Sensor Control for Temporal Coverage Optimization

Vahab Akbarzadeh, Christian Gagné, and Marc Parizeau
Laboratoire de vision et systèmes numériques
Département de génie électrique et de génie informatique
Université Laval, Québec (Quebec), Canada G1V 0A6
Emails: vahab.akbarzadeh.1@ulaval.ca, christian.gagne@gel.ulaval.ca, marc.parizeau@gel.ulaval.ca

***Abstract*—This paper proposes a new sensor control algorithm to adapt the operation parameters of the sensors in order to optimize temporal coverage. Traditional sensor control algorithms rely on non-probabilistic sensor coverage models and simple target trajectory prediction methods, while the new algorithm overcomes all these limitations. The proposed approach make an optimization of one sensor at the time, processed in a random order. The performance of the proposed algorithm is compared with a state-of-the-art general purpose optimization method (i.e., CMA-ES). Results show that our algorithm produces the results in a much shorter time in all the cases considered, while the final coverage of the network was superior in a smaller map and competitive in larger maps. Therefore, the new algorithm can be used in scenarios where response within some time limit is of great importance (e.g., real-time control of the sensors).**

I. Introduction

The aim of temporal coverage optimization in Sensor Networks (SNs) is to derive a mechanism which effectively and efficiently uses the sensing capabilities of sensors during a period of time in order to maximize the coverage of the network over a phenomenon, which itself can be changing over time (e.g., targets moving in a field).

Temporal coverage optimization consists itself of two main problems: predicting trajectories of the targets and controlling the sensors. For trajectory prediction, the goal is to predict the future location of moving targets. In this paper, we rather focus on sensor control, which aims at adjusting the parameters of the sensors (e.g., pan and tilt angles, zoom factor) based on trajectory prediction, so that the final network is able to properly cover the targets as they move within the environment. Coverage level over the targets and computation time of the algorithm are the two main performance criteria to consider for designing a sensor control algorithm.

Sensor control is closely related to the placement problem in SNs when a weighted coverage objective is used (as described in [1]). However, there are two main differences between the placement and sensor control problems: the time requirements and the capacity for the sensors to change over time (or not). As for the time requirement, the running time of the algorithm can be arbitrarily long for the placement problem as it is a one-time process done offline. In other words, once the best location and direction for sensors are determined there is no need to run the algorithm again, therefore the running time of the algorithm is not a critical issue. On the other hand, for the sensor control problem, the frequency of the runs is much higher and the system is operating online. Indeed, the sensor control algorithm should return the results before the next time step, otherwise the results are of little use. This means that the computation time is an important issue for sensor control done in real-time.

As for the static/dynamic nature of the problem, that comes from the fact the we only need to find the best location and direction for the sensors once for the placement problem, in order to achieve maximum coverage over the environment with the solution retained. For the sensor control problem, the location of the sensors are fixed, but their directions are variable. Therefore, sensor control can be viewed as spatial coverage optimization problem for sensors having fixed locations but constantly adapting their configurations over a weighted environment that is also dynamic.

This sensor control problem was discussed in the literature by different authors. For example, Cai et al. [2] suggested the application of the cover set algorithm to the spatial optimization of the directional sensors with fixed locations. At each iteration of a greedy method they designed for the problem, the target location that can be covered by the minimum number of sensor directions is selected and added to the solution set. Authors also proposed a distributed version of the algorithm for large-scale applications. In the distributed approach each sensor has only access to coverage information of its neighbours. A drawback of the proposed method proposed is that it does not work with multiple probabilistic predictions for location of the targets, while in many applications, the trajectory prediction produces different possibilities for the future location of a target.

Similarly, Ai et al. [3] proposed the maximum coverage by the minimum sensors method which follows the same principles as the one proposed in [2]. The proposed a centralized greedy algorithm that iteratively selects a sensor and calculates the number of targets that could change state from uncovered to covered using the sensor. Next, the direction which maximizes the coverage over uncovered targets is selected. The iteration continues until all of the targets are covered or all of the sensors have been selected. The main problem with the proposed approach is that the coverage of sensors over the targets is binary, therefore, the algorithm is not applicable if the sensor has partial coverage over the target.

In [4], Chen et al. defined a weight for each of the targets and each direction the sensor can take, which are called the target weight and orientation weight respectively. The target weight is calculated based on the number of sensors that can cover a target. Orientation weights are defined based on the total number of targets in the environment, the number of targets the orientation can cover, and the target weight for each of the targets in the coverage range. Wang et al. [5] proposed a priority based target coverage algorithm, where the importance of targets is different for the system. A genetic algorithm was applied on the direction of sensors to find the direction which best covers all of the targets. The main drawback of both algorithms is that they do not take into consideration any limitation on the movement speed of the sensors. In other words, they assume that sensors move from any direction to another direction in one time step. The other drawback of the proposed methods is that they only consider one variable parameter per sensor, and therefore iterate over all possible directions of the sensors. This approach is not reasonable when a sensor has two or three variable parameters, as the computation needed to iterate over all possible combinations becomes cumbersome.

In this paper, we introduce the Greedy Sensor-wise Coverage Optimization (GSCO) algorithm. The algorithm is specifically designed to find the best direction for sensors, to cover prediction of targets' location in the environment. The input to the algorithm handles different degrees of certainty for target locations, works with probabilistic coverage model of sensors, and finally accepts limitations speed of sensors movements. Besides, the computational need of the algorithm is very reasonable.

In the following, we first present more formally the sensor control problem we are tackling in Sec. II, along with some details on the specific sensor model used. Then, in Sec. III, we are presenting the GSCO method proposed to achieve effective and efficient sensor control. The paper concludes with a presentation of the experimental methodology and simulation results obtained with this method in Sec. IV, showing its good performance in comparison with an optimization conducted with a black-box method (namely CMA-ES).

II. Sensor Control Problem

In temporal optimization, the goal is to cover a number of targets moving in the environment using a SN. In this paper, we focus on Pan-Tilt-Zoom cameras (PTZ) as our sensors. The pan and tilt angles, and the zoom factor are variable through time, and this provides the capability for the sensors to change their coverage direction and to provide a maximal coverage over a set of moving targets. To provide this coverage, first the targets should be detected in the environment, their trajectories should be modelled and finally the parameters of the sensors should be modified so that the targets remain under coverage while they are present in the environment. We briefly explain the trajectory prediction problem in section II-C. The focus of our paper is on adapting the sensor parameters to the environment according to the task to achieve (i.e., sensor control).

More precisely, assume that we have a sensor network $N = \{\mathbf{s}_i | i = 1, \ldots, n\}$ consisting of n sensors (cameras in this paper) $\mathbf{s}_i$ with the pan, tilt, and zoom control parameters. Each sensor is attributed with a time parameters $t = \{1, 2, \ldots, M\}$, where M is the total simulation time. In this setting, the state of each sensor at time t is defined as $\mathbf{s}_i^{(t)} = (\mathbf{p}_i, \theta_i^{(t)}, \xi_i^{(t)}, f_i^{(t)})$, where $\mathbf{p}_i = (x_i, y_i)$ is the position of the sensor in the coordinate system, $\theta_i^{(t)}$ is the pan angle, $\xi_i^{(t)}$ is the tilt angle, and $f_i^{(t)}$ is the focal length of the sensor. It is also assumed that the sensors are positioned τ meters above the ground. The environment Ξ is defined by a Digital Elevation Model (DEM), where the elevation of each location $(x, y) \in \Xi$ is given as the function $k(x, y)$, and where the number of discrete locations in this model is denoted by $|\Xi|$. Therefore, the elevation of each sensor is $z_i = k(x_i, y_i) + \tau$. As it can be seen, the location of the sensor is fixed through time ($\mathbf{p}_i$), but the pan, tilt, and zoom parameters are variable ($\theta_i^{(t)}, \xi_i^{(t)}, f_i^{(t)}$).

The goal of the network N is to optimally cover J moving targets $\mathbb{T} = \{\mathbf{T}_j | j = 1, 2, \ldots, J\}$. Each target $\mathbf{T}_j$ is first detected at time u_j in the environment, then it makes n_j displacements, until it exits the environment at time $u_j + n_j$. Therefore the trajectory of each target can be represented as the sequence $[\mathbf{T}_j^{(u_j)}, \ldots, \mathbf{T}_j^{(u_j+n_j)}]$.

In the sensor control problem, the goal is thus to find the configuration for the variable parameters of the sensors (e.g., pan, tilt, and zoom), to have maximal coverage over the moving targets in the environment. The commands that the network applies to sensor $\mathbf{s}_i$ through time is defined by $\mathbf{\Pi_i} = \{\pi_i^{(t)} | t = 1, \ldots, M\}$, where $\pi_i^{(t)} = (\Delta_\theta, \Delta_\xi, \Delta_f)$ is the command given to sensor $\mathbf{s}_i$ at time t for updating its pan, tilt, and zoom values. The set of commands for the n sensors is summarized as $\mathbf{\Pi} = \{\mathbf{\Pi}_i | i = 1, \ldots, n\}$. Each sensor also has a set of characteristic parameters which define the maximum and minimum value of each control parameters, along with their associated speed of change:

$$\langle(\theta_{min}, \theta_{max}, v_\theta), (\xi_{min}, \xi_{max}, v_\xi), (f_{min}, f_{max}, v_f)\rangle. \quad (1)$$

It is clear that the given command at each time step should satisfy each sensor's movement constraints. More precisely we have as constraints:

$$|\Delta_\theta| \leq v_\theta, \quad \theta_{min} \leq \underbrace{\theta_i^{(t)} + \Delta_\theta}_{\theta_i^{(t+1)}} \leq \theta_{max}, \quad (2)$$

$$|\Delta_\xi| \leq v_\xi, \quad \xi_{min} \leq \underbrace{\xi_i^{(t)} + \Delta_\xi}_{\xi_i^{(t+1)}} \leq \xi_{max}, \quad (3)$$

$$|\Delta_f| \leq v_f, \quad f_{min} \leq \underbrace{f_i^{(t)} + \Delta_f}_{f_i^{(t+1)}} \leq f_{max}. \quad (4)$$

where the boundaries for each sensor was defined in Eq. 1.

A. Global Coverage Optimization

Now the coverage $C(\mathbf{s}_i^{(t)}, \mathbf{q})$ of sensor $\mathbf{s}_i$ on point $\mathbf{q} = (x_\mathbf{q}, y_\mathbf{q})$ in the environment at time t is defined as a function of distance $d(\mathbf{s}_i^{(t)}, \mathbf{q}) = \|\mathbf{p}_i - \mathbf{q}\|$, pan angle $\theta(\mathbf{s}_i^{(t)}, \mathbf{q}) =$

$\angle_\theta(\mathbf{q}-\mathbf{p}_i)-\theta_i^{(t)}$, tilt angle $\xi(\mathbf{s}_i^{(t)},\mathbf{q})=\angle_\xi(\mathbf{q}-\mathbf{p}_i)-\xi_i^{(t)}$, and visibility $v(\mathbf{s}_i,\mathbf{q})$ of the sensor:

$$C(\mathbf{s}_i^{(t)},\mathbf{q}) = f[\mu_d(\|\mathbf{p}_i-\mathbf{q}\|),\mu_\theta(\angle_\theta(\mathbf{q}-\mathbf{p}_i)-\theta_i^{(t)}), \mu_\xi(\angle_\xi(\mathbf{q}-\mathbf{p}_i)-\xi_i^{(t)}),v(\mathbf{p}_i,\mathbf{q})], \tag{5}$$

where $\angle_\theta(\mathbf{q}-\mathbf{p}_i)=\arctan(y_\mathbf{q}-y_{\mathbf{p}_i},x_\mathbf{q}-x_{\mathbf{p}_i})$ is the pan angle between sensor $\mathbf{s}_i$ and point $\mathbf{q}$, and $\angle_\xi(\mathbf{q}-\mathbf{p}_i) = \arctan(z_\mathbf{q}-z_{\mathbf{p}_i},\|\mathbf{p}_i-\mathbf{q}\|)$ is the tilt angle between sensor $\mathbf{s}_i$ and point $\mathbf{q}$. In other words, for $\mathbf{q}$ to be covered by sensor $\mathbf{s}_i$, we need to take into account its range, viewing angles, and visibility. The focal point $f_i^{(t)}$ indirectly affects the coverage through pan and tilt angles, and range. This effect will be later discussed in Sec. II-B. Let $\mu_d,\mu_\theta,\mu_\xi\in[0,1]$ represent some membership functions of the mentioned coverage conditions, then Eq. 5 can be rewritten as a product of these memberships:

$$C(\mathbf{s}_i^{(t)},\mathbf{q}) = \mu_d(\|\mathbf{p}_i-\mathbf{q}\|)\cdot\mu_\theta(\angle_\theta(\mathbf{q}-\mathbf{p}_i)-\theta_i^{(t)}) \cdot\mu_\xi(\angle_t(\mathbf{q}-\mathbf{p}_i)-\xi_i^{(t)})\cdot v(\mathbf{p}_i,\mathbf{q}). \tag{6}$$

Function $v(\mathbf{p}_i,\mathbf{q})$ is usually binary. Given a position $\mathbf{p}_i$, if the line of sight between sensor $\mathbf{s}_i$ and $\mathbf{q}$ is obstructed, then we assume that the visibility cannot be achieved ($v(\mathbf{p}_i,\mathbf{q})=0$), otherwise the visibility is fully attained ($v(\mathbf{p}_i,\mathbf{q})=1$).

Value $C=1$ means full coverage while $C=0$ indicates no coverage. If more than one sensor covers $\mathbf{q}$, a way to compute the local network coverage C_l of network N over location $\mathbf{q}$ at time t is:

$$C_l(N,\mathbf{q}) = 1-\prod_{i=1,\dots,n}\left(1-C(\mathbf{s}_i^{(t)},\mathbf{q})\right).$$

We define the following objective function for a given command set as the coverage it provides for all of the targets through time:

$$O(\mathbf{\Pi}) = \sum_{t=1}^{M}\sum_{\mathbf{q}\in\Xi} r_\mathbf{q}^{(t)}C_l(N,\mathbf{q}), \tag{7}$$

where $r_\mathbf{q}^{(t)}\in\mathbf{R}^{(t)}$ is the predicted probability of targets being at location $\mathbf{q}$ in the environment (provided by the trajectory prediction algorithm). In other words, the value of each location $r_\mathbf{q}^{(t)}$ within the prediction map defines the importance of that location for the coverage task. Then, the goal is to find a command set $\mathbf{\Pi}^*$ which maximizes this objective function:

$$\mathbf{\Pi}^* = \underset{\mathbf{\Pi}}{\operatorname{argmax}}\ O(\mathbf{\Pi}). \tag{8}$$

Notice in this formula, the trajectory prediction step defines the weights assigned to different locations in the environment through the weight parameter $r_\mathbf{q}^{(t)}$, and the sensor control step finds the command set $\mathbf{\Pi}^*$ based on that prediction. Therefore, both the prediction and control steps, affect the result of objective function. The defined objective function provides an estimate on the actual coverage that the network provides over the targets. The actual coverage is calculated using the following formula:

$$C_g(\mathbf{\Pi}) = \frac{1}{MJ|\Xi|}\sum_{j=1}^{J}\sum_{t=1}^{M}\sum_{\mathbf{q}\in\Xi}\mathbf{1}(\mathbf{T}_j^{(t)},\mathbf{q})\ C_l(N,\mathbf{q}), \tag{9}$$

where function $\mathbf{1}(\mathbf{T}_j^{(t)},\mathbf{q})$ returns one if target $\mathbf{T}_j$ is at location $\mathbf{q}$ at time t, and zero otherwise. The formula provides the average coverage of the network over all the targets during the whole simulation time.

B. PTZ Camera Coverage Model

The following real-valued membership functions are used in our sensor model. These functions provide a monotonically decreasing membership value over distance and relative angle of position to the sensor.

We propose to use the following function, based on the well-known sigmoid function, to evaluate the distance membership:

$$\mu_{di}^{(t)} = \mu_d(\underbrace{\|\mathbf{p}_i-\mathbf{q}\|}_{d_i}) = 1-\frac{1}{1+\exp\left(-\beta_d(d_i-\alpha_{di}^{(t)})\right)}, \tag{10}$$

with $\alpha_{di}^{(t)}$ and β_d as the parameters configuring the membership function. These parameters can be estimated using experimental observations on sensor behaviours (e.g., object recognition rate as a function of distance). Parameter β_d controls the slope of the function and $\alpha_{di}^{(t)}$ determines the distance where the sensor has 50% of its maximum coverage.

As for the pan angle membership functions, we propose the following function also based on the sigmoid function:

$$\mu_{\theta i}^{(t)} = \mu_\theta(\underbrace{\angle_\theta(\mathbf{q}-\mathbf{p}_i)-\theta_i^{(t)}}_{\theta_i'}) = \frac{1}{1+\exp\left(-\beta_\theta(\theta_i'+\alpha_{\theta i}^{(t)})\right)} - \frac{1}{1+\exp\left(-\beta_\theta(\theta_i'-\alpha_{\theta i}^{(t)})\right)}, \tag{11}$$

where $\alpha_{\theta i}^{(t)}$ controls the "width" of the function and β_θ controls the slope of the function at the boundaries. Note that the proposed function has a range of $\theta_i'\in[-180,180]$ degrees. Therefore, any calculated angle should be brought into this range accordingly. Similarly, membership function μ_ξ is defined as:

$$\mu_{\xi i}^{(t)} = \mu_\xi(\underbrace{\angle_\xi(\mathbf{q}-\mathbf{p}_i)-\xi_i^{(t)}}_{\xi_i'}) = \frac{1}{1+\exp\left(-\beta_\xi(\xi_i'+\alpha_{\xi i}^{(t)})\right)} - \frac{1}{1+\exp\left(-\beta_\xi(\xi_i'-\alpha_{\xi i}^{(t)})\right)}, \tag{12}$$

which has the range $\xi_i'\in[-90,90]$. The pan and tilt membership functions are related to the focal length value through the $\alpha_{\theta i}^{(t)}$ and $\alpha_{\xi i}^{(t)}$ parameters. More precisely, these parameters define the angle of view for the sensor, so we have:

$$\alpha_{\theta i}^{(t)} = 2\arctan(L_h,2f_i^{(t)}), \tag{13}$$

$$\alpha_{\xi i}^{(t)} = 2\arctan(L_v,2f_i^{(t)}), \tag{14}$$

where L_h and L_v are the horizontal and vertical sizes of the camera sensor, respectively.

The distance membership function is also related to the focal length value $f_i^{(t)}$ through the following formula:

$$\alpha_{di}^{(t)} = \left[\frac{\alpha_{d_{max}} - \alpha_{d_{min}}}{f_{max} - f_{min}} (f_i^{(t)} - f_{min}) \right] + \alpha_{d_{min}}, \qquad (15)$$

where $\alpha_{d_{max}}$ and $\alpha_{d_{min}}$ are the maximum and minimum value that $\alpha_{di}^{(t)}$ can take. In practice, the value of these parameters depends on the camera type, size of the targets, the minimum number of pixels inside the images needed for accurate detection of targets, and so forth.

C. Trajectory Prediction

Trajectory prediction is the process of estimating the future location of a given target using the information of its previous movements. More precisely, assume $\mathbf{T}_j^{(t)}$ is an ongoing trajectory of target j in the environment Ξ, currently being observed at time t, where $\mathbf{T}_j^{(t)} = (\mathbf{L}_j^{(t)}, \Delta\mathbf{L}_j^{(t)})$. Here, $\mathbf{L}_j^{(t)} = (x_j^{(t)}, y_j^{(t)})$ represents the location of the target and $\Delta\mathbf{L}_j^{(t)} = (\Delta x_j^{(t)}, \Delta y_j^{(t)})$ its previous time step displacement.

We wish to measure the location of the target after one time step (i.e., $\widehat{\mathbf{T}_j^{(t+1)}}$) as accurately as possible. Here we propose a probabilistic model for the future location of each target. More precisely, for a target $\mathbf{T}_j$ at time $t+1$ we define a discrete random variable $X_j^{(t+1)}$ over the sample space of all the locations $\mathbf{q} \in \Xi$. $X_j^{(t+1)}$ is defined by a probability mass function $P(X_j^{(t+1)} = \mathbf{q})$ which returns the probability of target $\mathbf{T}_j$ being at location $\mathbf{q}$ at time $t+1$. For brevity, we note this measure as $\mathcal{P}_{j\mathbf{q}}^{(t+1)}$. The goal is to incrementally learn the probabilistic model for each target as it moves within the environment.

The result of the trajectory prediction at time t is a prediction map $\mathbf{R}^{(t)}$. The prediction map, sums up the result of prediction for different targets which are moving in the environment. In other words, if the prediction method predicts that two targets are moving into one location with high probability, then that location is getting a high value in the prediction map. More precisely, for each element $r_{\mathbf{q}}^{(t)} \in \mathbf{R}^{(t)}$ we have:

$$r_{\mathbf{q}}^{(t)} = \sum_j \mathcal{P}_{j\mathbf{q}}^{*(t+1)}. \qquad (16)$$

In this paper, we use weighted probabilistic method to predict the displacement of each target in the environment. More precisely, the predicted probability that the target $\mathbf{T}_j^{(t)}$ moves to location $\mathbf{q}$ in the next time step ($\mathcal{P}_{j\mathbf{q}}^{(t+1)}$) is given by:

$$\mathcal{P}_{j\mathbf{q}}^{(t+1)} = \mathcal{N}_2(\mathbf{q}; \widehat{\Delta\mathbf{L}}_j^{(t+1)}, \mathbf{\Sigma}), \qquad (17)$$

where $\widehat{\Delta\mathbf{L}}_j^{(t+1)}$ is the mean of the bivariate distribution, and $\mathbf{\Sigma}$ is the covariance matrix. For simplicity we assume $\mathbf{\Sigma} = \sigma_p^2\mathbf{I}$. The mean is the exponential average of the recent movements of the target, given by:

$$\widehat{\Delta\mathbf{L}}_j^{(t+1)} = (1-\eta)\, \widehat{\Delta\mathbf{L}}_j^{(t)} + \eta\, \Delta\mathbf{L}_j^{(t)}, \qquad (18)$$

where $\eta \in [0,1]$ is the learning rate. Here, we assume $\eta = 0.8$ and $\sigma_p = 1$. For more on the trajectory prediction problem, readers are referred to [6].

III. Greedy Sensor-wise Coverage Optimization (GSCO)

There are several points which should be considered when designing the sensor control algorithm. First, the prediction map ($\mathbf{R}^{(t)}$) contains probabilities and not binary values as the predicted location of the targets. Therefore, the algorithm should be able to handle probabilistic predictions. Second, the sensors themselves have probabilistic coverage over the targets and this should be taken into consideration. Third, the movement of sensors is limited for each time step. These limitations put a boundary on the sensor's parameter values for the next time step. Finally, each sensor has several variable parameters and therefore a simple iteration over all possible combination of values is usually not possible.

Looking at the formulation of the problem, we observe that the sensor control problem has a combinatorial nature and therefore convex optimization algorithms do not apply for the problem. Besides, the algorithm is designed for an online setting, therefore we are aiming to have an algorithm which has linear complexity with respect to the number of sensors in the environment, so that we can get some guarantees on the running time of the algorithm in real settings.

Considering all of the mentioned criteria, we propose the GSCO algorithm. The main idea behind the GSCO algorithm is that target locations which have higher prediction value, and at the same time, can be covered by a fewer number of sensors should be given higher priority in the coverage task. In other words, in the GSCO algorithm, at each time step, the prediction map $\mathbf{R}^{(t)}$ is divided by the accessible coverage map $\Psi_{\mathbf{q}}^{(t)}$ (which defines the number of sensors that can cover any location in the environment), to define a weight map which determines the importance of each location for the coverage task. This weight map is later used to iteratively optimize the direction of all sensors in the environment.

The algorithm is shown in Algo. 1.

Different steps of the algorithm are explained in more detail below:

- Calculate $\psi_i^{(t)}$: For each sensor calculate the accessible coverage map $\psi_i^{(t)}$. This map holds the locations in the environment that the sensor $\mathbf{s}_i$ can effectively cover in the future. For its calculation, we take into consideration the current parameters $(\theta_i^{(t)}, \xi_i^{(t)}, f_i^{(t)})$, and the maximum speed (v_θ, v_ξ, v_f) for each sensor parameter.
  Another parameter used for the calculation of $\psi_i^{(t)}$ is a threshold parameter ϕ. This threshold was defined because (theoretically based on Eq. 6), the coverage of a sensor over all the locations from which it is visible is

Algorithm 1 The proposed GSCO algorithm for sensor control.

```
1:  inputs Sensor network $N$, parameter $\phi$, the prediction map $\mathbf{R}^{(t)}$, and the environment $\Xi$
2:  output Command set $\mathbf{\Pi}$
3:  while $N \neq \emptyset$ do
4:      for all $\mathbf{s}_i \in N$ do
5:          Compute coverage map $\psi_i$
6:          $\Psi_{\mathbf{q}} = \Psi_{\mathbf{q}} + \psi_{i\mathbf{q}}, \quad \forall \mathbf{q} \in \Xi$
7:      end for
8:      $\mathbf{U}_{\mathbf{q}} = \frac{\mathbf{R}^{(t)}_{\mathbf{q}}}{\Psi_{\mathbf{q}}}, \quad \forall \mathbf{q} \in \Xi$
9:      $\mathbf{s}_j = Rand(N)$
10:     Optimize parameters of $\mathbf{s}_j$ using $\mathbf{U}$.
11:     for all $\mathbf{q} \in \Xi$ do
12:         $r^{(t)}_{\mathbf{q}} = r^{(t)}_{\mathbf{q}} - (r^{(t)}_{\mathbf{q}} \,.\, C(\mathbf{s}^{(t)}_j, \mathbf{q}))$
13:     end for
14:     $\mathbf{N} = \mathbf{N} \setminus \mathbf{s}_j$
15: end while
```

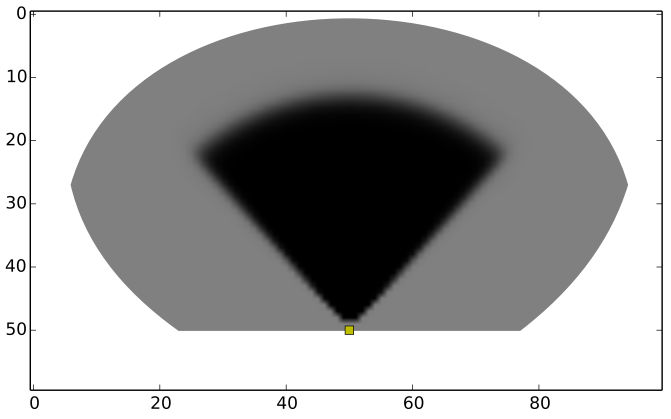

Fig. 1. Assuming that a sensor is positioned at (50,50) heading upward, the dark shaded area shows the current coverage of the sensor and the light shaded area shows the accessible coverage map ($\psi_i^{(t)}$) of the sensor in the next time step.

non-zero, while the coverage over most of these locations is close to zero. Therefore, by using the ϕ threshold, we only consider points for which the sensor can significantly change the coverage of the points it is covering. For an example of $\psi_i^{(t)}$, look at Fig. 1, where we have shown the current coverage of a sensor and its accessible coverage map.

- $\Psi^{(t)}_{\mathbf{q}} = \Psi^{(t)}_{\mathbf{q}} + \psi^{(t)}_{i\mathbf{q}}$: Using $\psi_i^{(t)}$, update the global accessible coverage map $\Psi^{(t)}$ for the environment. Each cell $\mathbf{q}$ in $\Psi^{(t)}$ has a value $\Psi^{(t)}_{\mathbf{q}}$ representing the effective accessible coverage of all sensors over this location. The $\Psi^{(t)}$ value for each location shows the potential of that location to be covered by different sensors. Notice that the values of different locations in the possible coverage map are real values (unlike previous work (e.g., as in [4], [5]) in which the values are integers). This is a natural consequence of the probabilistic coverage model we used for our sensors.
- $\mathbf{U}^{(t)}_{\mathbf{q}} = \frac{\mathbf{R}^{(t)}_{\mathbf{q}}}{\Psi^{(t)}_{\mathbf{q}}}$: Create a weight map $\mathbf{U}^{(t)}$. In this new weight map, the weight of the locations which can be covered by more than one sensor is proportionally reduced. This is the core of the algorithm, which assigns more weight to location that can be covered by less number of sensors.
- $\mathbf{s}_j = Rand(N)$: Randomly choose a sensor $\mathbf{s}_j$ from the sensors whose direction has not been optimized yet.
- Optimize parameters of $\mathbf{s}_j$ using $\mathbf{U}^{(t)}$: optimize the parameters of the sensor $\mathbf{s}_j$. The optimization is done using random sampling. Random sampling was shown to be more effective search method compared to other simple methods such as grid search [7], therefore we chose it as our optimization algorithm. For that purpose ω samples are randomly chosen from the range of possible pan and tilt angles, and zoom factors. The parameter set which achieves the best final coverage is chosen as the final direction of the sensor. The important point is that the evaluation for different parameter sets for a sensor is based on the weight map $\mathbf{U}^{(t)}$.
- $r^{(t)}_{\mathbf{q}} = r^{(t)}_{\mathbf{q}} - (r^{(t)}_{\mathbf{q}} \,.\, C(\mathbf{s}^{(t)}_j, \mathbf{q}))$: Once the direction and zoom level of the sensor $\mathbf{s}_j$ is found, its coverage is proportionally decreased from the prediction map. For example, assume that location $\mathbf{q}$ has the value of ($r^{(t)}_{\mathbf{q}} = 0.8$) in the prediction map. Also assume that the direction of sensor $\mathbf{s}_j$ is fixed and the coverage of the sensor over the location is $C(\mathbf{s}^{(t)}_j, \mathbf{q}) = 0.7$. Having this, the new value of the location in the prediction map becomes: $r^{(t)}_{\mathbf{q}} = 0.8 - 0.8 \times 0.7 = 0.24$.
- $\mathbf{N} = \mathbf{N} \setminus \mathbf{s}_j$: Remove the sensor $\mathbf{s}_j$ (for which the direction has been optimized) from the set of sensors.

The iteration continues until the parameters of all the sensors are determined.

IV. Experiments and Results

In the experimental section we compare the performance of the proposed GSCO algorithm with the CMA-ES method [8], a state-of-the-art metaheuristic for real-valued optimization. Notice we could not compare our method with other methods mentioned in Sec. I, given that none of the mentioned algorithms uses a probabilistic coverage model for sensors, and therefore our sensor coverage model is not applicable in those algorithms.

A. Simulated Dataset

In order to make the experiments for the trajectory prediction, we need an environment in which the targets displace and produce trajectories. Next, we explain this environment and how the target trajectories were simulated.

1) Map: For the experiment we used a map of Université Laval campus, in Québec, Canada. The map has 300 rows and 300 columns, with a resolution of one meter per dimension.

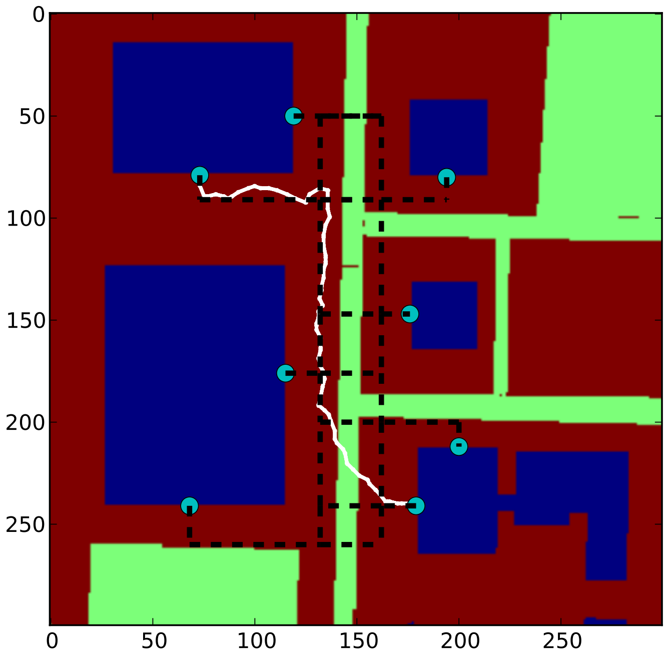

Fig. 2. There are eight gates on the campus map. Each gate is presented by a cyan circle. Targets can enter from any of the eight gates, walk around the campus and exit the environment through another gate. The trajectory of a sample target is also shown using the white line. The pedestrian paths inside the campus are shown using the dashed black line. The target follows the path to move between gates.

2) Targets: In our experiments each target is a simulation of a pedestrian walking on the campus. The target enters the environment from one gate of a building, walks inside the environment and exits from another gate (see Fig. 2). In this figure, blue areas represent buildings, green areas represent streets and parkings, and red areas are ground. Trajectories of targets are generated based on the current location of a target and on the temporary goal location it is trying to reach. The temporary goal is generated based on the shortest path between the initial gate and the final gate on the pedestrian path of the campus. Here, the pedestrian path is represented as a graph with intersections on the path being the vertices of the graph and the paths themselves being the edges. Each intersection on the shortest path between the two gates can be a temporary goal for a target.

At each time step, the next location of a target is randomly chosen from a distribution which itself is produced by the multiplication of two other distributions, namely the beta distribution and the Gaussian distribution. More precisely, at each time step, a Gaussian distribution is applied on the angle between the target's location and its temporary goal and a beta distribution is applied on the distance to determine the step size. The multiplication between these two distributions is normalized to sum up to one and then used to determine the probability of each location in the environment to be selected as the next location of the target. The Beta distribution has two parameters α and β and the Gaussian distribution has parameters μ and σ_a. The chosen values for these trajectory generation parameters are summarized in Table I.

TABLE I
PARAMETER VALUES FOR THE TRAJECTORY GENERATION.

Parameter	Value	Parameter	Value
α	2	β	2
μ	0	σ_a^2	125

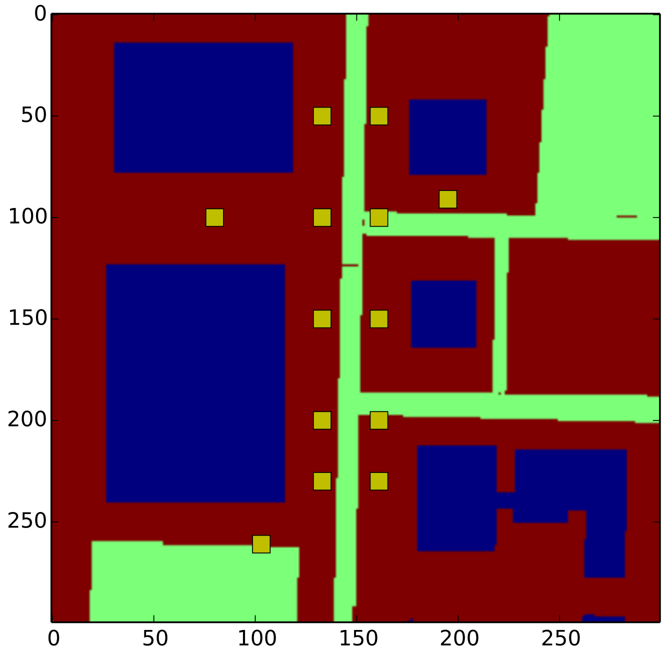

Fig. 3. The location of sensors is represented by the yellow squares on the map.

Notice that the dataset lacks the information for the sensors. The position of the sensors is presented in Fig. 3. In this figure the sensor locations are represented by yellow squares.

Sensors placed in the environment are the Pan-Tilt-Zoom cameras. The specification of the cameras is explained in Sec. II-B. For a reasonable model of a camera, we propose to use the parameters shown in Table II.

TABLE II
THE PARAMETER VALUES FOR A REALISTIC MODEL OF A PTZ CAMERA.

Parameter	Value	Parameter	Value
$L_h(mm)$	5.37	$L_v(mm)$	4.04
$\alpha_{d_{max}}(m)$	50	$\alpha_{d_{min}}(m)$	25
$f_{min}(mm)$	4.7	$f_{max}(mm)$	9.4
β_d, β_θ, and β_ξ	1	$\tau(m)$	1
ξ_{min}	-90^o	ξ_{max}	90^o
v_θ	$\pm30^{o/t}$	v_ξ	$\pm5^{o/t}$
$v_f(mm)$	±1.33	θ_{max}	360^o
θ_{min}	0^o		

TABLE III
PARAMETER VALUES FOR THE SENSOR CONTROL METHODS.

CMA-ES parameter	Value	GSCO parameter	Value
σ_{CMA}	0.33	ϕ	0.1
λ_{CMA}	14		

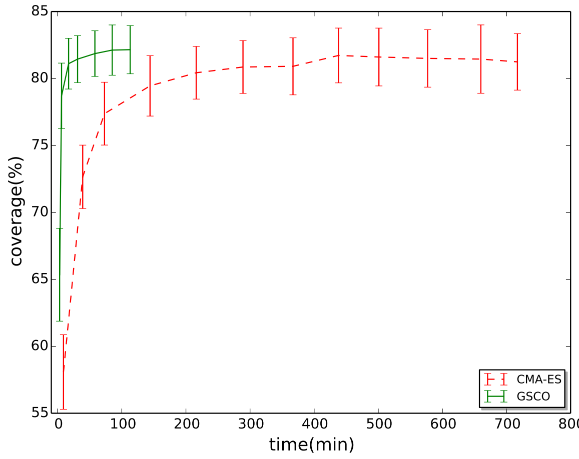

Fig. 4. Coverage percentage for experiments with 50 targets using CMA-ES and GSCO optimization methods over 200 time steps.

B. Sensor Control

We use the weighted probabilistic method (as explained in II-C) as the prediction method to determine the future location of targets. Then, the mentioned two optimization methods find the best commands for the sensors. The value of the parameters for the sensor control methods is summarized in Table III.

To test different scenarios, we produced simulations with different densities of targets. Therefore, at each time step, a number of targets equal to the ratio between the total number of targets to the simulation time enter the environment. For the experiments, we created simulations with 50, 100, and 150 targets in the map, with a total simulation time of $M = 200$ time steps. We produced 30 different simulation settings with each parameter set, and performed the experiments on them. The performance measure was previously defined (Eq. 9) as the average coverage the network provides over all of the targets through the whole simulation duration.

One of the critical parameters for the CMA-ES algorithm is the maximum number of iterations that the algorithm run for. This parameter directly affects the final performance and the running time of the algorithm. As it is a critical measure, we tested CMA-ES method with different values for the maximum number of iterations, ranging from 1 to 100. More precisely, the iteration numbers are $[1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$.

The same argument applies to the number of random samples ω taken by the GSCO algorithm. We performed several experiments, in which the ω parameter has the range

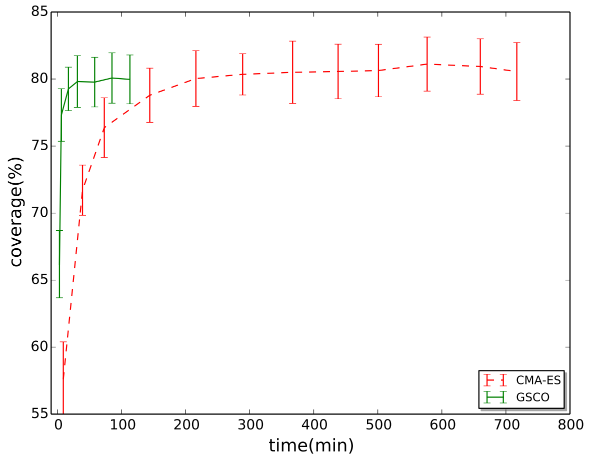

Fig. 5. Coverage percentage for experiments with 100 targets using CMA-ES and GSCO optimization methods over 200 time steps.

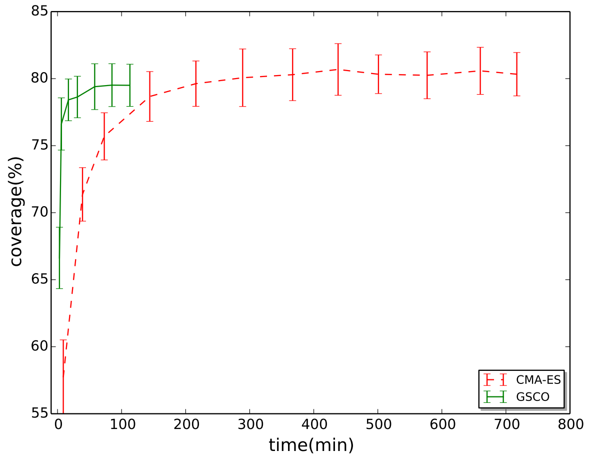

Fig. 6. Coverage percentage for experiments with 150 targets using CMA-ES and GSCO optimization methods over 200 time steps.

of $[1, 10, 50, 100, 200, 300, 400]$. The results are presented in Figs. 4, 5, and 6. The variance bars in the figures represent the result of the CMA-ES algorithm with one specific maximum number of iterations or the GSCO algorithm with a specific ω parameter over the 30 different simulation settings. In these experiments, we are looking for small running times and large coverage. Therefore, the best algorithm should produce results closer to the top-left corner of the figure. Also notice that the time axis represents the total simulation time over the 200 time steps in each simulation setting. In other words, to get the computation time of each algorithm for one time step, the total simulation time should be divided by 200.

We observe that in all of the experiments, the GSCO algorithm converged faster than the CMA-ES algorithm. This is an important aspect as the convergence time is crucial for sensor control algorithms, given that the results should be produced online, because the system needs to update the direction of the sensors at each time step, and therefore computation time becomes very important.

In terms of coverage performance, we observe that in the case of 50 targets, the GSCO algorithm performed better than the CMA-ES algorithm, while for the 100 and 150 targets cases, the performance of the CMA-ES was slightly better. Although in all the cases, the performance difference between the two algorithms is less than 1%, and not statistically significant.

We also observe that as the number of targets increases the overall coverage percentage decreases, which is also reasonable, as covering more targets is a more difficult task. The point is that the decrease in the performance of the GSCO algorithm is more significant than the CMA-ES algorithm. The reason could be that, the CMA-ES algorithm solves the sensor control problem as a general optimization algorithm and therefore the number of targets has less important effect on the algorithm, while the performance of the GSCO algorithm is directly related to the heuristic used to design the algorithm. It could be that the heuristic is more applicable to less crowded scenes with a lower number of targets.

We can argue that GSCO algorithm has a demonstrated advantage over CMA-ES for the problem of sensor control. Indeed, the final result of the two algorithms in terms of the coverage is similar while the GSCO algorithm converges faster and produces the result in less time, which is important for online control systems. Therefore, the GSCO algorithm is suitable for the systems which have real-time constraints.

V. Conclusion

In this paper we have presented the GSCO algorithm to optimize the parameters of the sensors in the temporal coverage optimization problem. The proposed algorithm has several advantages compared to other algorithms proposed in the literature, such as the capability to work with the probabilistic sensing model for sensors, to handle different certainties for target locations and to model the limitation of sensor movements.

We also compared the proposed method with a general purpose optimization algorithm (CMA-ES), and observed that our algorithm produces the results in a much shorter time in all the cases considered. Besides, the result of our algorithm was superior to that of the CMA-ES algorithm in simulations with a lower number of targets and competitive in simulations with more targets. Therefore the GSCO algorithm can satisfy the real-time requirements of systems without compromising on the performance in terms of the overall coverage of the network.

Acknowledgements

This project was funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Computational resources were provided by Calcul Québec / Compute Canada.

References

[1] V. Akbarzadeh, C. Gagné, M. Parizeau, M. Argany, and M. Mostafavi, "Probabilistic sensing model for sensor placement optimization based on line-of-sight coverage," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 2, pp. 293–303, Feb 2013.

[2] Y. Cai, W. Lou, and M. Li, "Cover set problem in directional sensor networks," in *Future Generation Communication and Networking*, vol. 1, Dec 2007, pp. 274–278.

[3] J. Ai and A. A. Abouzeid, "Coverage by directional sensors in randomly deployed wireless sensor networks," *Journal of Combinatorial Optimization*, vol. 11, no. 1, pp. 21–41, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10878-006-5975-x

[4] U.-R. Chen, B.-S. Chiou, J.-M. Chen, and W. Lin, "An adjustable target coverage method in directional sensor networks," in *IEEE Asia-Pacific Services Computing Conference*. IEEE, 2008, pp. 174–180.

[5] J. Wang, C. Niu, and R. Shen, "Priority-based target coverage in directional sensor networks using a genetic algorithm," *Comput. Math. Appl.*, vol. 57, no. 11-12, pp. 1915–1922, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.camwa.2008.10.019

[6] V. Akbarzadeh, C. Gagné, and M. Parizeau, "Target trajectory prediction in PTZ camera networks," in *Proc. of the IEEE Workshop on Camera Networks and Wide Area Scene Analysis (WCNWASA 2013)*, June 2013, pp. 816–822.

[7] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Resarch*, vol. 13, pp. 281–305, Feb. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2188385.2188395

[8] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159 – 195, 2001.