y} \, C_{matrix,x,y}$$

($0 \le i \le n$, n : # of sub-regions)
($0 \le j \le m$, m : # of sensors in the system)
($0 \le x \le X$, X : horizontal size of Imp_{matrix} and C_{matrix})
($0 \le y \le Y$, Y : vertical size of Imp_{matrix} and C_{matrix})

2.3 Maintaining Stealth

To evaluate the sensor system considering the stealth performance, the probability distribution function, which indicates the possible enemy locations, is used. A number of enemy observers are scattered on the terrain according to the probability distribution function specified by the user. After the line of sight calculations carried out between the enemy observers and sensor platforms, the numbers of sensors seen by the enemy observers are found out. This process is repeated several times and the average of the results of these line of sight calculations is used as the measure of maintaining stealth. Thus the stealth function used in the optimization algorithm is computed at time *t* by:

$$\Sigma \text{stealth}(a_i, s_j, t) = \frac{\Sigma_n \text{ \# of sensors seen by the enemy}}{n}$$

n : # of stealth performance finding processes.
($0 \le i \le$ n, n : # of sub-regions)
($0 \le j \le$ m, m : # of sensors in the system)

3. Possible Solutions to the Optimization Problem

The sensor optimization problem is a multi-objective optimization problem. As a result of the optimization, the attribute set of the sensors in the sensor system is found. The attributes in this set are the routes, coordinates and viewing directions of the sensors. Defining a mathematical objective function composed of these attributes is impossible, because the objective function is also dependent on the terrain that the sensor system is placed on. Thus optimization methods that depend on calculus and enumeration cannot be used in this sensor optimization problem.

Random search methods, which try to find the best solution by searching the solution space randomly, are appropriate for the sensor optimization problem. Some heuristics can be incorporated into the search method to decrease the convergence time.

Among the random search methods the genetic algorithm method is chosen for the sensor optimization problem described here.

4. The Genetic Algorithm

Genetic algorithm is based on the evolution concept that is present in the nature. It is a blind search method in the sense that the algorithm does not know what it is optimizing. To find an optimal solution in the solution space, it uses a *fitness* function to compare the goodness of the solutions. *Reproduction* and *mutation* operators are used to produce new solutions. The advantage of a genetic algorithm is that it deals with a group of solutions, so that it is avoided to follow a single path, which may not lead to a solution. The steps of a genetic algorithm can be listed as follows:

i. *An initial set of possible solutions is created randomly. This set forms the population and the coded solutions form the chromosomes.*
ii. *The fitness function is used to find how fit each chromosome is.*
iii. *Chromosome pairs are chosen for reproduction and mutation. A new population is created.*
iv. *Chromosomes in the old population, which are not fit enough, die and they are replaced by the fitter new generation chromosomes.*
v. *If the new generation is fit enough to be an optimal solution or enough number of generations have passed, then the solution is the best chromosome in the current population. Otherwise the steps from iii are repeated.*

The new solution found after mating the selected two chromosomes is generated by crossing-over the genes between the selected chromosomes. This *crossover* operation aims to combine to promising properties of the parent chromosomes to produce a fitter child. The *mutation* operator randomly changes the genes of a chromosome with a probability of mutation. It aims to cause jumps to new solution groups in the solution space.

The genetic algorithm may be unable to find the optimum solution of a problem but it guarantees to find an acceptable optimal solution in a reasonable time. The convergence time of a genetic algorithm is greatly affected by the factors such as the population size, the mutation probability, and the number of chromosomes die in each generation. A genetic algorithm can be fine-tuned by finding the best values for these parameters in a specific problem through experiencing.

5. Implementation

A genetic algorithm is implemented for optimization of the parameters of a sensor system in a virtual environment. The coverage and stealth performance functions are the critical functions in the algorithm, which are used to find how fit a solution is. The solution set is composed variable sensor parameters such as the route, location, and viewing direction.

These parameters are coded into a chromosome structure to be used in the genetic algorithm.

The sensor platforms in this problem are ground-based platforms; that is the moving platforms follow the surface of the terrain while static platforms are placed on a fixed location on the terrain. In the optimization process, which is carried out for a time interval *T*, the moving platforms are considered as static platforms at the time instant *t*. The only varying property of a moving sensor platform, in a time interval *T,* is assumed to be its position. The viewing direction of the sensor on the platform is assumed to be constant in the interval *T*.

A sample solution set, which is composed of the parameters of the sensor system of 1 moving and 2 static sensors, are coded as a *chromosome* and is shown below in Table 5.1.1:

Sensor	x	y	h	p
1. Static	20	45	227	5
2. Static	13	4	97	9

Sensor	i_x	i_y	α	h	p
1.Moving	10	60	20	47	10

Table 5.1.1 A Sample Solution Set

Here, the *x, y,* i_x, i_y values are in meters (coordinates on the terrain) and the *r, h, p* values are in degrees. The *x, y* coordinates for a static sensor indicate its position on the terrain. The *z* coordinate of the platform is the determined by the terrain height. For the moving platforms i_x, i_y coordinates indicate the initial position of the platform. Again the i_x coordinate is determined by the terrain height value. The *h* and *p* are the heading and pitch values of the sensor. The rolling value of the sensors is assumed to be zero. The α value of the moving sensor indicates the moving direction.

The initial population for the genetic algorithm is formed by generating random solution sets as in Table 5.1.1. The method of finding the fitness of a particular solution is explained in the following pseudo-code:

```
fitness = 0;
w1 = weight_of_coverage_performance;
w2 = weight_of_stealth_performance;
for t=t1 to t2 do step $\Delta t$
begin
    compute_new_coords_for_moving_platforms();
    update_coords_of_moving_platforms();
    fitness=fitness+(w1*cover_per()+w2*stealth());
end
```

The time interval *T=[t1,t2]* is the same for all the solution sets, so a normalization on the computed fitness values is not needed. The genetic algorithm uses these fitness values and tries to find the best solution with the best fitness value.

6. Conclusion

In this study a sensor system, which is composed of moving and static sensors, are optimized using a genetic algorithm. The objective of this optimization is to maintain the security of a specific area of interest. An optimal set of parameters of the sensors in the sensor system is found as a result of the optimization.

The genetic algorithm was able to find satisfactory solutions when tested on a 4x4 km. area. The area regions having high importance values were covered by more than one sensor most of the time, but no solution set was able to cover the whole region. The optimization times changed from 1 hour to 3 hour to find a good solution. To decrease the optimization time, some heuristics can be incorporated into the genetic algorithm.

7. References

[1] Briggs, Amy J., Donald, Donald, Bruce R.: “Robust Geometric Algorithms for Sensor Planning”, Proceedings of the Second International Workshop on Algorithmic Foundation of Robotics, Toulouse, France, 1996.

[2] Stamos, Ioannis, Allen, Peter K.: “Interactive Sensor Planning”, IEE Computer Society Conference on Computer Vision and Pattern Recognition, 1998.

[3] Cook, Diane J., Gmytrasiewics, Piotr, Holder, Lawrence B.: “Decision-Theoretic Cooperative Sensor Planning”, IEEE Transaction on Pattern Analysis and Machine Intelligence 18 (18), 1996.

[4] UAV Technical Paper, Silicon Graphics http://www.sgi.com/software/performer/brew/uav.html

Author Biographies

TOLGA CAN is research assistant in Department of Computer Engineering, Middle East Technical University. He received his B.Sc. from the same department in 1998. He is working in the Virtual Environments Group in Modeling and Simulation Laboratory as part of his master thesis.

VEYSI ISLER is a faculty member of the Department of Computer Engineering, Middle East Technical University (METU). He received his B.Sc. degree in Computer Engineering from the same university, in 1987. He worked as a research assistant and instructor for the Department of Computer Engineering and Information Sciences, at Bilkent University where he received his M.S. and Ph.D. degrees between 1987 and 1995. He is the coordinator of Virtual Environments Group in Modeling and Simulation Laboratory of METU.

ZIYA IPEKKAN received M.S. degree in OR from Naval Postgraduate School, Monterey, USA, in 1989. He initiated development of several models, approaches and solutions to assessment and evaluation of force structures. He is currently responsible for Modeling and Simulation activities within Turkish Armed Forces.