Motion Strategies for Maintaining Visibility of a Moving Target

Steven M. LaValle Héctor H. González-Baños Craig Becker Jean-Claude Latombe

Computer Science Department
Stanford University
Stanford, CA 94305
{lavalle,hhg,cdb,latombe}@robotics.stanford.edu

Abstract

We introduce the problem of computing robot motion strategies that maintain visibility of a moving target in a cluttered workspace. Both motion constraints (as considered in standard motion planning) and visibility constraints (as considered in visual tracking) must be satisfied. Additional criteria, such as the total distance traveled, can be optimized. The general problem is divided into two categories, on the basis of whether the target is predictable. For the predictable case, an algorithm that computes optimal, numerical solutions is presented. For the more challenging case of a partially-predictable target, two on-line algorithms are presented that each attempt to maintain future visibility with limited prediction. One strategy maximizes the probability that the target will remain in view in a subsequent time step, and the other maximizes the minimum time in which the target could escape the visibility region. We additionally discuss issues resulting from our implementation and experiments on a mobile robot system.

1 Introduction

Several applications require persistent monitoring of a moving target by a controllable vision system. In applications that involve automated processes that need to be monitored, such as in an assembly workcell, parts or subassemblies might need to be verified for accuracy or are determined to be in correct configurations. Visual monitoring tasks are also suitable for mobile robot applications [3]. In medical applications, one would like to move cameras around a surgery site to keep a designated area of interest (key tissue) in continuous sight, despite unpredictable motions of potentially obstructing people and instruments, and display a smooth sequence of images for the surgeon [12]. In a telepresence or virtual presence application, a vision system can be used in a remote location to automatically track a variety of moving objects such as vehicles, people, or other robots. Visual information can also be used to track robots or robot features that appear in an image, and be directly integrated into a servo loop (e.g., [5, 6, 14]).

A motion planning problem is considered in this paper in which a robot carries a camera that must maintain visibility of a target. The primary distinction between the problem considered in this paper and standard tracking problems is the introduction of global, geometric constraints on both visibility and robot configurations. The following conditions are assumed: 1) an *observer* must maintain visibility of a moving *target*; 2) the workspace contains static obstacles that prohibit certain configurations of both the observer and target; 3) the workspace also contains static obstacles that occlude the target from the observer; 4) a (possibly partial) motion model is known for the target. The first condition implies that target tracking is the primary interest, and visibility can be defined in a variety of ways, depending on the particular problem. The second condition introduces the geometric constraints that appear in the standard path planning problem [10]. The third condition complicates the tracking problem by prohibiting *pairs* of observer and target configurations at which the observer cannot "see" the target. In many cases an obstacle in the workspace will cause both motion and visibility constraints. The fourth condition provides predictive information that should be utilized when designing a strategy. For example, the entire trajectory of the target might be known, or alternatively, only a velocity bound might be known. In addition to the previous four conditions, it might also be important to optimize some criteria such as the total distance traveled, energy utilized by the observer, or the quality of the visual information.

Section 2 provides a precise formulation of the problem in terms of configuration space concepts and system theory concepts. The computation methods presented in this paper are divided into two sections on the basis of target predictability. Section 3 presents an off-line algorithm that determines optimal, numerical solutions for the case in which the target is completely predictable (i.e., the trajectory is known). The case in which the target is only partially-predictable is considerably more difficult, and is covered in Section 4. For this case, two different algorithms are presented that make on-line decisions that attempt to maintain future visibility. Both of these algorithms can be used in a real-time application using on-line information, and experimental results using two Nomad 200 robots are presented. Conclusions and discussion appear in Section 5.

2 Problem Formulation

Suppose that an *observer* and a *target* exist in a bounded, Euclidean workspace that is cluttered with static obstacles. In general, the observer and target can be considered as rigid or articulated bodies, with

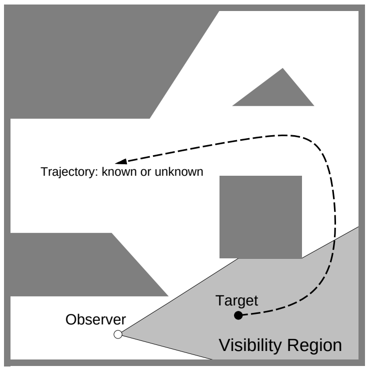


Figure 1. The goal is to maintain visibility of a moving target that may or may not be predictable.

standard configuration-space parameterizations [10]. Let $\mathcal{C}^o_{free}$ and $\mathcal{C}^t_{free}$ denote the free configuration spaces of the observer and target, respectively. Let $X = \mathcal{C}^o_{free} \times \mathcal{C}^t_{free}$ represent the *state space.* A state simultaneously specifies configurations for the observer and target.

Motion models will next be formulated for the observer and the target. Discrete-time representations will be used to facilitate the expressions when there is uncertainty in target prediction; however, continuous-time representations could alternatively be used. Let the index k refer to the stage or time step that occurs at time $(k-1)\Delta t$, for some fixed, Δt (which specifies the sampling rate). Let $\mathbf{q}^o_k$ and $\mathbf{q}^t_k$ denote specific configurations at stage k, for the observer and target, respectively.

The observer will be controlled through actions, u_k, chosen from some action space U. The discrete-time trajectory will be given by a transition equation of the form $\mathbf{q}^o_{k+1} = f^o(\mathbf{q}^o_k, u_k)$, which yields a new configuration of the observer for a given current configuration and action. Constraints that include nonholonomy and bounded velocity can be modeled using f^o.

The target will be described by a similar transition equation; however, the actions that control the target are generally unknown to the observer. Let $\mathbf{q}^t_{k+1} = f^t(\mathbf{q}^t_k, \theta_k)$, in which θ_k represents unknown actions, chosen from some space Θ. One important special case, which is the subject of Section 3, is when the target is *predictable.* In this case the transition equation can be represented as $\mathbf{q}^t_{k+1} = f^t(\mathbf{q}^t_k)$.

Together, f^o and f^t define a state transition equation of the form $x_{k+1} = f(x_k, u_k, \theta_k)$.

Recall that each state, x_k, represents a pair of configurations, $\mathbf{q}^o_k$ and $\mathbf{q}^t_k$. A binary relation on these configuration pairs can be defined that declares whether the target is visible to the observer. This visibility can be defined in a number of ways (e.g., omnidirectional field of view, fixed cone, etc.). Let $X_o \subset X$ represent the *visibility subspace*, which corresponds to the set of all states for which the visibility relation holds.

Next consider evaluating a state trajectory. Abstractly, the goal is to control the observer to ensure that the state remains in X_o. The cost of applying a sequence of control inputs is state trajectory can be expressed as

$$L(x_1, \ldots, x_{K+1}, u_1, \ldots, u_K) = \sum_{k=1}^{K} l_k(x_k, u_k) + L_{K+1}(x_{K+1}), \tag{1}$$

in which K represents the final time increment for issuing a action and $l_k(x_k, u_k)$ is a loss that accumulates in a single time step. The final state, x_{K+1} can also be penalized, using L_{K+1}.

A simple, useful form of l_k is

$$l_k(x_k, u_k) = \{0 \text{ if } x_k \in X_o;\ 1 \text{ otherwise}\} \tag{2}$$

This loss functional measures the amount of time that the target is not visible. One could also include a cost for choosing actions that produce motion. This would allow the robot motions to be optimized in addition to the time that the target is in view, and is considered in Section 3.

The loss functional evaluates a given trajectory. In the case of perfect target predictability, the trajectory can be inferred once the actions, $\{u_1, \ldots, u_K\}$ are specified. In the case of a partially-predictable target, the loss functional is used to evaluate a state-feedback strategy using expected-case or worst-case analysis. These concepts are deferred until Section 4.

3 Predictable Targets

In this section the assumption is made that $\mathbf{q}^t_k$ is known for all $k \in \{1, \ldots, K+1\}$. In this case, the state transition equation reduces to $x_{k+1} = f(x_k, u_k)$, which implies that the state trajectory, $\{x_2, \ldots, x_{K+1}\}$ is known once x_1 and inputs $\{u_1, \ldots, u_K\}$ are given.

For a problem that does not involve optimizing the robot trajectory, motions for the observer could be determined by recursively computing visibility and reachability sets from stage $K+1$ down to stage 1. Suppose the target and observer are both points. Let V_k denote the subset of the free space from which the target is visible. Let A_{k_1} denote the set of all locations from which the observer could move at stage $k-1$ and lie in V_k at stage k. At any given stage, the observer must lie in $V_k \cap A_k$. A feasible trajectory for the observer can be obtained by backchaining from the final stage, guaranteeing accessibility and visibility in each step, until a set of possible initial states is obtained.

In the remainder of this section, a method that employs the dynamic programming principle to minimize a specific loss functional of the form (1) is presented. Although the approach shares some similarities with Dijkstra's algorithm on graphs, the principle is applied in the present context over a continuous state space and over discrete time. Its use follows directly from the differential equations that express the dynamic programming principle in standard optimal control [9, 11].

3.1 Computational Approach

The computational approach can be organized into four basic steps:

1. Construct a discretized representation of $\mathcal{C}^o_{free} \times \mathbf{K}$, in which $\mathbf{K} = \{1, \ldots, K+1\}$.

2. For each k, mark all discretized values of $\mathbf{q}_k^o$ from which the target (at known $\mathbf{q}_k^t$) is visible.

3. Within X_o for each stage from $K+1$ to 1 perform dynamic programming computations with interpolation.

4. Extract the optimal sequence, $\{u_1^*, \ldots, u_K^*\}$, using *cost-to-go* representations.

Step 1: Because the target is predictable and parameterized through the stage index k, the subspace corresponding to $\mathcal{C}^o_{free}$ only needs to be considered as opposed to the entire state space. The stage index is included because the problem is time-varying. This is similar to the use of configuration-time space representations for motion planning among known moving obstacles [7, 10]. An array representation of the spaces is constructed, which ultimately limits the current approach to observers that have only a few degrees of freedom. In [8], it is shown that the Fast Fourier Transform can be used to efficiently obtain a C-space representation from the static obstacles and robot geometry.

Step 2: Since the primary task is to maintain visibility of the target, the acceptable observer locations are marked. Using omnidirectional visibility, this can be accomplished efficiently by performing scan conversion of a computed visibility polygon that emanates from the target. Using a standard sweep algorithm [13], the visibility polygon can be computed in $O(n \lg n)$ time. If a loss functional is defined that evaluates individual viewpoints then one would precompute real-valued costs, as opposed to binary flags. For example, one might indicate a preference for a certain distance between the target and observer that is reflected in the real-valued costs.

Step 3: The dynamic programming computations are the most significant portion of the computation. For each stage k a *cost-to-go* function, $L_k^* : X \rightarrow \Re$, is computed using the cost-to-go function of stage $k+1$. The cost-to-go function represents the loss or cost that will be ultimately accumulated by starting from configuration $\mathbf{q}_k^o$ and choosing the optimal action at each stage. Due to the dependencies between the cost-to-go functions, L_{K+1}^* is computed initially, and a cost-to-go function is computed for each prior stage until L_1^* is finished. Although the domain of the cost-to-go function is X, computations only need to be performed on $\mathcal{C}^o_{free}$ because the target configuration is fixed for each stage.

The dynamic programming principle defines the relationship between the cost-to-go functions:

$$L_k^*(x_k) = \min_{u_k} \{ l_k(x_k, u_k) + L_k^*(x_{k+1}) \} , \qquad (3)$$

in which l_k is defined in the loss functional (1), and x_{k+1} is obtained for each choice of u_k through the state transition equation, $x_{k+1} = f(x_k, u_k)$. The difference equation (3) defines a very local relationship between successive cost-to-go functions, yet the principle of optimality ensures that a globally optimal strategy will result.

One difficulty results from using discretized representations of the continuous dynamic programming principle. The next state, x_{k+1}, will usually not lie exactly at a discretized value. Rather than simply looking up L_{k_1} at the nearest quantized value, the value of L_{k+1} can be computed through linear interpolation of the cost-to-go values between all neighbors of x_{k+1}. This technique has been used previously in numerical dynamic programming computations; related issues are discussed in [9, 11].

Step 4: Suppose the cost-to-go functions have been computed. If the $\mathbf{q}_1^o$ is fixed, then u_1^* can be obtained by using (3) for the given initial state, x_1. If $\mathbf{q}_1^o$ is free, then an optimal initial configuration can be obtained by selecting the configuration $\mathbf{q}_1^o$ minimizes L_1^*. Once u_1^* has been determined, the next state $x_2 = f(x_1, u_1^*)$ can be inferred. Equation (3) can be used again to determine u_2^*. This process iterates until the final, optimal action u_K^* is obtained.

The time complexity of the method is linear in the number of actions and stages, and exponential in the dimension of the observer configuration space.

3.2 Computed Examples

The optimal strategies were computed for several strategies, and the resulting observer trajectories were simulated. Some of these results are shown in this section to demonstrate the method and to illustrate the global aspects of this motion planning problem. In all of the examples, the workspace is 2-D with dimensions 100 units by 100 units. All obstacles obstruct both visibility and motion. The target moves at its maximum speed of 3 units/stage. The observer is controlled through the simple holonomic model,

$$\mathbf{q}_{k+1}^o = \mathbf{q}_k^o + u_k^1 \Delta t \, [cos(u_k^2) \quad sin(u_k^2)]^T , \qquad (4)$$

in which $u_k^1 \geq 0$ represents the speed of the observer, and $u_k^2 \in [0, 2\pi)$ represents the direction of motion. No dynamics are considered in this model.

Figure 2 shows a simulation of a trajectory that is obtained from the computed optimal strategy. The actions $u_k^1 = 0$ (no motion) and $u_k^1 = 3$ and $u_k^2 \in [0, 2\pi)$ (motion at a fixed speed in some direction) were allowed. The loss functional is of the form $l_k(x_k, u_k) = l_m + l_v$. The term l_m is a penalty for motion, and $l_m = 0$ if $u_k^1 = 0$, otherwise $l_m = 1$. The term l_v is a penalty for losing visibility, which is much more important, yielding $l_v = 500$. There are 105 stages for this problem, and the dynamic programming computations took about 20 seconds on a SPARC 20 workstation. Note that although the target trajectory is quite long, the distance traveled by the observer is short. An initial position for the observer was automatically selected from which the target was visible during the first portion of the trajectory. The observer moves just barely far enough to the lower left, and then finishes by remaining in the lower right for the final segment of the target trajectory.

The example in Figure 3a involves the same geometry; however, the loss functional is slightly changed to yield $l_v = 0$ only if the target is both visible and the distance between the target and observer lies within 10 and 25 units. Also, the speed, u_k^1, is allowed to vary between 0 and 3, with a loss, l_m, that is proportional to the speed. In this case, the observer must travel a greater total distance, yet an optimized trajectory that maintains visibility is still obtained.

Figure 3b shows an example in which the maximum observer speed is 1.5, and the target speed is 3.0. There

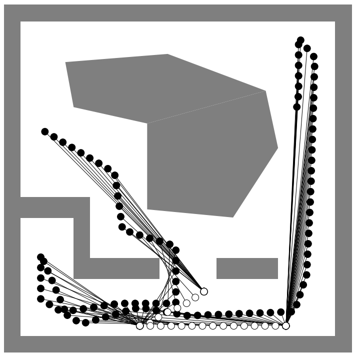


Figure 2. A simulation is shown of the optimal state trajectory for a tracking problem. The observer strategy, including the initial position, is chosen to maintain visibility and minimize the total distance traveled. The target positions are shown as black circles, and the target trajectory starts in the upper left. The observer positions are shown with white circles. The line-of-sight is shown between the observer and target at each stage.

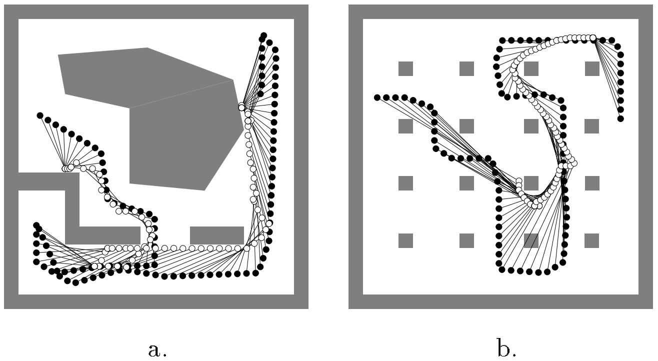
are many visual obstructions, and the observer is able to maintain visibility of the target in all but two stages. During this period, the observer moves quickly to the right to reacquire the target.

4 Partially-Predictable Targets

The problem in Section 3 allowed restricting the state space to the observer configuration space because of target predictability. In this section it is assumed that only weak information, such as a velocity bound, is known regarding the target. In principle, a dynamic programming approach can be taken to determine optimal strategies for the partially-predictable case; however, even for a simple planar problem the state space is four-dimensional. This increased complexity motivates the consideration of alternative approaches which can provide reasonable behavior by making a tradeoff between computational cost and the quality of the solution. Experiments with an on-line algorithm that is presented in this section were performed using a mobile robot system that is described in [2].

4.1 Optimal Strategies

The notions of a strategy and of optimality become more interesting if there is uncertainty in target prediction. Recall that $\theta_k \in \Theta$ refers to an unknown action that can be applied to move the target using $\mathbf{q}^t_{k+1} = f^t(\mathbf{q}^t_k, \theta_k)$. Two alternative interpretations of the unknown actions are possible. If the unknown actions are modeled as *nondeterministic uncertainty* (as referred to in [4, 11]), then it is only assumed that $\theta_k \in \Theta$ for some specified Θ. In this case, one would design a strategy that performs the best given the worst-case choices for θ_k. Alternatively, a *probabilistic uncertainty* model can be used, in which it is additionally assumed that $p(\theta_k)$ is given, in which $p(\cdot)$ denotes a probability density function. In this case, one could design a strategy that minimizes the loss in the *expected* sense.



a. b.

Figure 3. a) This example differs from the previous one by additionally requiring that the observer remains within a specified distance range from the target; b) In this example, the target can moves at twice the maximum speed of the observer. In the optimal trajectory of the observer, there are only two stages in which the target is not visible.

Because the state trajectory cannot be predicted, a state-feedback strategy is designed, as opposed to directly specifying the actions (the actions must respond to on-line changes). This represents a standard notion used in optimal control, and has been applied to motion planning problems that involve other forms of prediction uncertainty in [11]. Let $\gamma_k : X \rightarrow U$ denote a *strategy at stage k*. Let $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_K\}$ denote a *strategy*. Let Γ denote the space of possible strategies for the robot.

For the case of nondeterministic uncertainty, a strategy, $\gamma^* \in \Gamma$, can be selected that yields the smallest worst-case loss:

$$\check{L}(x_1, \gamma^*) = \inf_{\gamma \in \Gamma} \check{L}(x_1, \gamma) = \inf_{\gamma \in \Gamma} \sup_{\gamma^\theta \in \Gamma^\theta} L(x_1, \gamma, \gamma^\theta) \quad (5)$$

for all $x_1 \in X$, and γ^θ represents a choice of θ_k for every stage. This indicates that from any initial state, the strategy will guarantee the least possible loss given the worst-case actions of nature. This concept has been used previously to design controllers based on worst-case analysis [1].

With probabilistic uncertainty, a strategy, $\gamma^* \in \Gamma$, can be chosen that minimizes the expected loss:

$$\bar{L}(x_1, \gamma^*) = \inf_{\gamma \in \Gamma} \bar{L}(x_1, \gamma) = \inf_{\gamma \in \Gamma} \int L(x_1, \gamma, \theta) p(\theta) d\theta \quad (6)$$

for all $x_1 \in X$. This corresponds to selecting a strategy that minimizes the loss in the expected sense, as considered in stochastic optimal control theory.

These expressions capture the general design problem; however, for most practical problems, it may be preferable to consider simplified strategies. Sections 4.2 and 4.3 describe two on-line approaches that attempt to optimize local criteria that are related to the global task.

4.2 Maximizing the Probability of Future Visibility

For any given state, x_k, we could compute the sequence of future actions that will maximize the probability that the target will remain in view over the next m

states. The computational cost, however, increases dramatically with large choices of m. A strategy that has worked in practice is to limit the scope at $m = 1$, and select the action u_k that will maximize the probability that the target will remain in view at stage $k + 1$.

This approach assumes that a probabilistic uncertainty model can be obtained in order to predict the target's motion. Hence, it is assumed that $p(\theta)$ is given (where θ represents the unknown actions of the target), from which a density $p(\mathbf{q}^t_{k+1}|x_k)$ can be obtained using the motion model for the target. Formally, $u^*_k \in U$ is selected to maximize

$$P[\mathbf{q}^t_{k+1} \in V(\mathbf{q}^o_{k+1})] = \int_{X_o} p(x_{k+1}|x_k, u_k) dx_{k+1}, \quad (7)$$

in which $V(\mathbf{q}^o)$ represents the set of target configurations at which the target is visible when the observer is at $\mathbf{q}^o$. This defines a state-feedback strategy $\gamma(x_k) = u_k$.

Our first implementation of this approach assumes a planar workspace for both the target and the observer, and that the obsever has omnidirectional visual sensing. This is justified in practice if an independent servo exists that keeps the camera orientation aimed at the target. We also assume that little is known about the target other than its maximum speed, $||v^t||$, so we model $p(\mathbf{q}^t_{k+1}|x_k)$ as a disk of uniform probability density with radius $||v^t||$, centered at $\mathbf{q}^t_k$. Geometric constraints are incorporated by setting zero probability mass in the regions within the disk that correspond to configuration-space obstacles.

Because the approach involves a significant tradeoff from optimality, experimental studies were performed to assess its utility. We have performed numerous experiments using two Nomad 200 mobile robots, one of which is equipped with a vision system (see Figure 4). Visibility is assumed to be omnidirectional, but the range is bounded below and above to reflect physical limitations in the vision system.

In our experimental setup, the planner is integrated into a larger system composed of several components engaged in landmark-based localization, motion and camera control, and image feature tracking. The current implementation of the system consists of eight modules: a visual target tracker, a landmark detector, the motion planner, a graphic interface, a calibration module, the control module, and a low-level control driver. For more specific details see [2].

We have also studied the observer behavior in a variety of simulated environments. A sequence of frames from one such simulation is shown in Figure 5.

Performance can generally be improved by incorporating additional information into the prediction. For example, the current heading of the target can be utilized to provide more realistic expressions of $p(\mathbf{q}^t_{k+1}|x_k)$. A dynamic model of the target could be utilized to improve prediction for on-line strategies.

For nonreactive targets the motion prediction problem is completely independent from the motion planning one. The target future position distribution may be computed numerically and in some cases analytically. We have studied both approaches for the case of a *Cartesian* target, in which independent actuators drive the

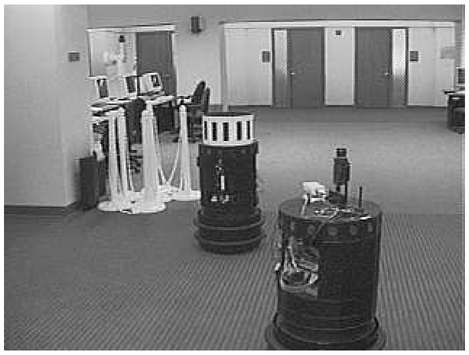


Figure 4. Two Nomad 200 robots are used for the experiments. One is equipped with a rotary camera.

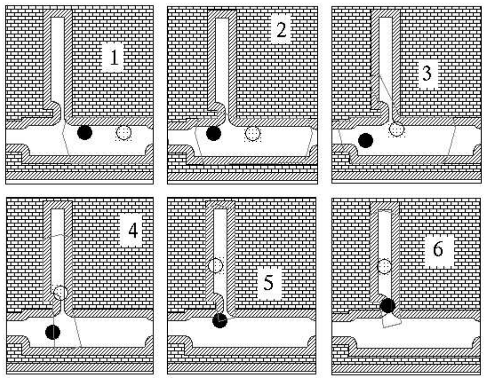


Figure 5. A sample execution is shown of the on-line planner. The black disk is the observer.

target along different orthogonal motions. We assume that the target decides a new acceleration action every time τ, which will be the *target's sampling rate* (not to be confused with Δt, the *planner's scope*). The values of acceleration will be selected from a set $\mathcal{A}$, which is the *acceleration possibility set*. We presuppose that the acceleration values at each step are i.i.d. random variables.

Typically the planner scope Δt is larger than the target sampling rate τ, hence we are interested in predicting the probable target locations after $n = \Delta t/\tau$ steps given a measurement of the target configuration at stage k. The prediction is then used to compute the observer location that maximizes the probability of observation at stage $k + 1$. If n is small and $\mathcal{A}$ is a finite and countable set then the distribution may be computed numerically by evaluating the possible sequences of actions, computing the resultant final configurations, and storing the associated probabilities in a data structure.

If the size of $\mathcal{A}$ is p (the number of alternatives) then after n steps there are p^n possible target configurations, so the search space is exponential. However, the discretized equations of motion can be formulated in such way that most of the operations can be precomputed. Execution can also be improved by detecting equivalent sequences of actions in order to transform the search tree into a lattice.

For n large enough ($\gg 30$) the resultant distribution can be proved to be Gaussian. If the possible accelerations are i.i.d. with mean μ and variance σ^2, then the location of the Cartesian target after n steps given an initial position and velocity x_0 and v_0 is described by $x_n = x_0 + n\tau v_0 + S_v + S_a$, with S_v being a Gaussian distribution of mean $\frac{1}{2}\Delta t(\Delta t - \tau)\mu$ and variance $\frac{1}{6}\Delta t\tau(\Delta t - \tau)(2\Delta t - \tau)\sigma^2$, and S_a a Gaussian with mean $\frac{1}{2}\Delta t\tau\mu$ and variance $\frac{1}{4}\Delta t\tau^3\sigma^2$. Currently our efforts concerning target modeling are aimed toward the study of the effect of how obstacles distort the probability distribution of the target motions under this model.

4.3 Maximizing the Time to Escape

This section describes an alternative on-line approach that uses nondeterministic uncertainty and worst-case analysis as opposed to probabilistic analysis. Assume that the target has a maximum speed, $||v^t||$. Let $d(x_k)$, denote the distance from $\mathbf{q}^t$ to the nearest boundary of $V(\mathbf{q}^o)$. The *minimum time to escape*, t_{esc}, represents the smallest interval of time within which the target could escape if the observer remains at $\mathbf{q}^o$. Clearly, $t_{esc} \leq ||v^t||d(x_k)$.

Assume that $V(\mathbf{q}^o)$ is a truncated cone, as in Section 4.2. The on-line strategy is to select an action $u_k = \gamma_k(x_k)$ that maximizes t_{esc}. This corresponds to preparing for the worst-case motions of the target. Typically, $\Delta t \ll t_{esc}$, which implies that the strategy effectively considers target actions that are several stages into the future. This strategy appears to be an improvement over maximizing the probability of future visibility; however, it is still not a globally optimal solution.

This approach is particularly attractive because by maximizing the t_{esc} we also reduce the observer's susceptibility to uncertain and spurious sensor readings from its cameras. Since this strategy is a worst-case scenario analysis, the system will be robust to uncertainties in the target location. We are currently in the process of experimenting with this strategy to evaluate its performance.

5 Discussion

A research problem that involves maintaining visibility of a moving target has been identified and formally characterized. Workspace geometry introduces standard motion planning difficulties into the visual tracking problem. For the case of predictable targets, an algorithm was presented that provides numerical, optimal solutions for problems that have a low-dimensional observer configuration space. For the case of partially-predictable targets, optimal strategies were characterized, and two on-line strategies were presented. The strategy that maximizes the probability of future visibility has been tested in experimentation and simulation, and the strategy that maximizes the minimum time to escape is currently being evaluated.

Several interesting extensions and variations can be considered in future investigations. In future investigations it may become useful to consider approximation algorithms that can provide a solution that is within some bound of the optimal in order to establish definite limits on a system performance. Many interesting possibilities exist for coordinating multiple observers and/or multiple targets. For example, several observers may be able to track a faster-moving target by making decisions to "cover" disparate regions of the cluttered environment. Another interesting extension is tracking a *reactive* target that attempts to avoid being observed. Finally, the intentions of the target could be speculated by the observer in order to narrow the number of possible target routes.

Acknowledgments

We thank David Lin for his helpful suggestions and contributions to the experimental mobile robot system. This work is supported by NSF grant IRI-9506064, ARO MURI grant DAAH04-96-1-007, ONR grant N00014-94-1-0721.

References

[1] T. Başar and P. R. Kumar. On worst case design strategies. *Comput. Math. Applic.*, 13(1-3):239–245, 1987.

[2] C. Becker, H. González-Baños, J.-C. Latombe, and C. Tomasi. An intelligent observer. In *Preprints of International Symposium on Experimental Robotics*, pages 94–99, 1995.

[3] A. J. Briggs and B. R. Donald. Robust geometric algorithms for sensor planning. In J.-P. Laumond and M. Overmars, editors, *Proc. 2nd Workshop on Algorithmic Foundations of Robotics.* A.K. Peters, Wellesley, MA, 1996. To appear.

[4] M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.

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

[6] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Trans. Robot. & Autom.*, 12(5):651–670, October 1996.

[7] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *Int. J. Robot. Res.*, 5(3):72–89, 1986.

[8] L. E. Kavraki. Computation of configuration-space obstacles using the Fast Fourier Transform. *IEEE Trans. Robot. & Autom.*, 11(3):408–413, 1995.

[9] R. E. Larson and J. L. Casti. *Principles of Dynamic Programming, Part II.* Dekker, New York, NY, 1982.

[10] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1991.

[11] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.

[12] S. Lavallée, J. Troccaz, L. Gaborit, A. L. Benabid P. Cinquin, and D. Hoffman. Image-guided operating robot: A clinical application in stereotactic neurosurgery. In R. H. Taylor, S. Lavallée, G. C. Burdea, and R. Mösges, editors, *Computer-Integrated Surgery*, pages 343–351. Mit Press, Cambridge, MA, 1996.

[13] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.

[14] N. P. Papanikolopoulos, 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. Robot. & Autom.*, 9(1):14–35, February 1993.