A Sampling-Based Motion Planning Approach to Maintain Visibility of Unpredictable Targets

Rafael Murrieta-Cid
Mechatronics Research Center
EGIC
Tec de Monterrey
Estado de México Campus
Edo de México, México

rafael.murrieta@itesm.mx

Benjamín Tovar
Department of
Computer Science
Siebel Center
University of Illinois
Urbana, IL 61801 USA

btovar@uiuc.edu

Seth Hutchinson
Dept. of Electrical and
Computer Engineering
The Beckman Institute
University of Illinois
Urbana, IL 61801 USA

seth@uiuc.edu

Abstract

This paper deals with the surveillance problem of computing the motions of one or more robot observers in order to maintain visibility of one or several moving targets. The targets are assumed to move unpredictably, and the distribution of obstacles in the workspace is assumed to be known in advance. Our algorithm computes a motion strategy by maximizing the shortest distance to escape *—the shortest distance the target must move to escape an observer's visibility region. Since this optimization problem is intractable, we use randomized methods to generate candidate surveillance paths for the observers. We have implemented our algorithms, and we provide experimental results using real mobile robots for the single target case, and simulation results for the case of two targets-two observers.*

Keywords: **pursuit-evasion, motion planning, visibility**

1 Introduction

In this paper, we consider the surveillance problem of maintaining visibility of mobile evaders (*the targets*) by sensor-equipped mobile robots (*the observers*) in a workspace containing obstacles. We assume that the observers are provided with a map of the environment, but that the trajectories of the targets are not known a priori. Thus, the planning problem is to find collision-free trajectories for the observers such that each target is always within the field of view of at least one observer. A preliminary version of this work appeared in [36] and related works are presented in [37, 38].

Our surveillance problem incorporates aspects of several difficult research problems, including path planning, pursuit-evasion, target tracking, vision-based control, and optimization. Even the problem of finding a collision-free path between two robots configurations has been shown to be intractable; the best known algorithm requires time that is exponential in the robot's degrees of freedom [8, 28]. As a consequence, in its general form, our surveillance problem is intractable. For this reason, we have adopted a sampling method for planning optimal observer trajectories. Randomized methods were introduced to solve robot path planning problems in the late eighties [3, 4], and have seen increasing popularity in recent years (see, e.g., [24, 15, 11]). Here, we adapt the random sampling approach to the problem of finding optimal surveillance paths.

A great deal of previous research exists in the area of pursuit and evasion, particularly in the area of dynamics and control [14, 13, 1]. This work typically does not take into account constraints imposed on observer motion due to the existence of obstacles in the workspace, nor visibility constraints that arise due to occlusion. In this paper, we focus on these often neglected geometric aspects of the problem, specifically those aspects of the problem related to collision avoidance and those that arise due to visibility constraints

imposed by obstacles in the workspace. In the present paper, we are less concerned with aspects of the problem related to dynamics, since many mobile robots have fairly simple dynamics (e.g., the two-wheeled differential drive robots that we have used in our experiments), and we confine our attention to the geometric aspects of the problem.

In the traditional robot path planning problem, initial and goal configurations are specified, and the problem is to find a collision-free path to connect them. In our problem, the planner must determine the goal configuration before planning the path. This determination is made by maximizing a utility function that measures the shortest distance the target must move to escape an observer's visibility region.

As with all randomized methods, the quality of the solutions obtained by our algorithm depends in some measure on the method used to generate the random samples. In our approach, sample generation is not uniform over the configuration space; visibility computations are used to bias the sampling, creating candidate robot configurations that have a better chance of being useful.

The execution time of our planning algorithms must be small enough to allow the observers to maintain visibility of the moving targets. Therefore, we perform a number of precomputations that are useful for on-line planning. This represents a trade-off between computation time and memory requirements.

We have implemented the proposed algorithms. In the case of a single observer, the algorithms were tested on real robots. Several experimental results are included in the paper. Algorithms for maintaining visibility of two evaders with two pursers have also been implemented, and simulation results for this case are included in the paper.

The remainder of the paper is organized as follows. Related research is discussed in Section 2. The proposed planner for maintaining visibility of a moving target is described in Section 3. In Section 3.1 we present an algorithmic description of this planner. We describe the required preprocessing in Section 3.2. A planner for two-pursuer/two-evaders is presented in Section 3.3. The robot architecture is described in Section 4. Experiments and results in real robots are also presented in Section 4. Conclusions and future work for the target tracking task are presented in Section 5.

2 Previous Research

As mentioned above, our problem is related to the problems of pursuit-evasion and path planning (among others), and in this section we review research in these areas as it relates to our problem.

The pursuit-evasion problem is often framed as a problem in non cooperative dynamic game theory [1]. A pursuit-evasion game can be defined in several manners. One of them consists in *finding* an evasive target with one or more mobile pursuers that sweep the environment so that the target does not eventually sneak into an area that has already been explored. Deterministic [42, 45, 12, 30, 43] and probabilistic algorithms [47, 16] have been proposed to solve this problem. The pursuers could also be interested to actually "catch" the evaders, that is, move to a contact configuration or closer than a given distance [13].

The previous problems are related but not the same as ours. In this paper, the problem consists of determining a motion pursuer strategy to always maintain the visibility between the evader and the pursuer. We call such a task target tracking. We assume that initially the pursuer can establish visibility with the evader.

The target tracking problem has often been attacked with a combination of vision and control techniques (see, e.g., [41, 9, 20]). Purely control approaches, however, do not take into account the existence of obstacles in the the environment, or the problem of coordinating different robots to track several targets. The basic question that must be answered is *where should the robot observer move in order to maintain visibility of a target moving in a cluttered workspace?* Both visibility and motion obstructions must be taken into account, and thus, a pure visual servoing technique can fail because it ignores the global geometry of the workspace.

A number of researchers have studied the problem of path planning in the context of target tracking. Game theory [1] is proposed in [29] as a framework to formulate the tracking problem. The case for predictable targets is also presented in [29], which describes an algorithm that computes numerical and optimal solutions for problems of low dimensional configuration spaces. However, the assumption that the motion of the target is known in advance is a very limiting constraint.

In [5], a tracking algorithm is presented that operates by maximizing the probability of future visibility of the target. This algorithm is also studied with more formalism in [29]. The approach uses a disk centered at the current target location to approximate the set of possible future target locations. The planner computes the area of the disk that remains visible when the observer position is randomly perturbed around its current location, and it selects the position that maximizes the area of the disk that remains visible from the new observer location. It is assumed that the disk has uniform probability density, and geometric constraints are incorporated by setting zero probability mass in the regions within the disk that correspond to obstacles. This technique was tested with a Nomad 200 mobile robot and gave relatively good results. However, the probabilistic model assumed by the planner is often too simplistic, and more accurate models are difficult to obtain in practice.

The work in [10] presents an approach that takes into account the positioning uncertainty of the robot observer. One contribution of this work is a technique that periodically commands the observer to move into a region that has no localization uncertainty (a landmark region) in order to relocalize and better track the target afterward.

In [44], an approach is presented for the problem of actively controlling the configuration of a team of mobile agents equipped with cameras so as to optimize the quality of the estimates derived from their measurements. This approach has been applied to a target tracking task, and demonstrated both in simulation and with actual robot platforms.

In [21], a technique is proposed to track a target without the need of a prior map. Instead, a range sensor is used to construct a local map of the environment, and a combinatorial algorithm is then used to compute a differential motion for the observer at each iteration. The advantage of this technique is that no explicit self-localization mechanism is required. Thus, the implementation of the tracking system becomes simpler.

Recently, some works have considered the problem of maintaining visibility of several targets with multiple robots. In [40] a method is proposed to accomplish this task in uncluttered environments. The objective is to minimize the total time in which targets escape observation by some robot team member. In [23] an approach is proposed to maintain visibility of several targets using mobile and static sensors. A metric for measuring the degree of occlusion, based on the average mean free path of a random line segment is used.

Other approaches have been proposed that are related to our research, even though they are not directly intended for maintaining visibility of a moving target.

In [17], a randomized motion planner is presented for robots that must avoid collision with moving obstacles under kinematic and dynamic constraints. This planner samples the robot state×time space by picking control inputs at random and integrating the equation of motions. This planner has been tested in both simulated and real environment.

In [25], an approach is presented that efficiently detects collision among multiple ballistic spheres moving in a 3D space. The approach subdivides the space in a hierarchical uniform scheme. Based on this subdivision three types of events are defined. The first is due to actual collision and the other two types occur when spheres move from subspace to subspace (entering and leaving).

In [33] a framework for real time planning in changing environments is presented. This approach is closely related to probabilistic road map approaches, but in contrast with other probabilistic road map methods, the preprocessing step creates a representation of the configuration space that can be easily modified in real time to account for changes in the environment.

As in [25] and [17], in our approach, we also deal with at least one moving obstacle — the target. It is possible to consider that a manner for the target to escape is by colliding with the robot pursuer. Our approach is also able to deal with this additional problem (see section 3). Our approach is closer to the approach presented in [33], where an intensive preprocessing step is used.

The motion planning algorithms proposed in this paper use a random sampling approach. These techniques were introduced into the robot path planning literature in [3, 4], and they have been shown to be very efficient in solving motion planning problems even in high dimensional configuration spaces [24]. These methods abandon the idea of constructing an explicit representation of the configuration space, by instead focus on exploring it. The main drawback is that these algorithms are not deterministically complete. However, it is possible to prove that the probability to find a collision free path (if it exists) will converge to 1 as

the number of samples tends to infinity [4]. In general these methods are simple to implement and to apply to real robots.

3 Planner

A basic issue in target tracking is determining the time horizon h of the plan. Tracking can be seen as a sequence of motion decision problems, and each decision in this sequence represents the action executed by the observer at each stage. If the target is totally predictable (i.e., if the target trajectory is known) it is then possible to make a global optimization for long time horizons. For instance, a utility function that maximizes the time that the observer sees the target and minimizes the observer motion can be defined and globally optimized. Planners that require knowledge of the target's trajectory produce off-line strategies.

For the case of partial or totally unpredictable targets, it makes more sense to compute short term plans. Unanticipated changes in the target trajectory can be taken into account by replanning. An on-line strategy computes a motion plan for the next h future stages, and replans in the next iteration for the following h future stages. Typically, h is a very small number.

We have designed a reactive planner that anticipates a worst case scenario for the target's next move (i.e., $h = 1$). This worst case move is assumed to be in the direction of the shortest path connecting the target position q^t with a point in free space outside the visibility region of the observer. The length of this path is the *shortest distance to escape* (SDE).

The planner computes an observer position that locally maximizes the shortest distance to escape. This maximization is done using a sampling based algorithm. The algorithm samples a number of potential configurations near the observer (its *reachability region*), and selects from among them the one that maximizes the SDE.

The quality and success of the generated plans depend on the observer's capabilities. One of these capabilities is the shape of its reachability region, which is a parameter in our planner. Since the pursuer is a nonholonomic robot we only generate samples at configurations that can be easily reached from the previous observer configuration.

Visibility is used to define the observer reachability region (see 3.1). This visibility is computed over a superset of the configuration space, namely a geometric expansion of the workspace that yields a similarly shaped polygon (without curves).

3.1 Algorithmic description

We represent each robot observer and each target by a point, and we model their motion using discrete time transition equations. Let each time step be of length δ. The position of the observer at time $k\delta$ is denoted by q^o_k, and that of the target by q^t_k. The transition equation for the observer is $q^o_{k+1} = f(q^o_k, \phi_k)$, where ϕ_k is an action chosen from a given action space Φ. Constraints such as velocity bounds can be encoded in f or Φ. Similarly, the equation for the target is $q^t_{k+1} = g(q^t_k, \theta_k)$, where θ_k is an action taken from some space Θ. When the target is only partially predictable, the observer knows Θ, but not the specific action θ_k to be executed by the target.

In our on-line planner, the action ϕ_k is computed at each step in order to maximize the shortest distance to escape (SDE) between the target location q^t_k and the boundary of the visibility region at the observer's future position q^o_{k+1}. Let $V(q^o_{k+1})$ be this visibilty region. This is illstrated in figure 1, in which the observer is the disk labeled with O, the target the disk labeled with T. The observer visibility region is the cone like area, the obstacles are in dark (gray), the dashed arrows indicate the distance from the target to the free edges. The dots are the sample candidates where the observer could go in order to maximize SDE. The large arrow indicates the closest point at which the target can escape the observer's field of view.

One basic operation in the calculation of the SDE is the computation of region $V(q)$ for different candidate configurations q. For a polygonal model with n edges, this computation can be done in $O(n \log n)$ time using a ray-sweep technique [39]. In this technique, one point sees another one if the line segment between them does not cross an obstacle at any point other than the endpoints. However, limitations in vision sensing

require the use of more realistic models, such as the viewing frustum of the sensor (angular field of view) and its maximum range. We have adapted the ray-sweep technique to take into account these parameters, which the planner reads as inputs. For typical cameras, the visibility region is shaped as a cone.

Each edge in $V(q)$ borders either an obstacle or free space. We denote the edges that border the obstacles as *solid* edges E_{fs} and the edges that border the free space as *free* edges E_{fr}. Also, let E^*_{fr} be the free edge of $V(q)$ closest to the target.

Note that one possible strategy for the target to escape is to chase the observer and try to collide with it. For this reason, the target itself is considered to define a free edge by our algorithm. Thus, maximizing SDE will also prevent a collision between the robot pursuer and the moving target.

To maximize the distance between the target and the boundary of the visibility region of the observer we need to compute the distance between q^t_k and E^*_{fr}, where $E^*_{fr} \in V(q)$ and q is candidate for q^o_{k+1}. Hereafter we call this distance $D_{q^t_k|E^*_{fr}}$. In order to compute $D_{q^t_k|E^*_{fr}}$, we need to determine the distance between the target position q^t_k and every free edge belonging to $V(q)$ for all candidates q.

We can identify two main cases for computing the distance between q^t_k and a given free edge E_{fr}. The first one happens when the target can see a given free edge E_{fr}. The computation of this distance is easily done using the Euclidean metric. The second case happens when the E_{fr} is not directly visible from the current location of the target. In this case, we use a geodesic metric to determine the distance between a given free edge E_{fr} and the target location q^t_k. This is illustrated in figure 2. The evader is the disk labeled T, the observer is the disk labeled O, the observer visibility region is in light gray. The obstacles are in dark gray. The first case (the target can see the free edge) is indicated with the number 1, and the second case (the target cannot see the free edge) with number 2.

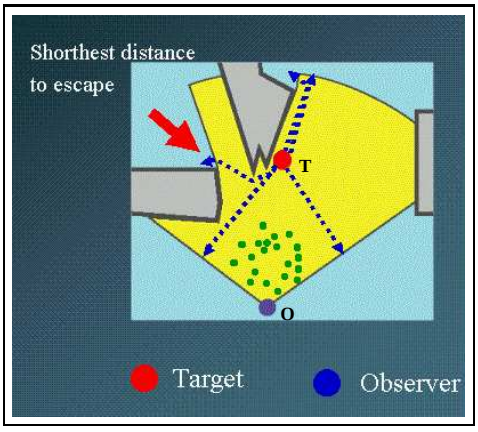


Figure 1: Shortest distance to escape

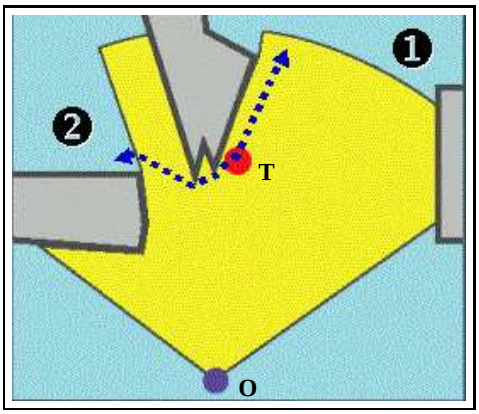


Figure 2: Shortest distance to escape: cases 1 and 2

The computation of the geodesic distance is done by determining the shortest path between q^t_k and all the free edges in $V(q)$ that are not visible from q^t_k by using the visibility graph of the polygonal map. To save time during the execution of the on-line planner, some of this computation is done in a preprocessing stage.

Visibility is also used to define the observer reachability region The original workspace is expanded by the robot radius. This expansion can be done offline with a Minkowsky sum. Curves are approximated by a set of straight lines. Let us call this new polygonal workspace P_r and R the robot reachability region. A visibility polygon $V(q)$ from the current robot position is computed over P_r. This visibility computation assumes unbounded range and omnidirectional sensing. The visibility region computation is done online only over line segments that are near the current robot position. More precisely, visibility is computed only taking into account the line segments associated to the bin of the index array (see below) where the robot currently lies.

Let us call R_{free} the intersection of the visibility polygon $V(q)$ and the robot reachability region R. Only the samples lying inside R_{free} are considered as candidates when computing the target shortest distance to escape. Note that, in this manner, it is certain that straight line robot paths lying totally inside R_{free} are collision free.

3.2 Preprocessing

Several precomputations are performed by our system. As with other approaches to motion planning [4, 24, 33], our method uses a two-stage approach. First, a preprocessing stage, and second an on-line stage. In the second stage planning is reduced to query processing. The main idea here is that the cost of planning in the fist step will be amortized over may episodes. Further, it allows us to do planning in real time during the second stage. Of course, such an approach requires significant memory to store the results of the preprocessing stage, but in our experiments (using a typical PC), memory limitations have not been a problem, even in maps with several thousands of vertices.

We assume that a polygonal map of the environment is provided, such as described in [46]. The algorithm PREPROCESSING is shown below, and its steps are as follows.

1. Compute the the visibility region from every vertex in the polygonal map (line 2 of algorithm PREPROCESSING) — see Fig. 3(B).

2. The space is discretized into a very thin grid. For every cell center of this grid (called the *visibility grid*), we compute a visibility region (line 5 of algorithm PREPROCESSING). These regions are stored in an indexed array. Retrieval time is thus linear in the size of the visibility region retrieved from the array.

3. The time to compute the *visibility grid* may be very large if we consider environments with thousands of vertices. To reduce the time required by the visibility computations we use another grid called the *index array*. We associate to every bin of this grid a list of the segments in the environment that lie inside the bin (see Fig. 3(A)). For every visibility computation in the visibility grid, we use the segment list from the corresponding bin in the index array to calculate the visibility region. The index array is computed at a much lower resolution than the visibility grid (line 1 of algorithm PREPROCESSING).

4. An approximate *visibility graph* is computed (line 8 of algorithm PREPROCESSING). Two vertices in the map are considered to be visible from each other only if the Euclidean distance is less than 2ρ and the line-of-sight between them is in free space —see Fig. 3(C). The constant ρ is the range of the observer's sensor.

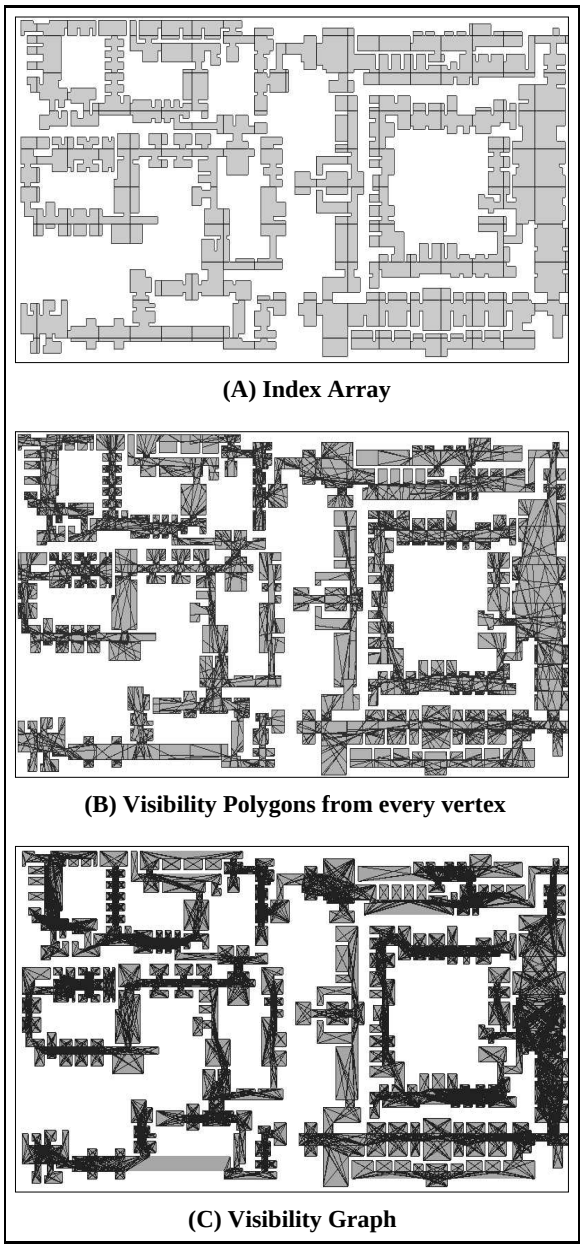
5. The shortest path between every pair of vertices in the visibility graph is computed using Dijkstra's algorithm (line 9 of algorithm PREPROCESSING).

We now describe how the precomputations are used to reduce the computational cost during run time.

First, the visibility computation takes constant time using the lookup table instead of $O(n \log n)$. When a free edge of $V(q)$ is not visible from a given target location, the shortest path between the target and such an edge must pass through a vertex of the environment. As a consequence, the geodesic distance between a free edge and the target in the general case can be computed as follows:

1. Determine the vertices of $V(q)$ that are visible to the target by testing which vertices are contained in the target's visibility region. Store these vertices in the list L_v (line 2 of algorithm TARGET_TRACKING).

2. Temporarily add the current target location q_k^t as a new node in the visibility graph, and connect this node to those ones corresponding to the vertices stored in L_v. The shortest distance between q^t and any vertex visible to the target can be computed by running Dijkstra's algorithm on this new graph.



(A) Index Array

(B) Visibility Polygons from every vertex

(C) Visibility Graph

Figure 3: Representations that are created during preproccesing.

3. The distance between q_k^t and each free edge in $V(q)$ is the solution to:

$$\boxed{D_{q_k^t|E_{fr}^*} = \min_{(v_o,v_f)}\{D_{q_k^t|v_o} + D_{v_o|v_f} + D_{v_f|E_{fr}}\},} \quad (1)$$

where v_o is any vertex in L_v, and v_f is any vertex in $V(q)$ that sees the free edge E_{fr}.

Visibility regions for different robot configurations are associated to every cell of the visibility grid (line 5 of algorithm PREPROCESSING). Note that since these visibility regions have limited angle of view and range, several have to be computed for every grid location (to cover the full 360 degrees) Samples around the current observer configuration are used to select a subset of those visibility regions computed prior to execution (line 6 of algorithm TARGET_TRACKING).

Samples are generated using three different integers, each of which corresponds to one of the parameters of the robot configuration (x, y, θ). The integers are used as indices to relate samples to the precomputed visibility regions at each grid location.

The visibility graph is used to compute the distance between every pair of vertices in $V(q)$. Thus, the term $D_{v_o|v_f}$ can be quickly evaluated using a precomputed lookup table. The visibility region from every vertex of the polygonal map is used to query which vertices in $V(q)$ can see a free edge in order to evaluate $D_{v_f|E_{fr}}$, these free edges are stored in list $L_{E_{fr}}$ (line 7 of algorithm TARGET_TRACKING). Thus, the use of precomputations allows us to accelerate the calculation of $D_{q_k^t|E_{fr}^*}$ during run-time operation.

The smallest distance $D_{q_k^t|E_{fr}^*}$ among all free edges of $V(q)$ is finally selected as the shortest distance to escape (SDE) (line 8 of algorithm TARGET_TRACKING).

Even though calculations for the creation of the *visibility grid* for an entire map with n vertices would require an algorithm with complexity $O(n \log n)$, the use of an *index array* can reduce the order of the calculations to a minimum of $O(km \log m)$, where m is the average number of vertices within each of the k bins in our array, $m \sim \frac{n}{k}$ and $m \ll n$.

Besides, for the *visibility graph* [48], only vertices with a distance of 2ρ are considered to be visible by the observer. If a robot could see all of the n vertices in the entire map, we would have n^2 paths between every pair of vertices. However, if there is a uniform distribution of the vertices over the map, we could expect an average of $\sigma(\frac{n}{\sigma})^2$ paths, where the σ factor can be considered as $(\frac{\alpha}{4\pi\rho^2})$, having α as the total area of the map, $\alpha \geq 4\pi\rho^2$. The memory used by our visibility graph preprocessing phase is bounded by these two limits, but in general the amount of memory is closer to the lower bound.

Algorithm PREPROCESSING describes the precomputation used in our approach and Algorithm TARGET_TRACKING the gives the entire procedure.

3.3 Planner for n observers and m targets

In our approach there is no predetermined assignment of a given target to a given observer. At any instant in time, the n observers locate themselves so as to maximize the distance to escape required by the m targets. We denote $V(q_{k+1}^{o_i})$ as the visibility region of observer i at location $q_{k+1}^{o_i}$. The shortest distance to escape for a target tracked by n observers is defined as the shortest distance between $q_k^{t_j}$ (target j at configuration q and iteration k) and the boundary of the union of the visibility regions $V(q_{k+1}^{o_1}) \cup V(q_{k+1}^{o_2}) \ldots \cup V(q_{k+1}^{o_n})$. We denote $d_j = D_{q_k^{t_k}|E_{fr}^*}$ as the distance between the j_{th} target and the closest free edges belonging to the boundary of $V(q_{k+1}^{o_1}) \cup V(q_{k+1}^{o_2}) \ldots \cup V(q_{k+1}^{o_n})$.

Thus, by reasoning over the union of the observer visibility regions and given that there is no predetermined assignment of a given target to a given observer, it is theoretically possible to generalize our approach for n observers m targets. However, it is understood that there is a real-time issue.

An additional issue, in the case multiple and targets multiple observers is to chose an appropriate metric to select the future observers' configurations $q_{k+1}^{o_i}$. We maximize the power mean also called Hölder mean

Algorithm 1 PREPROCESSING(Map)

Input:
Map: Polygonal map
Output:
V_v: Vertices visibility polygons
$DistTable[i,j]$: Array of distances between pairs of vertices
V_{grid}: Visibility grid
I: Index Array

Local variables
V_{graph} Visibility graph, p point, b_i bin, $V()$ Polygon representing a visibility region

```
I ← COMPUTE_INDEX_ARRAY(Map)
V_v ← COMPUTE_VISIBILITY_FOR_EVERY_VERTEX(I)
for all b_i ∈ I do
  for all p ∈ b_i do
    V(p) ← COMPUTE_VISIBILITY(p, b_i)
    V_grid ← V_grid ∪ V(p)
  end for
  V_graph ← COMPUTE_VISIBILITY_GRAPH(Map)
  DistTable[i,j] ← COMPUTE_LOOK_UP_TABLE(V_graph)
end for
```

Algorithm 2 TARGET_TRACKING(q_o,q_t,V_v,$DistTable[i,j]$,V_{grid})

Input:
q_o: Observer configuration
q_t: Target position
V_v: Vertices visibility polygons
$DistTable[i,j]$: Array of distance between couples of reflex vertices
V_{grid}: Visibility grid
Output:
q_{o+1} Observer new configuration

Local variables
P array of configurations (x,y,θ), p configuration (x,y,θ), $D_{q_k^t|E_{fr}^*}$ distance target-closet free edge
$dist$ double, $V()$ Polygon representing a visibility region, L_v list of vertices, $L_{E_{fr}}$ list of free edges

```
while q_t ∈ V(q_o) do
  L_v ← COMPUTE_TARGET_SEE_VERTICES(q_t,V_v)
  P ← SAMPLES(q_o)
  D_{q_k^t|E_fr^*} ← 0
  for all p ∈ P do
    V(p) ← SELECT_VISIBILITY(p,V_grid)
    L_{E_fr} ← COMPUTE_VERTICES_SEE_FREE-EDGES(V(p),V_v)
    dist ← MIN_DIST_ESCAPE(p,q_t,L_{E_fr},L_v,DistTable[i,j])
    if dist > D_{q_k^t|E_fr^*} then
      q_{o+1} ← q_o
      D_{q_k^t|E_fr^*} ← dist
    end if
  end for
end while
```

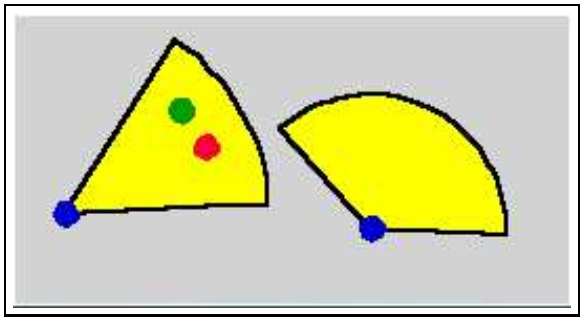


Figure 4: Observer tracking two targets

U, defined in [7] as

$$U = \left[\frac{1}{m}\left(\frac{d_j}{w_j}\right)^p\right]^{\frac{1}{p}}, \quad \text{with } d_j = D_{q_k^{t_j}|E^*_{fr}} \tag{2}$$

in which the $q_k^{t_j}$ are the positions of target j at configuration k, and d_j denotes the distance between target j and the closest free edges belonging to the boundary of $V(q_{k+1}^{o_1}) \cup V(q_{k+1}^{o_2}) \ldots \cup V(q_{k+1}^{o_n})$. Here w_j, and p are positive constants. The constants w_j weight the relative importance of keeping track of each target. U can take different forms according to the value of p:

1. $U = \min(d_1, d_2, \ldots, d_m)$ when p tends to ∞ (assuming $w_i = 1$). The general idea of this metric is that the performance of the system will depend on the weakest component.

2. When p tends to 0 then $U = \sqrt{d_1 d_2 \ldots d_m}$ which corresponds to the geometric mean.

We have tested in similation these two metrics for the specific case of two observers and two targets. In the first case, an oscillation in the observers appears if the targets exchange positions. The second criterion produced better results, and less oscillations occurred when the targets exchanged positions.

We have developed, implemented and simulated a planner for two observers and two targets. Given that there is no predetermined assignment of a given target to a given observer, the observers can switch targets in order to maximize U. We have left for future work a number of cases for which there is an undesirable 'shadowing' effect: if one observer sees both targets and the other observer does not see any of the targets, the last observer remains motionless rather than assisting the first observer (see figure 4).

4 Implementation and Experiments

Our tracking system has been implemented on a Super-Scout mobile robot from Nomadic Technologies. The Super-Scout is a differential-drive robot, and is equipped with a Pentium 233 MHz computer. The robot is fitted with an upward-pointing Sony XC-75 CCD camera for landmark detection, and a forward Sony EVI-30 CCD moving camera for target tracking.

4.1 Architecture

Our target tracking software incorporates 5 main modules: (1) a frame server, (2) a visual target detector and camera motion controller, (3) a localization module based on artificial landmark detection, (4) a motion planner and (5) a motion controller and system coordinator. Our robot is equipped with two cameras. The frame server grabs the RGB images from the hardware and separates the RGB components of the image.

A Sony EVI camera with an integrated pan-tilt unit is used to detect a target. The visual target detector and the camera controller maintain a lock on the target. The detector module recognizes a target and

identifies its pose with respect to the camera in real time. The visual target tracking and visual servoing problems have received considerable attention in the robotic and computer vision communities over the last years. Several techniques have been reported in the literature, and a variety of algorithms have been proposed [9, 41, 22, 20].

Our visual target detection uses a very simple and fast vision algorithm. A cylindrical mobile robot (a Nomad 200) acts as target, and unobtrusive rectangular patterns are placed on its hull. Each pattern has a binary barcode identifier. The algorithm computes subpixel image positions of the pattern's corners to estimates its 3D pose [27]. With the pose and barcode information of the detected patterns, the algorithm then infers the location and bearing of the target The detection algorithm we use is very simple, but we could instead use more advanced tracking algorithms such as those described in [18, 41, 34]. We have not done so, since our main concern is the motion planning problem, and not the vision problem of tracking moving objects. The range of the target detection module is approximately 80 inches and runs at a rate of 30 frames per second.

We use the pan-tilt unit to extend the maximal range and angular field of view of the camera. This unit is able to execute $[-100, 100]$ deg. pan action, $[-25, 25]$ deg. tilt action and active zoom (f = 5.4 mm to 64.8 mm). Our implementation presently only uses the pan action. We are currently incorporating tilt and zoom actions. The motion of the camera is computed by a dedicated controller rather than by the planner. This camera motion, however, is taken into account by the planner, which considers the total field of view of the vision system as the sum of the field of view of the camera (40 deg.) plus the motion of the pan-tilt unit ([-100 100] deg.).

As the observer moves around in its environment, it must keep track of its current position. To localize the robot we use artificial landmarks. Our landmarks are placed on the ceiling at known positions throughout the observer's workspace. Several works have dealt with the use of landmarks in robot navigation [19, 26, 32, 35]. The landmark detection module onboard our observer is the result of the work developed in [6]. The idea behind this approach is to provide the positions of the landmark as an input map to the observer. Since it is not necessary to relocalize at a high frequency rate, the landmark detector runs at 0.5-1 Hz.

The global planning algorithm is the one described in Section 3. The output of this planner is sent to a motion controller that drives the robot observer to the goal provided by the planner.

It has been shown in [2] that the time optimal trajectories for a differential drive robot consist in turning on site and in straight line robot motions. The paths generated by our planner have the same structure. That is, the robot executes a plan in three steps. First, the robot rotates aiming to the goal. Second, it translate to this goal. Finally, the robot rotates again to reach its final orientation. Additionally, these resulting robot paths lying inside R_{free} are not needed to be tested for collision (as it is mentioned above).

We avoid the use of a real-time kernel with a specific controller design. The controller consists of a triple layered strategy: A linear compensator, an adaptive scheme that keeps the compensator tuned, and a time pacer that regulates the control cycle to a specified rate [21]. The linear compensator is designed using a pole placement scheme. The adaptive loop keeps the compensator poles in place by recomputing the controller parameters as the control rate and the duty cycle drift from their initial estimated values. The time pacer is particularly important as it gives the kernel an appropriated time slack to attend other processes. Without the time pacer, the control cycle will be interrupted arbitrarily during the execution.

In order to improve the performance of the whole system, we run the planner program on an off-board computer. This allows us to increase the execution speed of the system by splitting tasks among two processors. The motion controller, the landmark module and vision programs run onboard the robot. The planner runs on a separate Celeron 600 MHz computer

4.2 Simulation and experiments using a mobile robot

We have tested our planner in several simulated scenarios and in actual experiments with the SuperScout robot. We will describe here the results of 3 simulations: one-robot/one-target in an environment with holes, one-robot/one-target in an environment composed of over a thousand vertices, and a two-robot/two-target example in a simple environment. We also provide snapshots of a test run with the SuperScout robot using our planner.

Results

In general, the performance of the planner and its eventual ability to keep the target always in view depend on a number of parameters: (1) the size of the viewing frustum of the sensor (cone angle), (2) the maximum range of the sensor, (3) the number of samples generated as candidates of q^t_{k+1}, (4) and the shape and size of the observer's reachability region.

We have tested four reachability region shapes (see figure 5): a circle centered at the observer position, two circles that intersect at the observer position (figure 5a), two ellipses that intersect at the observer position (figure 5b), two triangles sharing a single vertex at the observer position (figure 5c). The shape of the reachability region that experimentally gave the best results in tracking the moving evader has the shape of two triangles that share a common vertex. The two triangles gave better results because the differential drive robot used in the experiments can reach more quickly the sample configurations lying inside the triangles.

Figure 6 shows the result of a simulation for one-robot/one-target in a typical environment. The disk in dark gray represents the observer and the disk in light gray represents the target. The cone like region represents the observer's visibility region. Figure 7 shows a more complicated example consisting of an environment composed of 48 edges and 8 holes. The target is shown with as a small disk and the observer as a bigger disk. For this example, plans can be computed with a frequency of 19.44 Hz for a sampling set of size 9, at 10.31 Hz for a set of size 18, and at 7.71 Hz for a set of size 50

Figure 8 shows a very large polygonal map. The maps corresponds to a section of the Louvre museum. In this figure, the target is shown with a small light disk and the observer with a bigger dark one. The map has 1407 vertices. The *index array* consists of 400 cells, the running time to compute this index array was 37 s. The *visibility grid* has 92,416 cells. The time required to compute this grid was 9 min. and 30 s. The visibility graph was computed in 144 s. The visibility regions for all the vertices in the map required 5.9 s. The entire precomputation process took about 13 minutes. At each iteration during run time, the planner generated 75 samples in a neighborhood around q^o_k in order to compute the next observer position. By using precomputations, the planner can compute q^o_{k+1} with a freq. of 13 Hz. Without the precomputation the sole visibility region computation with 1400 vertices takes about 3 seconds per sample.

The performance of the planner depends on the size of the reachability region, independently of the number of samples. In our experiments using real robots, we have determined that in general samples lying further than 70 inches or so from the current robot position do not give good results. It takes too much time for the robot to reach them. We have tested our planner with a set of samples of size: 9, 18, 50 and 75. Results were somewhat better for 50 samples, however they are almost equivalent for all of these. A small set of samples explores a smaller number of configurations, but it takes a smaller amount of time to test smaller sample sets, thus the target does not have the opportunity to move far. Sets containing more than 100 samples do not give good results — the target escapes. The main reason for this failure is that reachability regions having more than 100 samples are very large, and consequently it takes too much time to reach the most distant of these samples.

Figure 9 shows a simulation experiment with two-robots/two-targets. It is possible to see how the observers switch targets. At first, observer 1 (square in dark gray) tracks target 1 (disk in dark gray) and observer 2 (square in light gray) tracks target 2 (disk in light gray). Once the targets cross each other the switch is done, observer 1 tracks target 2 and observer 2 tracks target 1.

Figure 10 presents as experiment with the mobile robot. The tracking is done by pure visual servoing without any planner. The controller tries to track the target from a given distance (50 in.) and maintain the target in the center of the camera's field of view. We can see in the figures that this strategy fails when the target turns around a corner, and the observer runs into a wall.

Figure 11 shows snapshots of an experiment performed on a real robot running our planner. The planner and global architecture presented in this paper were used. It is possible to see how the robot observer is capable of tracking the target even when it turns around a corner. The experiment shown in figure 11 was done under the same conditions as that shown in figure 10. The velocity and trajectory of the target were the same in both cases. The difference between both experiments is that one was done with planner and the other without it.

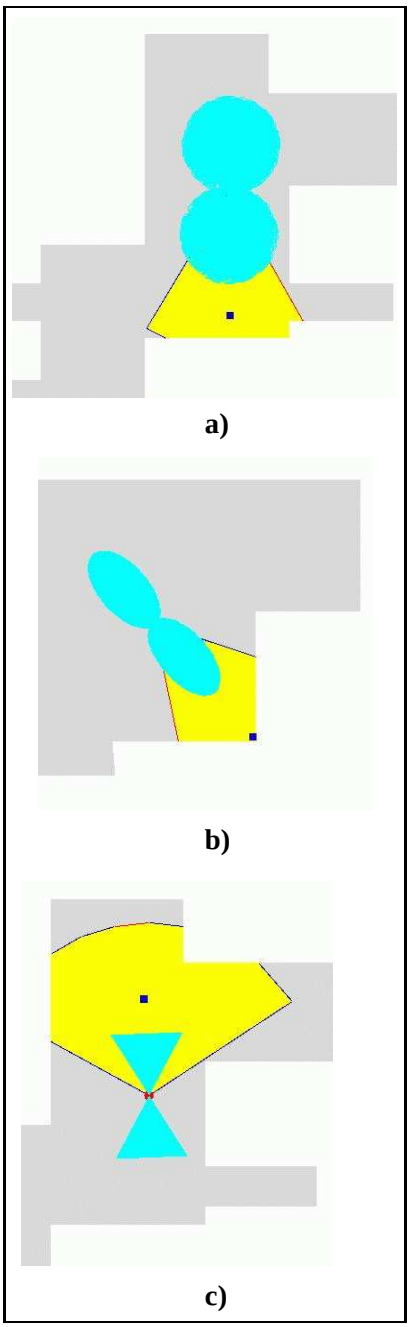


Figure 5: Reachability region shapes

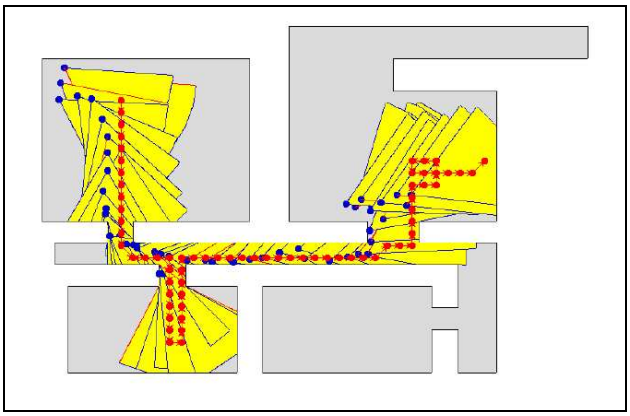


Figure 6: Simulation using the map of the Stanford's Robotics Laboratory.

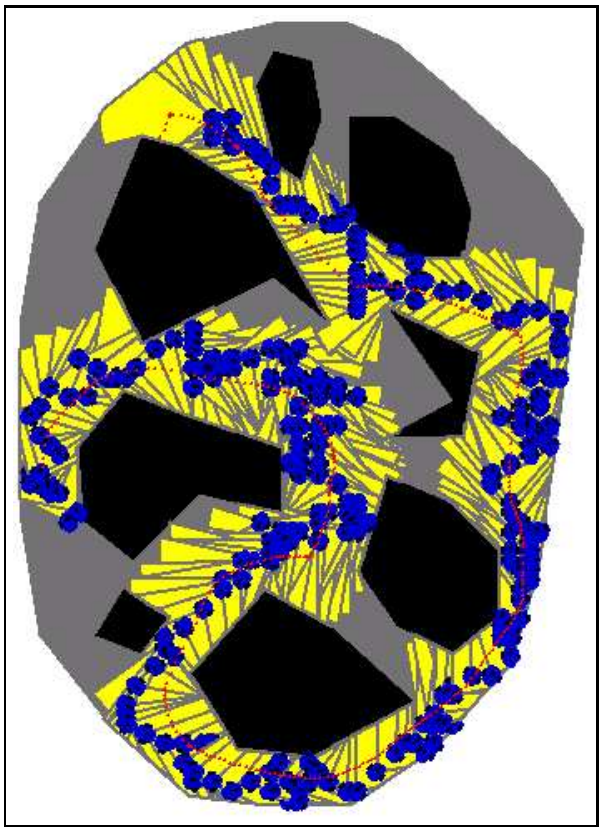


Figure 7: An example of an environment with holes.

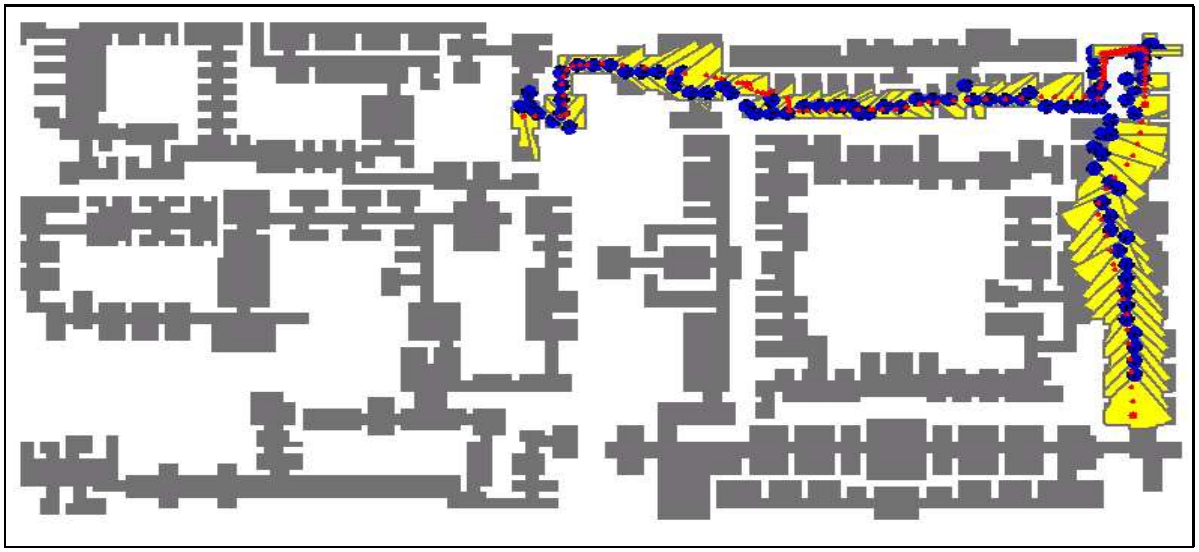
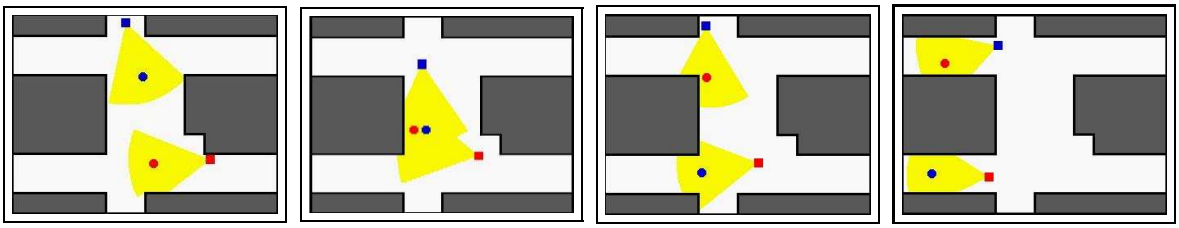
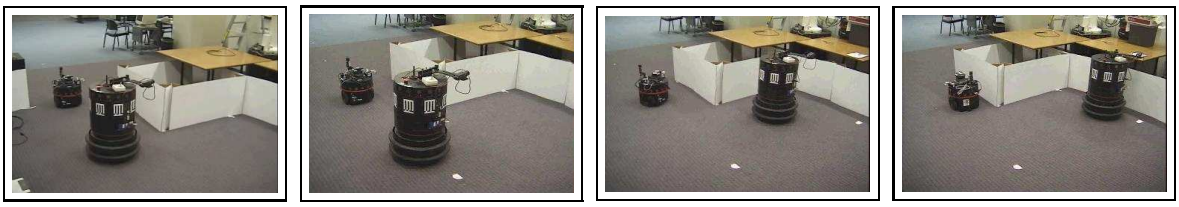


Figure 8: Large Map



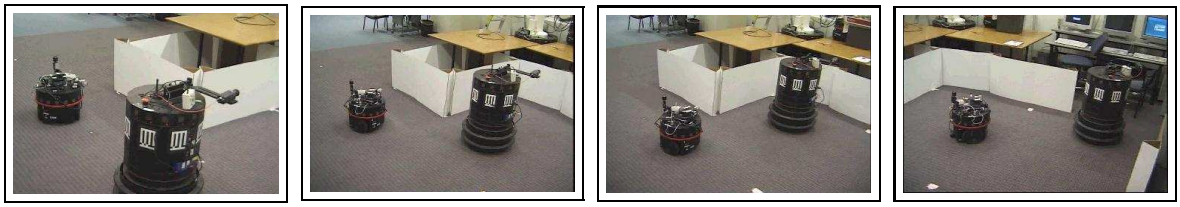
(a) (b) (c) (d)

Figure 9: Observers switching targets



(a) (b) (c) (d)

Figure 10: Experiment without planner



(a) (b) (c) (d)

Figure 11: Experiment with planner

Failures

In some cases, the size of the observer's reachability region is too small and the planner fails to track the target. On the other hand, a large reachability region requires more samples, and thus a plan requires more time to compute. This can also be a reason of failure. A small sample set can also hurt performance, because it affects the precision by which the SDE is calculated.

Another reason for failure is a small visibility region. Usually, this is due to a very narrow viewing frustum or a short sensor range. In simulation this can be easily corrected, but not so in a real system. For a real system, small visibility regions can only be improved through better sensing hardware.

Finally, there are conditions related to the target speed and target position relative to the obstacles, where the target will always escape. In our current work [37, 38], we are investigating complete algorithms able to determine whether or not the evader can escape.

5 Conclusions and Future Work

In this paper, we presented a motion planning approach to maintain visibility of moving evaders with mobile robots. One characteristic feature of this study is the need to satisfy visibility constraints while planning motions. The motion planning algorithms proposed in this paper are based on sampling.

In our work, the goal is to maintain visibility of the target in the presence of obstacles. This is a significant difference from other methods that have been proposed for the target tracking problem. For example, in [44] an approach is presented to the problem of actively controlling the configuration of a team of mobile agents equipped with cameras so as to optimize the quality of the estimates derived from their measurements. This approach has been applied to a target tracking task. However, this approach does not deal specifically with the problem of maintaining visibility of the targets; visibility obstructions are not taken into account.

The originality of our work derives in part from the fact the robot goal must be determined at each iteration of the algorithm. Another difference with classical motion planning is that we directly take into account sensor and actuator capabilities in order to generate the motion strategy.

The crux of the algorithms proposed here is a sampling based motion planning method that selects where to move the robot next. Our planner selects the next robot configuration based on maximizing a *utility* that is a function of the minimal distance required by target to escape the pursuer visibility region. The proposed planner selects a sample candidate that can be reached by following a collision-free path and maximizes this minimal distance to escape.

In this paper we have argued that visual servoing approaches to target tracking [41] have limited success because they do not take into account the complexity of the environment. Geometry-based algorithms go beyond these limitations.

Our strategy is an improvement over maximizing the probability of future visibility [5, 29, 10] for those cases when a probability model of the target is hard to obtain. By maximizing the distance to escape, we compute an observer motion that anticipates the worst case action of the target in the immediate future. The result is a short term reactive planner that can run in real time.

In [21], an approach is given that build a local map online and, based on this map, a plan is generated to move the observer to a position that maintains visibility of the target. Our method follows different approach. First a global map is divided in several small maps. To save time during the execution of the on-line planner, several computations are done in a preprocessing stage. We split the global map and perform the preprocessing step in order to deal with maps having thousands of vertices. However, if the map is small enough (e.g., if it contains less than 100 vertices or so) our approach does not need the preprocessing step. It is able to make all the computation in real time. Another difference between our approach and the work presented in [21] is that we have extended our method to deal with multiple targets and observers.

The proposed algorithms have been implemented and experiments on real robots are included. The quality of the plans mainly depends on the number of generated samples and the robot observer capabilities.

For future work, we want to sample the control space as opposed to the workspace in order to compute the SDE. In this scheme, the maximization of SDE is carried out in control space, but the evaluation of each sample is still done using the workspace geometry. With this strategy we can avoid the use of heuristics for estimating the size and shape of the reachability region. Instead, the observer's angular and translational speeds are computed directly by the planner. This scheme will also improve the hand shaking between the planner and the controller because the commands sent to the robot observer will be generated more smoothly.

Our current strategy assumes that the target is unpredictable. It is for this reason that our planner is reactive. If a motion model of the target is available, it should be possible to compute a longer term plan by maximizing SDE over the feasible target trajectories. Currently, the shortest escape path computed by our algorithm may not be feasible to the target at all. As a result, our algorithm may act more conservatively than it is required.

As mentioned above, in our current work, we are investigating complete algorithms able to determine whether or not the evader can escape observer visibility given a map of the environment and observer capabilities (i.e, maximal observer speed, maximal visibility range, delay due to image processing or execution time of motion planning algorithms) [37, 38].

For the case of multiple robots and multiple targets, an interesting topic for future research is the study of the occlusions caused by the observers on each other. Another open problem is the coordination of multiple observers with a decentralized planner. Our current approach is centralized, and does not admit an easy decomposition into distributed components. For a real system consisting of many observers, such a decentralized planner will be preferable over a centralized approach.

Acknowledgments

The authors thank Jean-Claude Latombe for his contribution to the ideas presented in this paper. The authors also want to thank Héctor González Baños for his suggestions on the implementation of our algorithms, Cheng-Yu Lee for his help in the development of the robotic system and Erik Millan for his comments of the final manuscript. This research was partially funded by CONACyT project J34670-A and by the ITESM Campus Ciudad de México, México.

References

[1] T. Başar and G. Olsder. Dynamic Noncooperative Game Theory. *Academic Press. 1982.*

[2] D.J. Balkcom and M.T. Mason. Geometric construction of time optimal trajectories for differential drive robots. *Fourth Workshop on Algorithmic Foundations of Robotics*, pp 1-13, 2000.

[3] J. Barraquand, L. Langlois and J.C. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. In *proc Fifth Int. Symposium on Robotics Research*, 1989.

[4] J. Barraquand and J.C. Latombe. Robot motion planning: A distributed representation approach. In *Int. Journal on Robotics Research* 10(6):628-649, 1991.

[5] C. Becker, H. Gonzlez-Baños, J.-L. Latombe and C. Tomasi. An intelligent observer. In *Int. Symposium on Experimental Robotics*, 1995.

[6] C. Becker, J. Salas, K. Tokusei and J.C. Latombe. Reliable navigation using landmarks. In *IEEE Int. Conf. on Robotics and Automation*, 1995.

[7] P.S. Bullen The Power Means Chapter 3, in Handbook of Means and Their Inequalities. In *Kluwer*, 2003.

[8] J.F. Canny. The complexity of the robot motion planning. *MIT press, Cambridge, MA*, 1988.

[9] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Trans. Robot and Autom.*, 8(3):313-326, June 1992.

[10] P. Fabiani and J.C. Latombe. Tracking a partially predictable object with uncertainty and visibility constraints: a game-theoretic approach, *IJCAI*, 1999.

[11] R. Geraerts and M. H. Overmars. A comparative study of probabilistic roadmap planners. In *Proceedings of Workshop on Algorithmic Foundations of Robotics*, pages 43–57, 2002.

[12] L. Guibas, J.-C. Latombe, S.M. LaValle, D. Lin and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. *In Proc 5th Workshop on Algorithms and Data Structures*, 1997.

[13] R. Isaccs. *Differential Games.* Wiley, New York,NY, 1975.

[14] O. Hájek. *Pursuit Games.* Academic Press, New York, 1965.

[15] Li Han and Nancy M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *Proceedings of Workshop on Algorithmic Foundations of Robotics*, 2000.

[16] J. Hespanha, M. Prandini, and S. Sastry. Probabilistic Pursuit-Evasion Games: A one-step Nash approach. In *proc Conference on Decision and Control*, 2000.

[17] D. Hsu, R. Kindel, J.C. Latombe and S. Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles. In *Workshop on Algorithm Foundations of Robotics*, 2000.

[18] D.P. Huttenlocher, W.J. Rucklidge and J.J. Noh. Tracking non-rigid objects in complex scenes. In *Fourth Int. Conf. on Computer Vision*, 1993.

[19] S. Hutchinson. Exploiting visual constraints in robot motion planning. In *IEEE Int. Conf. on Robotics and Automation*, 1991.

[20] S. Hutchinson, G. Hager and P. Coke. A tutorial on visual servo control. In *IEEE Transactions on Robotics and Automation*, Vol 12 No. 5 1996.

[21] H.H. González-Baños, C.-Y. Lee and J.-C. Latombe. Real-Time Combinatorial Tracking of a Target Moving Unpredictably Among Obstacles. In *Proc IEEE Int. Conf. on Robotics and Automation*, 2002.

[22] S. Jiansho and C. Tomasi. Good features to track. In *Conf. on Computer Vision and Pattern Recognition*, 1994.

[23] B. Jung and G. Sukhatme. Tracking targets using multiple robots: the effect of environment occlusion. In *Journal Autonomous Robots*, vol. 12 pp. 191-205, 2002.

[24] L. Kavraki, E. Svestka, J.C. Latombe and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In *IEEE Trans. on Robotics and Automation* 12(4):556-580, 1996.

[25] D. Kim, L. Guibas, S. Yong and S. Shin, Fast Collision Detection Among Multiple Moving Spheres, *IEEE Trans. Visualization and Computer Graphics*, 4(3):230-242, 1998.

[26] D.J. Kriegmen, E. Triendl and T.O. Binford. Stereo vision and navigation in buildings for mobile robots. *IEEE Trans. on Robotics and Automation*, 5(6):1722–1727, 1991.

[27] K. Kanatani, Geometric Computation for Machine Vision, *Oxford Science Publications*, 1993.

[28] J.-C. Latombe. Robot Motion Planning. Kluwer Academic Publishers, 1991.

[29] S. Lavalle, H. H. González-Banos, C. Becker, and J. C. Latombe. Motion strategies for maintaining visibility of a moving target. In *IEEE Int. Conf. on Robotics and Automation*, volume 1, pages 731–736, april 1997.

[30] S.M. LaValle, J. Hinrichsen. Visibility-based pursuit-evasion: An extension to curved environments. In *Proc IEEE Int. Conf. on Robotics and Automation*, 1999.

[31] S. Lavalle, M.S. Branicky and S.R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. In *Int. Journal of Robotics Research*, 2003.

[32] A. Lazanas and J.C. Latombe. Landmark-based robot navigation. *Algorithmica*, 13:472–501, 1995.

[33] P. Leven and S. Hutchinson. A Framework for real-time path planning in changing environments. In *Int. Journal of Robotics Research*, Vol. 21 No 12, 2003.

[34] R. Murrieta-Cid, M. Briot and N. Vandapel. Landmark identification and tracking in natural environment. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 1998.

[35] R. Murrieta-Cid, C. Parra and M. Devy. Visual Navigation in Natural Environments: From Range and Color Data to a Landmark-based Model. In *Journal Autonomous Robots*, vol. 13, no. 2, pp. 143-168, September 2002.

[36] R. Murrieta-Cid, H.H. González-Baños and B. Tovar. A Reactive Motion Planner to Maintain Visibility of Unpredictable Targets. In *Proc IEEE Int. Conf. on Robotics and Automation*, 2002.

[37] R. Murrieta-Cid, A. Sarmiento and S. Hutchinson. On the Existence of a Strategy to Maintain a Moving Target within the Sensing Range of an Observer Reacting with Delay. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003.

[38] R. Murrieta-Cid, A. Sarmiento, S. Bhattacharya and S. Hutchinson. Maintaining Visibility of a Moving Target at a Fixed Distance: The Case of Observer Bounded Speed. In *IEEE Int. Conf. on Robotics and Automation*, 2004.

[39] J. O'Rourke. *Visibility.* Handbook of Discrete and Computational Geometry, 467-479, J.E. Goodman and J. O'Rourke, 1997.

[40] L. Parker. Algorithms for Multi-Robot Observation of Multiple Targets. In *Journal Autonomous Robots*, vol. 12 pp. 231-255, 2002.

[41] N.P. Papanikolopous, P.K. Khosla, and T. Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. In *IEEE Trans. Robotics and Automation*, 9(1):14-35, February 1993.

[42] T.D. Parsons. Pursuit-evasion in a graph. In *Y. Alani and D.R. Lick, editors, Theory and Application of Graphs*, pages 426-441, Springer-Verlag, Berlin, 1976.

[43] S. Shas, S. Rajko and S.M. LaValle. Visibility-based pursuit-evasion in an unknown planar environment. *Submited to Int. Journal on Robotics Research*, 2003.

[44] J.R. Spletzer and C.J Taylor. Dynamic Sensor Planning and Control for Optimally Tracking Targets. In *Int. Journal of Robotics Research*, Vol. 22 No 1, 2003.

[45] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput*, 21(5):863-888, October 1992.

[46] B. Tovar, R. Murrieta-Cid and C. Esteves. Robot Motion Planning for Map Building. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002.

[47] R. Vidal, O. Shakernia, H. Jin, D. Hyunchul and S. Sastry. Probabilistic Pursuit-Evasion Games: Theory, Implementation, and Experimental Evaluation. In *IEEE Trans. Robotics and Automation*, 18(5):662-669, October, 2002.

[48] E. Welzl. Constructing the visibility graph for n-line segments in $O(n^2)$ time. In *Proceedings of Information Processing Letters*, pages 167–171, 1985.

Rafael Murrieta-Cid received the B.S degree in Physics Engineering (1990), and the M.Sc. degree in Automatic Manufacturing Systems (1993), both from "Instituto Tecnológico y de Estudios Superiores de Monterrey" (ITESM) Campus Monterrey. He received his Ph.D. from the "Institut National Polytechnique" (INP) of Toulouse, France (1998). His Ph.D research was done in the Robotics and Artificial Intelligence group of the LAAS/CNRS. In 1998-1999, he was a postdoctoral researcher in the Computer Science Department at Stanford University. From January 2000 to July 2002 he was an assistant professor in the Electrical Engineering Department at ITESM Campus México City, México. In 2002-2004, he was working as a postdoctoral research associate in the Beckman Institute and Department of Electrical and Computer Engineering of the University of Illinois at Urbana-Champaign. Since August 2004, he is director of the Mechatronics Research Center in the ITESM Campus Estado de México, México. He is mainly interested in sensor-based robot motion planning and computer vision.

Benjamin Tovar received the B.S degree in electrical engineering from ITESM at Mexico City, Mexico, in 2000, and the M.S. in electrical engineering from University of Illinois, Urbana-Champaign, USA, in 2004. Currently (2005) he is pursuing the Ph.D degree in Computer Science at the University of Illinois. Prior to M.S. studies he worked as a research assistant at Mobile Robotics Laboratory at ITESM Mexico City. He is mainly interested in motion planning, visibility-based tasks, and minimal sensing for robotics.

Seth Hutchinson received his Ph. D. from Purdue University in West Lafayette, Indiana in 1988. He spent 1989 as a Visiting Assistant Professor of Electrical Engineering at Purdue University. In 1990 Dr. Hutchinson joined the faculty at the University of Illinois in Urbana-Champaign, where he is currently a Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute for Advanced Science and Technology. Dr. Hutchinson is currently a senior editor of the *IEEE Transactions on Robotics and Automation*. **In 1996 he was a guest editor for a special section of the** *Transactions* **devoted to the topic of visual servo control, and in 1994 he was co-chair of an IEEE Workshop on Visual Servoing. In 1996 and 1998 he co-authored papers that were finalists for the King-Sun Fu Memorial Best Transactions Paper Award. He was co-chair of IEEE Robotics and Automation Society Technical Committee on Computer and Robot Vision from 1992 to 1996, and has served on the program committees for more than thirty conferences related to robotics and computer vision. He has published more than 100 papers on the topics of robotics and computer vision.**