A Reactive Motion Planner to Maintain Visibility of Unpredictable Targets

Rafael Murrieta-Cid* Héctor H. González-Baños† Benjamín Tovar*

* ITESM Campus Ciudad de México
Calle del Puente 222, Tlalpan, México D.F.
{rmurriet,betovar}@campus.ccm.itesm.mx

† Honda's Fundamental Research Labs
800 California St. 300, Mnt. View, CA 94041
hhg@hra.com

Abstract— **This paper deals with the 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 needs to move in order to escape the observer's visibility region. Three main points are discussed: 1) The design and implementation of a reactive planner; 2) the integration and testing of such a planner in a robot system which includes perceptual and control capabilities; 3) The design and simulation of a motion planner for the task of maintaining visibility of two targets using two mobile observers.**

Keywords— **Target tracking, motion planning, visibility constraints, robotics, shortest distance to escape.**

I. Introduction

The target-tracking problem has been traditionally attacked with a combination of vision and control techniques [18], [4]. Purely control approaches, however, do not take into account the complexity of the environment nor the problem of coordinating different robots to track several targets. The basic question which has to 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 have to be taken into account. Thus, a pure visual servoing technique will fail because it ignores the geometry of the workspace.

Maintaining visibility of targets is intimately related to the *art-gallery problem* [17], where the goal is to compute the locations of a minimal number of guards such that all points in the workspace (the art gallery) are visible to at least one guard. In tracking, we are interested in guarding a moving point (the target) using a mobile guard (the observer).

Previous works have studied the motion planning problem for target tracking. The case for predictable targets is presented in [15], 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. The algorithm in [15] relies on a heavy discretization of the environment in order to apply a recursion based on the *dynamic programming principle*. This discretization severely limits the number of degrees of freedom that can be handled by the algorithm.

*This research was partially funded by CONACyT, México

In [1], a tracking algorithm is presented which operates by maximizing the probability of future visibility of the target. This algorithm is also studied with more formalism in [15]. Given a polygonal map of the workspace, the planner first computes a disk of radius V_R centered at q^t, where q^t is the current location of the target and V_R is the maximum speed of the target multiplied by the sampling rate. This disk represents a very crude probability distribution of the target's location. Next, the planner computes the area of the disk that remains visible when the observer position is randomly perturbed around its current location. The planner 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 configuration-space obstacles. This technique was tested in a Nomad 200 mobile robot with relatively good results. However, the probabilistic model assumed by the planner is often too simplistic, and accurate models are difficult to obtain in practice.

The work in [5] presents an approach which takes into account the positioning uncertainty of the robot observer. The game theory is proposed as a framework to formulate the tracking problem. The main 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 re-localize and better track the target afterwards.

In [7], a technique is proposed to track a target without the need of a prior map. Instead, a range sensor is used to contruct 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. This work has yet to be extended to multiple observers and multiple targets.

II. 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, an utility function which 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 total unpredictable targets, it makes more sense to compute short term plans. Unanticipated changes in the target trajectory can be taken into account by re-planning. An on-line strategy computes a motion plan for the next h future stages, and re-plans in the next iteration for the following h future stages. Typically, h is a very small number.

We have designed a reactive planner that expects the target to execute the worst possible move in the next iteration (i.e., $h = 1$). This worst 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 randomized algorithm. We have opted for a randomized approach in order to "break" the complexity of the computation and handle multiple observers and multiple targets. The algorithm samples a number of potential milestones near the observer (its *reachability region*), and selects among them the one that maximizes the SDE.

The quality and success of the generated plans depends significantly on the observer's capabilities, mainly on the size and shape of its reachability region. In this paper, we study this dependence in terms of high-level parameters describing the sensor, such as maximum sensor range and viewing frustum.

A. Algorithmic description

We represent each robot observer and each target with 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. 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 know the specific action θ_k 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 region.

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. 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. Let us 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.

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.

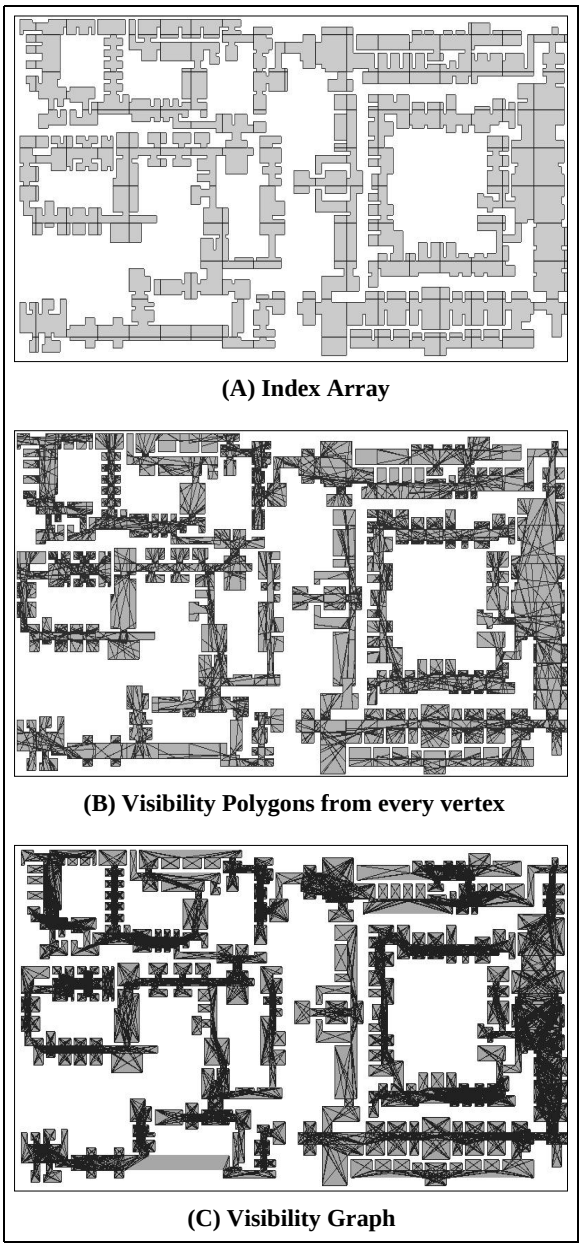
We can identify two main cases for computing the distance between q^t_k and a given free edge E_{fr}. The first one happens when the target can see a given free edge E_{fr}. The computation of this distance is easily done using the Euclidean metric. The second case happens when the E_{fr} is not directly visible from the current location of the target. In this case, we are using a geodesic metric to determine the distance between a given free edge E_{fr} and the target location q^t_k.

The computation of the geodesic distance is done by determining the shortest path between q^t_k and all the free edges in $V(q)$ which are not visible from q^t_k by using the visibility graph of the polygonal map. In order to save time during the execution of the on-line planner, part of this calculation is pre-processed prior to the real experiment.

B. Pre-processing

Since the planner must be capable to run in real time, we exchange time requirements for memory requirements to accelerate execution.

Given a polygonal map of the environment, the precomputation steps are the following: 1) Compute the the visibility region from every vertex in the polygonal map — see Fig. 1(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. 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. 1(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. 4) An approximate *visibility graph* is computed. 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. 1(C). The constant ρ is the



(A) Index Array

(B) Visibility Polygons from every vertex

(C) Visibility Graph

Fig. 1. Pre-proccesing

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.

We now describe how the pre-procomputations are used to reduce the comp. 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 edge must then pass through a vertex of the environment. In 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 this vertices in the list L_v.
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 computed by running Dijkstra's algorithm on this new graph.
3. The distance between q_k^t and each free edge in $V(q)$ is the solution to:

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

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}.

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 pre-computed lookup-table. Similarly, we use the pre-computed visibility grid to determine the list L_v and evaluate $D_{q_k^t/v_o}$. And finally, the visibility region from every vertex of the polygonal map (first step of the pre-computation process) is used to query which vertices in $V(q)$ can see a free edge in order to evaluate $D_{v_f/E_{fr}}$. Thus, the use of pre-computations allows us to accelerate the calculation of $D_{q_k^t/E^*_{fr}}$ during run-time operation.

The smallest geodesic distance $D_{q_k^t/E^*_{fr}}$ among all free edges of $V(q)$ is finally selected as the shortest distance to escape (SDE).

C. General Algorithm

The general algorithm to maintain visibility of an unpredictable target is as follows:

Preprocessing
$I \leftarrow$ COMPUTE_INDEX_ARRAY(Map)
for every bin $b_i \in I$ do
  for every discret point $p \in b_i$ do
    $V(p) \leftarrow$ COMPUTE_VISIBILITY(p, b_i)
    $V_{grid} \leftarrow V_{grid} \cup V(p)$
  COMPUTE_VISIBILITY_GRAPH(Map)

Target Tracking
do
  $V(q_\tau) \leftarrow V_{grid}(q_\tau)$
  $P \leftarrow$ RANDOM_POINTS
  $D_{q_k^{t_1}/E^*_{fr}} \leftarrow 0$
  for every $p \in P$ do
    $V(p) \leftarrow V_{grid}(p)$
    $dist \leftarrow$ MIN_DIST_ESCAPE(t, $V(p)$)
    if $dist > D_{q_k^{t_1}/E^*_{fr}}$ then
      $q_{\tau+1} \leftarrow p$
      $D_{q_k^{t_1}/E^*_{fr}} \leftarrow dist$
  $q_\tau \leftarrow q_{\tau+1}$
while $t \in V(q)$

D. Planner for two-observers/two-robots

We have developed, implemented and simulated a planner for two-observers/two-targets. What is interesting in our approach is that there is no predetermined assignment of a given target to a given observer. At any instant in time, the two observers locate themselves so as to maximize the distance to escape required by either of the targets.

Let us denote $V(q_{k+1}^{o_1})$ as the visibility region of observer 1 at location $q_{k+1}^{o_1}$, and $V(q_{k+1}^{o_2})$ as the visibility region of observer 2 at location $q_{k+1}^{o_2}$. The SDE for a target tracked by two observers is defined as the shortest distance between q_k^t and the boundary of the union of the visibility regions $V(q_{k+1}^{o_1}) \cup V(q_{k+1}^{o_2})$.

We use $\max(U)$ as a criteria to select the future positions $q_{k+1}^{o_1}$ and $q_{k+1}^{o_2}$ for both observers, with U defined as:

$$U = \left[\frac{1}{2}\left(\frac{d_1}{w_1}\right)^{-p} + \frac{1}{2}\left(\frac{d_2}{w_2}\right)^{-p}\right]^{-\frac{1}{p}}, \quad (2)$$

$$\text{with } d_1 = D_{q_k^{t_1}/E_{fr}^*} \text{ and } d_2 = D_{q_k^{t_2}/E_{fr}^*}, \quad (3)$$

where $q_k^{t_1}$ and $q_k^{t_2}$ are the positions of target 1 and target 2, respectively, and d_1 and d_2 denote the distances between the two targets and the closest free edge belonging to the boundary of $V(q_{k+1}^{o_1}) \cup V(q_{k+1}^{o_2})$. Here w_1, w_2 and p are positive constants.

The constants w_1 and w_2 weight the relative importance of keeping track of each target. U can take different forms according to the value of p: (1) $U = \min(d_1, d_2)$ when p tends to ∞ (assuming $w_1 = w_2 = 1$); (2) $U = \sqrt{d_1 d_2}$ when p tends to 0. The first case was tested in simulations and behaved as expected (good performance). An oscillation in the observers, however, appears if the targets exchange positions. The criteria for $p \to 0$ produced better results (second case), and less oscillations occurred when the targets exchanged positions.

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. Currently, we are working on a number of cases where 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 still rather than assisting the first observer.

III. Implementation/Experimentation

Our tracking system is implemented on a SuperScout mobile robot from Nomadic Technologies. The SuperScout 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.

A. Architecture

Our target-tracking software incorporates 5 main modules: (1) A frame server, (2) a visual target detector and motion camera 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. Both cameras are synchronized with each other and the images are combined into a single RGB image by a junction box. 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 servoing problem has received considerable attention in the computer vision community over the last years. Several techniques have been reported in the literature, and a variety of algorithms have been proposed for visual servoing [3], [10].

Our visual target detection uses a very simple and fast vision algorithm. A cylindrical mobile robot (a Nomad 200) acts as target, and un-obtrusive rectangular patterns are placed on its hull. Each pattern has a binary bar-code identifier. The algorithm computes sub-pixel image positions of the pattern's corners to estimates its 3-D pose [11]. With the pose and bar-code information of the detected patterns, the algorithm then infers the location and bearing of the target

Although our detection algorithm is very simple, we could instead use more advanced tracking algorithms such as those described in [9], [18], [16].

The range of the target detection module is approximately 80 in. 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 are using 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 [12], [13], [14], [19]. The landmark detection module on-board our observer is the result of the work developed in [2]. 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 re-localize at a high frequency rate, the landmark detector runs at 0.5-1 Hz.

The global planning algorithm is the one described in Section II. The output of this planner is sent to a motion controller that drives the robot observer to the goal provided by the planner. The robot executes a plan by first aiming to the goal and then translating to the goal. In general, it is not possible to execute a

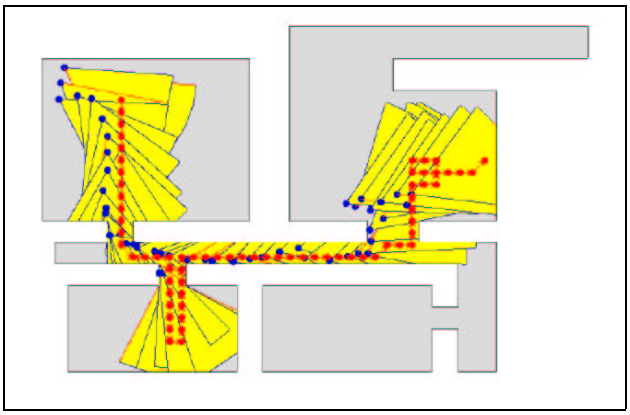


Fig. 2. Simulation using the map of the Stanford's Robotics Laboratory.

plan in just one step with a nonholonomic robot. However, it is possible to stabilize two of the configuration variables (x, y) or (x, θ) [20].

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 rates [8]. 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 in an off-board computer. This allow 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 on-board the robot. The planner runs on a separate Celeron 600 MHz computer

B. 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) 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.

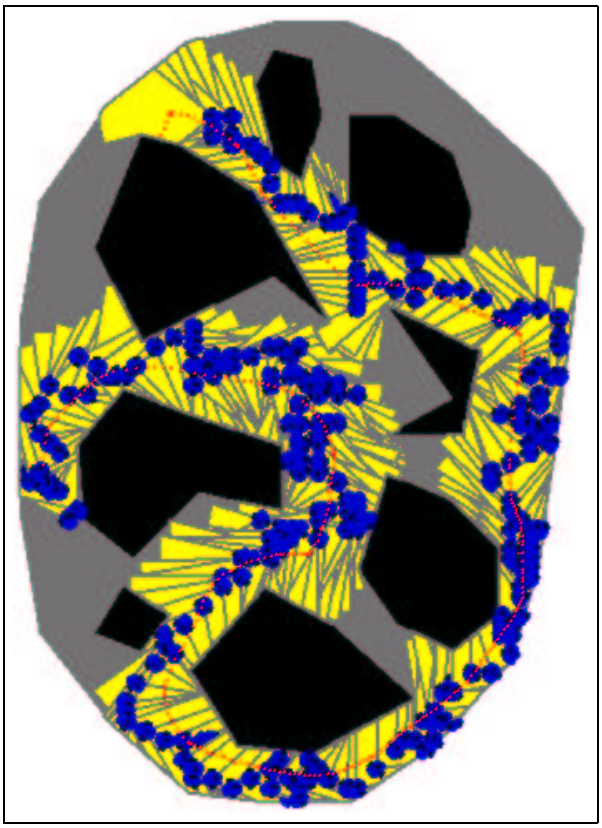


Fig. 3. An example of an environment with holes.

Figure III-B shows the result of a simulation for one-robot/one-target in a typical environment. The blue point represents the observer and red point represents the target. The yellow region represents the observer's visibility region. Figure III-B shows a more complicated example consisting of an environment composed of 48 edges and 8 holes. The target is shown with a red disk and the observer with a blue (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 4 (next page) 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 red disk and the observer with a blue (bigger) disk. 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 pre-computation process took about 13 minutes. At each iteration during run-time, the planner generated 75 random samples in a neighborhood around q^o_k in order to compute the next observer position. By using pre-computations, the planner can compute q^o_{k+1} with a freq. of 13 Hz. Without the pre-computation the sole visibility region computation with 1400 vertices takes about 3 seconds per random sample.

Fig. 4. Large Map

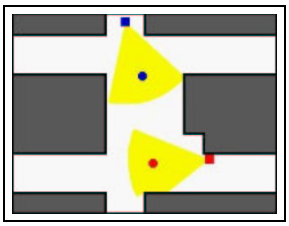


Fig. 5.

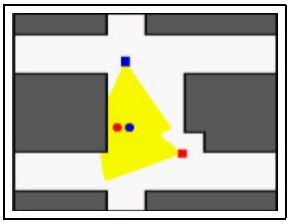


Fig. 6.

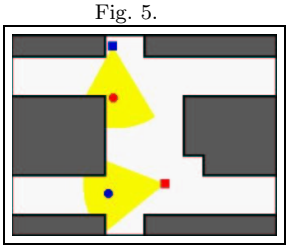


Fig. 7.

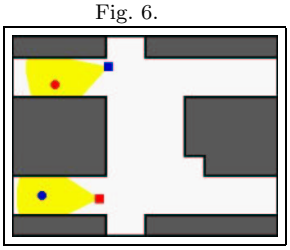


Fig. 8.

Fig. 9.

Fig. 10.

Fig. 11.

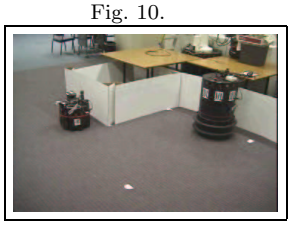


Fig. 12.

Figures 5, 6, 7 and 8 show a simulation experiment with two-robots/two-targets. It is possible to see how the observers switch targets. At first, observer 1 (red square) tracks target 1 (red disk) and observer 2 (blue square) tracks target 2 (blue disk). Once the targets cross each other the switch is done, observer 1 tracks target 2 and observer 2 tracks target 1.

Figures 9, 10, 11 and 12, present 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.

Figures 13, 14, 15 and 16, show 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

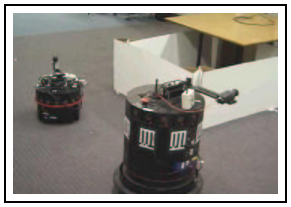


Fig. 13.

Fig. 14.

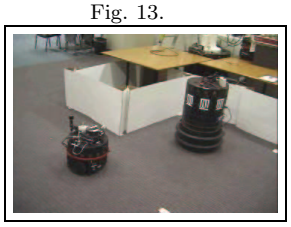


Fig. 15.

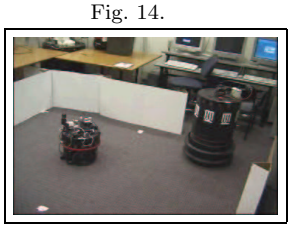


Fig. 16.

robot observer is capable of tracking the target even when it turns around a corner. The experiment shown in Figures 13-16 were done under the same conditions as that shown in Figures 9-12. 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.

Failures

Often 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.

IV. Discussion and future work

In this paper we have argued that visual-servoing approaches to target tracking 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 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.

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.

For the case of multiple robots and multiple targets, an interesting topic for future research 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 on many observers, such decentralized planner will be preferable over a centralized approach.

Acknowledgements We want to thank Jean-Claude Latombe for his advice and suggestions on the implementation of our algorithms, and Cheng-Yu Lee for his help in the development of the robotic system. This research was partially funded by CONACyT, México.

References

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

[2] C. Becker, J. Salas, K. Tokusei, and J.C. Latombe, Reliable navigation using landmarks. *IEEE Int. Conf. on Robotics and Automation*, pages 401-406, May 1995.

[3] P. Delagnes, J. Benois and D. Barba, Adjustable polygons: a novel active contour model for objects tracking on complex background. *Journal on communications*, 8(6), 1994.

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

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

[6] H.H. González Baños E. Mao, J.-C. Latombe, T.M. Murali and Alon Efrat, Planning robot motion strategies for efficient model construction *In Robotics Research - The 9th Int. Symp*, 1999.

[7] H.H. González-Baños, C.-Y. Lee and J.-C. Latombe, Motion Strategies for Maintaining Visibility of a Moving Target *In Proc IEEE Int. Conf. on Robotics and Automation*, 2002.

[8] H.H. González Baños, J. L. Gordillo, D. Lin, J.-C. Latombe, A. Sarmiento and C. Tomasi, The Autonomous Observer: A Tool for Remote Experimentation in Robotics *SPIE Intl. Symp. on Voice, Video, and Data Communications*, 1999.

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

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

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

[12] S. Hutchinson, Exploiting visual constraints in robot motion planning, *In Proc IEEE Int. Conf. on Robotics and Automation* ,pages 1722-1727, 1991.

[13] 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.

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

[15] S.M. LaValle, H.H. González-Baños,C. Becker and J.C. Latombe, Motion Strategies for Maintaining Visibility of a Moving Target *In Proc IEEE Int. Conf. on Robotics and Automation*, 1997.

[16] R. Murrieta-Cid, M. Briot, N. Vandapel, Landmark identification and tracking in natural environment *In Proc. Int. Conf on Intelligent Robots and Systmens*, 1998.

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

[18] 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, *IEEE Trans. Robotics and Automation*, 9(1):14-35, February 1993.

[19] C. Parra, R. Murrieta-Cid, M. Devy, and M. Briot, 3-D Modelling and Robot Localization from Visual and Range Data in Natural Scenes. *In Proc First Int. Conf on Vision Systems*. Henry I. Christense Ed. January 1999.

[20] C. Samson and K. Ait-Abderrahim, Feedback control of a nonholonomic wheeled cart in cartesian space. *Technical Report 1228 INRIA, France*. Octuber 1990.