Department of Computer Science
George Mason University
Technical Report Series

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/
703-993-1530

Following a Large Unpredictable Group of Targets Among Obstacles

Christopher Vo
cvo1@cs.gmu.edu

Jyh-Ming Lien
lien@cs.gmu.edu

Technical Report GMU-CS-TR-2010-3

Abstract

Camera control is essential in both virtual and real-world environments. Our work focuses on an instance of camera control called target following, and offers an algorithm, based on the ideas of *monotonic tracking regions* and *ghost targets*, for following a large coherent group of targets with unknown trajectories, among known obstacles. In multiple-target following, the camera's primary objective is to follow and maximize visibility of multiple moving targets. For example, in video games, a third-person view camera may be controlled to follow a group of characters through complicated virtual environments. In robotics, a camera attached to robotic manipulators could also be controlled to observe live performers in a concert, monitor assembly of a mechanical system, or maintain task visibility during teleoperated surgical procedures. To the best of our knowledge, this work is the first attempting to address this particular instance of camera control.

1 Introduction

In *multiple-target following*, the camera's primary objective is to follow and maximize visibility of multiple moving targets. Multiple-target following is essential in both virtual and real-world environments. For example, in robotics, a camera attached to robotic manipulators could be controlled to observe a swarm of mobile robots, or live performers in a concert; monitor assembly of a mechanical system; or maintain task visibility during teleoperated surgical procedures. In video games, a third-person view camera may be controlled to follow a group of characters through complicated virtual environments. In general, it is difficult for a user to manually control the camera while also concentrating on other critical tasks. Therefore, it is desirable to have an *autonomous camera system* that handles the camera movement. This paper focuses on an instance of camera control called *multiple-target following*, and offers an algorithm for *autonomous* following a large coherent group of $10 \sim 100$ targets with unknown trajectories, among known obstacles.

The camera control problem has been studied extensively in both robotics and computer graphics because of its broad applications, such as dynamic data visualization [1, 2], robotic and unmanned vehicle teleoperation [3], and video games [4, 5]. Unfortunately, many of these methods are not applicable to follow a large group of targets in real-time *among obstacles*. For example, there is a large body of work in robotics, where researchers have studied similar problems such as pursuit and evasion, visual servoing [6] and cooperative multi-robot observation of multiple moving targets (CMOMMT) [7]. However, these strategies usually apply to environments with sparse or no obstacles. While there exist methods that do consider occlusion [8, 9], they still only consider situations where the target's trajectory is known *a priori*, or are only applicable to follow a few (2 or 3) targets [10, 11]. Many researchers [12, 13, 14] have also considered the case where both trajectory and environment are unknown. The main idea of these motion planners is to greedily minimize the *escaping risk* or maximize the *shortest escaping distance* of the target. In all of these methods, the camera trajectory can be computed efficiently, but is usually sub-optimal because only local information about the environment is considered, and the time horizon for planning is very short. Moreover, to the best of our knowledge, these motion strategies are all designed to track a single target.

In computer graphics, much work on camera planning (see recent surveys [15, 5]) is script-based [16], purely reactive [17, 6] (whose time horizon equals one), or mostly focuses on problems with predefined target trajectories [18, 19, 20, 21, 22, 23, 24]. Many of these methods are based on constraint solving, objective satisfaction or both. They are mostly designed for offline use and take a long time (usually seconds) to find a single camera placement.

The main goal of this paper is to provide an initial investigation into this important problem. Due to the nature of the aforementioned applications (video games, mobile robot swarms, virtual prototyping and group control), our investigation will focus on following a coherent group of targets (such as a crowd or a flock) by a single camera. Maintaining the visibility of a coherent group in some aspects is easier than tracking a single target because there is more than one target that the camera can follow. However, tracking a group is more difficult if the camera needs to maximize the number of visible targets over time. Difficulty also stems from the fact that a group can assume different shapes (e.g., forming a long line in a narrow corridor and a blob in an open area), clutter around the obstacles, or even split into multiple sub-groups (for a short period of time).

Our Work and Main Contributions. In this paper, we present a motion strategy that allows a single camera to robustly follow, at interactive rates, a large group (e.g., with 100 targets) among obstacles. In the aforementioned applications, some information about the environment is usually available. We believe that, given this information about the environment, the planner should be able to perform deeper *lookahead* in the search space and therefore provide better real-time camera following strategies even when the motion of the targets is unknown. The main idea is to preprocess the given environment offline to generate a data structure called *monotonic tracking regions* (MTRs) (defined later in Section 5) that can be used to assist real-time planning. This representation allows the camera to plan more efficiently by reducing the possible target movements to a smaller, discrete space. This significant increase in efficiency allows us to generate and evaluate multiple alternative plans in real-time. Our method also uses target coherency to better predict target positions.

In addition to the MTR-based motion strategy, we also present three new strategies (in Section 9) extended from the existing single-target techniques. We present reactive, sampling-based [25], and escaping-risk-based [26] methods. Our experimental results show that MTR-based method performs significantly better than the other strategies, especially in the environments with small obstacles and sinuous tunnels. To the best of our knowledge, this work is the first attempting to address this particular instance of camera control.

2 Related Work

There exist many methods for following a target with known trajectory, such as work done by LaValle et al. [8] using dynamic programming, and by Goemans and Overmars [27] using probabilistic motion planner. In this section, we will review strategies for tracking a target with unknown trajectory. We will also provide a brief review on camera planning methods in computer graphics. From our review, no method has focused on the problem of following a large group.

Planning with Unknown Target Trajectories in Known Environment. There exists some work considering tracking targets with unknown trajectories in a known environment [11, 28, 29]. The general idea is to partition the space into non-critical regions in which the camera can follow the target without complex *compliant motion*, that is, rotating the line of visibility between the camera and the target around a vertex of an obstacle. The main benefit of this line of work is the ability to determine the *decidability* of the camera tracking problem [30]. Unfortunately, the decomposition usually results in many small components even for a very simple environment. The brute-force approach for computing a visibility graph, visibility polygons of all vertices, and the visible regions of all finely discretized grid cells, is not scalable enough for real-world environments. While the nature of workspace decomposition in our MTR method is similar to the combination of [11, 28], MTR creates much fewer components and therefore can handle more realistic problems. Moreover, in [11, 28], Murrieta-Cid et al. require that the distance between the camera and the target is fixed and must precompute and store exhaustive visibility information. MTR does not have these limitations.

Recently, Li and Cheng [4] have proposed a real-time planner that tracks a target with unknown trajectory. Their main ideas include a *budgeted roadmap method* with lazy evaluation and a simple linear extrapolation to predict target's motion. However, their method has no guarantees on the performance and quality of the camera path. Oskam et al. [26] developed a *visibility transition planning* method which exhaustively precomputes visibility on a roadmap of overlapping spheres in the free space. This visibility roadmap enables quick prediction of spheres that represent high risk of occlusion so that proactive motion can be taken to prevent this occlusion.

However, this grid-based sphere generation step has difficulty in narrow passages, and exhaustive visibility computation is inefficient.

Planning with Unknown Target Trajectories in Unknown Environment. Many researchers have also considered the case that both trajectory and environment are unknown. For example, Becker et al. [25] proposed a very simple planner to follow a target with unknown trajectory in an environment with landmarks. The idea is to predict the target's next position and place the camera at the position that can see most of the predicted target positions. Later, González-Baños et al. [12, 13] proposed the main idea of greedily minimizing the *escaping risk* or maximizing the *shortest escaping distance* of the target. Recently, Bandyopadhyay et al. [31, 14, 32] improved the definition of *escaping risk* by introducing the idea of *vantage zone* and showed that this definition can improve the camera's tracking ability. All of these methods focus on tracking a single target. There are also extensive studies on multiple object tracking, e.g., [33], in which the goal is to maintain a belief of where the target(s) are using Kalman or particle filters. On the contrary, our goal is to maintain the visibility of targets.

Methods Based on Constraint Satisfaction. In computer graphics, camera planning is often viewed as a constraint satisfaction problem, and so there have been several attempts to represent the problem so that it can be solved efficiently with constraint satisfaction techniques. For example, several works use the idea of *screen space* or *image space* constraints, e.g., Blinn [18] and Gleicher and Witkin [24]. There are a number of works which involve the use of metaheuristics to compute optimal positions or trajectories for the camera. For example, Drucker and Zeltzer [34] used an A* planner to compute a camera path. Along the path, the orientation of the camera is then solved frame-by-frame to satisfy given constraints.

Constraints can also be represented geometrically. For example, Bares et al. [23] proposed a method to find camera positions (rather than a path) to meet given constraints. In this method, the constraints are represented as convex objects. The solutions are in the intersection of these convex objects. Jardillier and Languénou [19] proposed an offline planner (assuming known object geometry and trajectory), where constraints are specified using a declarative model. The constraints are then solved using *interval mathematics*. Later, Christie and Languénou [35] proposed another declarative model approach, this time to describe trajectories of cameras as sequences of parameterized elementary movements called *hypertubes*. However, this method is designed for offline purpose, taking 3 to 6 seconds to solve for a given trajectory.

Our MTR method is similar to the idea of *semantic space partitioning*. Christie and Normand [15] used this idea to partition the space so that in each partition, the cinematographic properties remain the same. However, their method is designed to produce a camera view that satisfies certain criteria—not to follow a target.

Reactive and Real-time Behaviors. Some studies have focused on developing reactive behaviors for real-time camera motion. For example, Courty and Marchand [6] used visual servoing along with obstacle and occlusion avoidance. Prediction of a target's movement is also relevant for developing real-time camera behaviors. For example, Halper et al. [17] introduced a camera planner that predicts state based on the past trajectory and acceleration. They also proposed the idea of PVR (potential visibility region) for visibility computation, and a pipelined constraint solver.

3 Preliminaries

In this section, we formally define the problem that we attempt to solve and notations used throughout the paper. We assume that the workspace is populated with known obstacles represented by polygons. These polygons are the projection of 3D objects that can potentially block the camera's view. This projection essential reduces our problem to a 2D workspace. We further assume that, initially, at least one of the targets is visible by the camera C, and, during the entire simulation, the targets T exhibit certain degree of coherence in their motion, and it is also possible that T can split into multiple subgroups for a short period of time (similar to a flock of birds). The targets are either controlled by the user or by another program, so the trajectories of the targets are not known in advance. However we assume that the size of T and T's maximum (linear) velocity v_T^{max} are known. The position $x_\tau(t)$ and the velocity $v_\tau(t)$ of a target $\tau \in T$ at time t are known only if τ is visible by the camera.

The camera C also has a bounded linear velocity v_C^{max}. The camera C's view is defined as a tuple: $(\alpha, r_{near}, r_{far})$, where α is the angle of view, and r_{near} and r_{far} define the near and far view range. The exact configuration of this view range at time t, denoted as $\mathcal{V}_C(t)$, is defined by the camera's view direction $\theta_C(t)$ and location $x_C(t)$. The position x_C of the camera is simply governed by the following equation: $x_C(t+\triangle t) = x_C(t) + \triangle t \cdot v_C(t)$, where $v_C(t)$ is the camera's velocity at time t.

Given the positions of the targets and the position of the camera, one can compute the camera's view direction so that the number of targets inside the view range is maximized.

Therefore, the problem of target following then is reduced to find a sequence of velocities $v_C(t)$:

$$\arg\max_{v_C(t)} \left(\sum_t \text{card}(\{T' \subset T \mid X_{T'}(t) \subset \mathscr{V}_C(t)\}) \right) , \quad (1)$$

subject to the constraints that, for all t, $v_C(t) \leq v_C^{max}$, and $x_C(t)$ is collision free.

4 Overview of Our Method

The main ideas of the proposed method are to (1) identify regions with simple (monotonic) topological feature so that the planner can repetitively use the same data structure and strategy to follow the target group, and (2) utilize the fact the targets form a coherent group.

Monotonic Tracking Region (MTR). The first step of the proposed method decomposes the environment into a set of *monotonic tracking regions* (MTRs). These regions usually look like tunnels and may overlap with each other. Intuitively, in these tunnel-like regions, the camera can *monotonically* maintain the visibility by moving forward or backward along a trajectory that *supports* the tunnel. More specifically, the main property MTR is that each MTR is topologically a *linear subdivision* so that the problem of camera following in an MTR can be represented as a *linear programming problem*. Note that such an MTR needs not to be convex or star shaped, and, in fact, it can have an arbitrary number of turns (like a sinuous tunnel). Moreover, MTR decomposition usually creates much fewer components than convex or star-shaped decompositions but, as we will see later, still provide similar functionality in target tracking. More precise definition of MTR and the process of computing these regions will be given in Section 5. In Sections 6 and 7, we will discuss how to track the target in a single MTR and then in multiple MTRs.

Ghost Targets. Our planner takes advantage of the fact that the camera is following a group of somewhat coherent targets. When the number of the targets visible by the camera is smaller than the total number of the targets, the planner will generate a set of *ghost targets* in the invisible regions. Our experiments show that the idea of ghost targets significantly increases the visibility. More precise definition of the ghost targets will be discussed in Section 8.

5 Build Monotonic Tracking Regions (MTRs)

Definition. We let a region M_π be a 2D generalized cylinder defined with respect to a *supporting path* π. We say π is a supporting path of M_π if every point $x \in M_\pi$ can see a subset of π. Because of this property, M_π can essentially be viewed as a linear object and the camera can see every point in M_π by moving along π. Specifically, we define M_π as:

$$M_\pi = \{x \mid \exists y \in \pi \text{ s.t. } \overline{xy} \subset C_{free}\} , \quad (2)$$

where $\overline{xy}$ is an open line segment between x and y, and C_{free} is the free space (i.e., the area without obstacles). Furthermore, we define the subset of π visible by x as: $V_\pi(x) = \{y \in \pi \mid \overline{xy} \subset C_{free}\}$. Note that $V_\pi(x)$ can have one or multiple connected components. Finally, we say a region $M_\pi \in C_{free}$ is an MTR supported by π if

$$|V_\pi(x)| = 1, \forall x \in M_\pi , \quad (3)$$

where $|X|$ is the number of connect components in a set X. Because each $x \in M_\pi$ can see only an interval of π, we can compactly represent the visible region (called visibility interval) of x as a tuple $V_\pi(x) = (s,t), 0 \leq s \leq t \leq 1$, if we parameterize π from 0 to 1. Fig. 1 shows an example of MTR and its supporting path π.

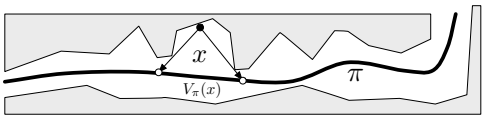


Figure 1: An example of monotonic tracking regions defined by π

With the definition in hand, our task here is to first find a set of supporting paths whose MTRs that will cover C_{free}, and, next, from a given path π, we compute the associated MTR and the visibility interval $V_\pi(x)$ for every point x in the MTR. We will describe these two steps in detail next.

Constructing supporting paths. Our strategy here is to find the homotopy groups G of the C_{free}. We propose to use the medial axis (MA) of the C_{free} to capture G because of its several interesting relationships with MTRs. First of all, we can show that the *retraction region* of every edge π on the MA forms an MTR (supported by π).

Lemma 5.1. *The retraction region $R \subset C_{free}$ of an edge on the* MA *forms an* MTR.

Proof. Let π be an edge on the medial axis MA of C_{free}. The retraction region $R \subset C_{free}$ of π is simply a set of

points that can be continuously retracted to π by a retraction function $r : R \rightarrow \pi$ [36]. By definition, given an arbitrary point $x \in R$, the largest circle c centered at the point $r(x)$ with x on c's boundary must be empty. Therefore, it follows naturally that each point in R must be able to see at least a point on π.

Now we briefly show that each point x can only see a consecutive region of π. We prove this by contradiction. Assuming that x can see multiple intervals of π. This means that there must be an obstacle between x and π. However, this contradicts the fact that x can be retracted to π in a straight line. Thus, we conclude that R forms an MTR. □

Therefore, the supporting paths are simply constructed by extracting the edges from the MA of a given environment.

Constructing MTRs. Given an edge π of MA, its retraction region R forms an MTR supported by π. However, simply using R as π's MTR can be overly conservative. The set of points that satisfy Eq. 2 and 3 is usually larger than R. To address this issue, we iteratively expand R by considering the points adjacent to R until no points can be included without violating the definition of MTR.

Next, we compute the visibility interval for every point in an MTR. The brute force approach that computes the visibility interval for each point from scratch is no doubt time consuming. To speed up the computation, our approach is based on the following observation.

Observation 5.2. *If x and x' are (topological) neighbors, and x is further away from π than x' is, then $V_\pi(x) \subset V_\pi(x')$.*

For example, in Fig. 1, imagining a point x' below x, x' can see a larger interval of π than x does. That is if we can compute the visibility intervals $V_\pi(x')$ for all the points x' on π, then we should be able to obtain the visibility intervals for x that are $\triangle d$ away from x' by searching inside $V_\pi(x')$.

Dominated MTRs. The exact MA of a given environment can contain small (and in many cases unnecessary) features and result in many small MTRs. In many cases, these MTRs are unnecessary and should be removed. This is when an MTR is dominated by another MTR. We say MTR M' is dominated by another MTR M if $M' \subset M$. In our implementation, we use an approximate MA [36] to avoid small features, and then identify and remove dominated MTRs.

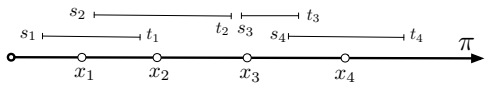


Figure 2: Make predictions for the next $h = 4$ future steps.

6 Follow the Targets in an MTR

The motivating idea behind decomposing the environment into a set of MTRs is that the target following problem in MTR can be solved much easily than that in the original environment. In fact, as we will see in this section, the camera can solve a long time horizon plan in MTR using linear programming.

Follow a single target. To simplify our discussion, we will first describe how a single target can be tracked in MTR. Let $x_\tau(t)$ be the current position of the target τ. Since we know the cureent speed of the target, we can estimate the positions $x_\tau(t+\triangle t)$ in the next time step. In order to keep the target in the view, the camera's next position $x_C(t+\triangle t)$ must be: $x_C(t+\triangle t) \in V_\pi(x_\tau(t+\triangle t))$. Note that this estimation can be applied to an arbitrary value of $\triangle t$. However, when $\triangle t$ is bigger, the position of the target becomes less accurate.

Let $I_i = V_\pi(x_\tau(t+i\cdot\triangle t)) = (s_i, t_i)$. Here i is an integer from 1 to h, where h is the user-defined time horizon. Recall that both s_i and t_i are parameters on the parameterized path π. In order to follow the target for h steps, the planner needs find a sequence of camera locations x_i from a sequence of parameterized intervals such that every point x_i is in its corresponding interval I_i without violating the constraint on the camera's max speed (see Fig. 2). This can be done by solving a h dimensional linear programming problem:

$$\begin{aligned} \min \quad & t_h - x_h \\ \text{s.t.} \quad & s_i \le x_i \le t_i \\ & 0 \le (x_{i+1} - x_i) \le \frac{v_C^{max}}{|\pi|}, \forall x_i \ , \end{aligned} \tag{4}$$

where $v_C^{max}/|\pi|$ is the maximum *normalized* distance that the camera can travel on π. Finally, the camera's future locations are simply $x_C(t+\triangle t\cdot i) = \pi(x_i)$.

Note that the rationale behind the minimization of $(t_h - x_h)$ is that when the target moves further away beyond h steps in the future, the camera will have better chance of keeping the target in the view when it is located closer to t_h along the path π. We call the above linear programming problem the *canonical following problem*. Solving a canonical following problem can be done efficiently since h is usually not large ($h = 20$ is used in our exper-

iments) given that modern linear programming solvers can handle thousands of variables efficiently.

It is possible that the linear programming problem has no feasible solution. We reduce the plan horizon iteratively until a solution is found.

Follow multiple targets. Now, we will extend this canonical following problem to handle multiple targets T. Let $x_T(t)$ be the current positions of the targets T. Similar to the case of a single target, we estimate the positions $x_T(t+\triangle t)$ in the next time step. In order to see a least one target, the camera must move so that

$$x_C(t+\triangle t) \in I(\triangle t) = \bigcup_{x \in x_T(t+\triangle t)} V_\pi(x) \ .$$

To simplify our notation, let $I_i = I_i(i \cdot \triangle t) = (s_i, t_i)$. By placing the camera in I_i, we can guarantee that at least one target is visible. However, our goal is to maximize the number of visible targets, at least over the planning horizon. To do so, we segment I_i into j sub-intervals I_i^j, each of which can see n_i^j targets. Then our goal is to pick a sub-interval from each I_i so that the total number of visible targets is maximized while still maintaining the constraint that the minimum distance between I_i^j and I_{i+1}^k is smaller than $v_T^{max} \triangle t$. Fortunately, this optimization problem can be solved greedily by iteratively adding the sub-interval with the largest n_i^j without violating the constraint. Once the sub-interval from each I_i is identified, the problem of finding the camera positions is formulated as a linear programming problem in the same way as Eq. 4.

When the targets and the camera moves (but still in the same MTR), we may need to repetitively solve the canonical following problem. Instead, we use *lazy update*. That is, we only update the old solution if it cannot satisfy all the constraints in Eq. 4.

7 Follow the Target between MTRs

In this section, we will discuss strategies to follow the targets across MTRs. Without loss of generality, we only consider the case with two MTRs. The same approach can be naturally extended to handle three or more intersecting MTRs.

Given two MTRs, M and M', we let $X = M \cap M'$. The intersection X of two MTRs plays two critical roles. On the one hand, when the target enters X, the possibility for the target to escape increases. On the other hand, when the camera enters X, the possibility of seeing the target increases.

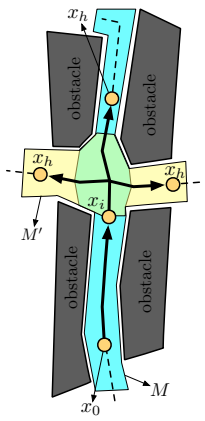


Figure 3: Search tree for multiple MTRs.

Recall that we plan the camera's location by predicting the targets' future locations $x_T(t+\triangle t)$ and by computing the visible intervals $I(\triangle t)$ of $x_T(t+\triangle t)$. When $x_T(t+\triangle t)$ reaches X, each $x_T(t+\triangle t)$ can have two intervals, one in M and the other one in M'. To compute the future locations of the camera, we solve at most *four* canonical following problems. Two of these are in M and the other two are in M'. Fig. 3 shows an example, in which the targets may move from M to M' after time step i. Therefore, the camera will solve one canonical following from step 0 to i and three canonical following problems from step i to h. Finally, the best solution that maximizes the visibility will be used to move the camera.

8 Put It All Together

In this section, we summarize the strategies for following a group and describe the implementation details. In particular, we distinguish between offline and online computations.

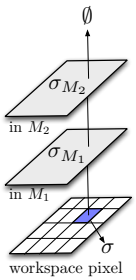


Figure 4: Data structure for MTRs.

Offline computation. Instead of a continuous space, our implementation is built on a regular grid which uni-

formly discretizes the workspace. The dimension of the (square) grid cell is the size of a target. We first approximate the workspace's medial axis MA using [36] and compute an MTR for each MA edge. Each MTR is a collection of cells that can see part of its supporting path π. Each cell in the grid stores (1) a boolean variable which indicates if it is in C_{free}, and (2) a list of linked MTR *cells*, denoted as σ_M. Each σ_M belongs to the MTR, M and stores the visibility interval $V_\pi(\sigma_M) = (s_i, t_i)$. A cartoon on the left illustrates this. Advantages of using grid-based representation also include efficiency and applicability to video games and mobile robots, which often store the environmental data in bitmaps.

Online computation. This is where the actual tracking take place. In every time step, the camera obtains a list of visible targets, and computes the ghost targets accordingly (see below). To update the existing plan, the positions of the visible and ghost targets are used to retrieve occupied MTRs and visibility intervals from the precomputed grid, which acts like an open hash table. Then, the canonical following problems, if any, associated with the new MTRs are solved. The camera then simply follows the trajectory that maximize the visibility.

Ghost targets. Our planner can estimate the positions of the targets even when they are not visible by the camera. We call these invisible targets "ghost targets." Note that the planner does not distinguish if a target is visible or is a ghost. Therefore the planning strategy described in the previous sections remains the same. The positions of the ghost targets are estimated based on the following assumptions: (1) targets tend to stay together as a group, and (2) invisible targets are in C-free outside the visibility region of the camera. Therefore, even if targets are invisible, they must be in some occluded regions nearby. Again, in our implementation, we use the grid to find the ghost targets by determining the occluded regions near the cells occupied by the visible targets.

9 Experimental Results

9.1 Three Additional Following Strategies

We also developed three additional group following strategies. These strategies are extensions of the existing methods which are originally designed to track a single target. Since there is no prior work on group following, we will also use these strategies to compare against the proposed MTR camera.

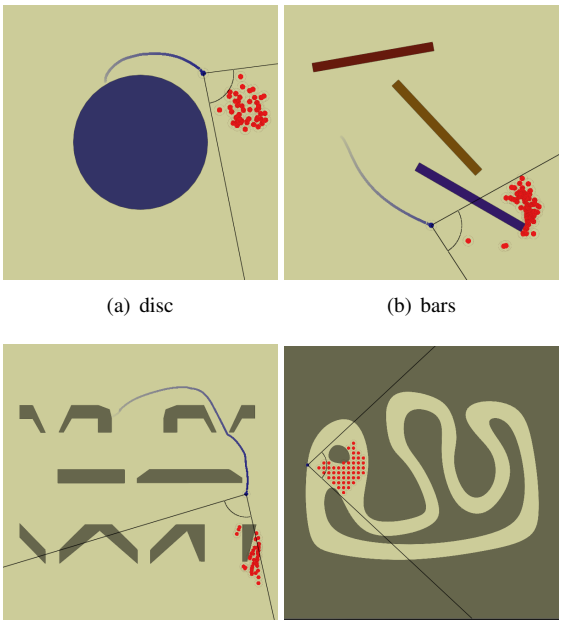
Reactive Camera. This reactive camera determines its next configuration by placing the visible targets as center in the view as possible based only on the targets' currently positions. The motivation is that by placing the visible targets at the center of its view, the camera will have better chance to make invisible targets visible.

IO Camera. IO camera is a sampling-based method extended from [25]. At each time step, given the visible targets T, the planner first creates k point sets P_T, where k is a parameter. Each point set contains $|T|$ predicted target positions. The predicted position of each target τ is sampled from the region visible from τ, and is at most $(v_T^{max} \cdot \triangle t)$ away from τ. The planner then creates a set P_C of camera configurations that are at most $(v_C^{max} \cdot \triangle t)$ away from C. To decide the next camera configuration, we simply determine $\arg\max_{x \in P_C} (\sum_{X \in P_T} \text{vis}(x, X))$, where $\text{vis}(x, X)$ is the number of points in X visible by x.

To simplify our discussion, we will use the notation IO-k to denote an IO camera that samples k point sets for target prediction (e.g., in Fig. 6). It's important to remember that the IO cameras usually cannot be used as an online planner for tracking a large group (e.g., more than 50 targets) because of the large number of visibility checks between the sampled target positions and camera configurations in every time step.

VAR Camera. VAR camera is based on [26]. Here, we first obtain a coarse representation of the environment by sampling a grid of discs in C-free. A roadmap is formed over the intersections of the discs. Finally, visibility is computed between each pair of discs with Monte-Carlo raytracing. Our VAR method is a hybrid approach that uses the constructed visibility information in a reactive behavior when the camera has good visibility of the targets (more than 50% of the targets are currently visible by the camera), and uses visibility-aware path planning from [26] to plan short, alternative paths to reach predicted locations of targets when the camera is far away.

The reactive behavior in VAR computes a waypoint for the camera on each frame. First, we find a disc D_c that is closest to the camera, and a disc D_r (from the precomputed roadmap) that represents an imminent occlusion risk. That is, the disc D_r is one that is in the direction of the visible targets' velocity, closest to the centroid of the visible targets, and whose visibility from the camera's disc is less than 100%. A waypoint is selected along the ray extending from D_r passing through D_c. The visibility-aware path planner is the same as described in [26]. It uses an A* algorithm on the pre-computed roadmap with a heuristic function to compute a path to the D_r which simultaneously minimizes distance traveled and maximizes visibility of the target.



(a) disc (b) bars

(c) islands (d) RSS

Figure 5: Experimental environments.

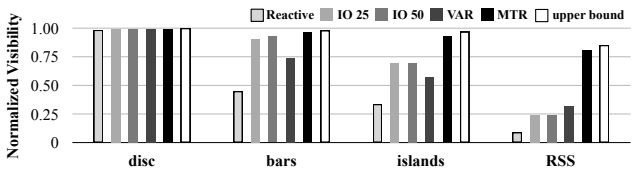


Figure 6: Following 50 targets in four environments.

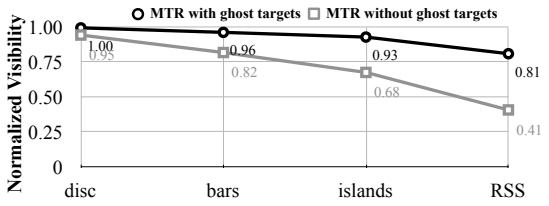


Figure 7: Comparing MTR cameras with and without ghost target.

9.2 Experiments and Results

In our experiments, the target group is constantly moving toward a random goal, which is not known by the camera. If all targets are invisible, the camera will stay stationary. Throughout the experiments, we measure the performance of the cameras by computing the *normalized visibility* which is the ratio of the visible targets during the entire simulation. Every data point presented in the plots in this section is an average over 32 runs, each of which is a 10000 time-step simulation. We set the planning horizon $h = 20$ for all MTR cameras.

We perform our experiments in four workspaces shown in Fig. 5. These workspaces are designed to test the performance of the cameras in various conditions, such as large open space (disc), open space with narrow gaps (bars), small irregular obstacles with many narrow gaps (islands) and long sinuous narrow passages (RSS). Both islands and RSS environments are considered difficult as the targets tend to separate around the small obstacles or hide behind a bend in the passage.

MTR outperforms other strategies. Our first experiment in Fig. 6 shows strong evidence that MTR camera consistently performs better than the other cameras when following 50 targets in all environments. Note that we also include data called *upper bound* obtained from an MTR camera that has no speed limitation, i.e., it can move to the best configuration instantly. The strong performance of MTR is further supported by the small difference between the MTR camera and the upper bound in all four environments.

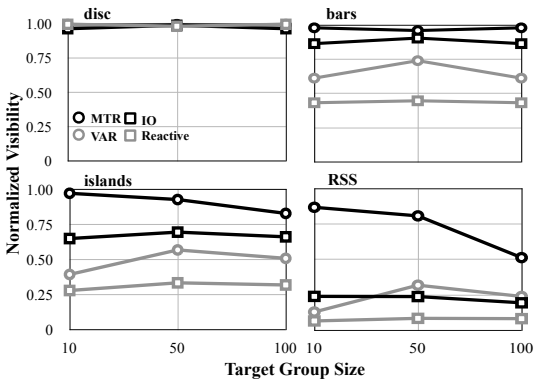
It is clear that the reactive camera performs worst, except in the disc environment. The VAR camera is the second worst, except in the RSS environment. However, as we will see, the VAR camera seems to handle large and fast moving targets better because of its ability in estimating risks. Although IO cameras perform well in some situations, IO-25 in the bar environment is more than 200 times slower than VAR (≈2600 fps) and 12 times slower than MTR (≈157 fps), thus cannot be used in many applications, such as real-time task monitoring and video game. There is no significant difference between IO-25 and IO-50.

Ghost targets boost performance. In this experiment shown in Fig. 7, we attempt to estimate quantitively the performance gain due to the idea of *ghost targets* (GS). Our result shows that the performance gain is more significant in more difficult environments. If compare to the (very time consuming) IO cameras in Fig. 6, MTR without GS is only slightly better in the island and RSS environments, but there are significant differences between the IO cameras and MTR with GS. In the future, it will be interesting to measure the performance gain by applying GS to reactive and VAR cameras.

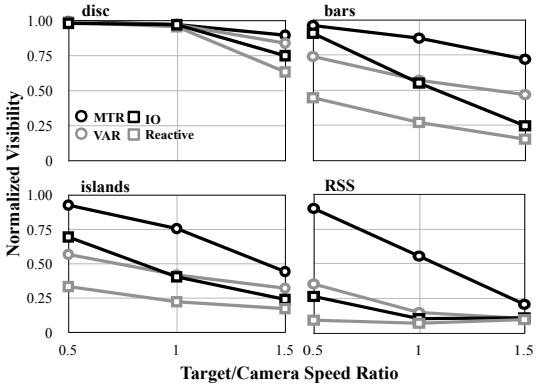
Target size and target speed. In our last experiment, we attempt to gain better understanding of our cameras by varying the size and the maximum speed of the targets. In Fig. 8(a), we vary the sizes of the targets from 10 to 100, and we can see that the size change has little

effect on their performance, except in RSS environment. By further examining the simulations, we observed that this performance drop is in fact inevitable (unless multiple cameras are used) because it is simply impossible for a single camera to see the entire group, e.g., when the group contour bends with the environments.

In Fig. 8(b), we vary the speed of the target from $\frac{1}{2}$ (which is what used in the previous experiments) to twice the camera speed. When the targets and camera have the same maximum velocity, MTR keeps more than 70% visibility of the targets in all environments and is still significantly better than other cameras. However, when the targets are 1.5 times faster than the camera, MTR, although still performance noticeably better than other cameras, its performance drops quite significantly, especially in the island and RSS environments. Again, we found that this performance drop is due to the environmental constraints (e.g., when the targets escaped, the RSS environment provides no shortcut for the camera to re-capture the targets), and is inevitable unless multiple cameras are used.



(a)



(b)

Figure 8: Camera performance with varying target sizes and maximum speeds.

10 Conclusion

In this paper, we contribute an online camera planning method, called MTR, which is suitable for autonomous following of multiple targets with unknown trajectories among known obstacles. The idea is to preprocess the known environment offline to obtain *monotonic tracking regions* (MTRs) which can then be used to increase the efficiency of the online planner. In addition to MTR, we present three new methods for camera planning that are extensions of existing single-target work, and we compare the performance of MTR with each of these methods. In the tested scenarios, MTR performs significantly better than all other methods, particularly in situations where environments are cluttered with many obstacles or long sinuous tunnels. We also show that adding *ghost targets* to MTR increases the performance.

Additional experiments involving varied target sizes and maximum velocities show that in some situations multiple cameras may be needed to achieve better performance. Therefore, we are working on applying this method to tracking scenarios involving multiple cameras, and scenarios involving multiple objectives (such as quality, or cinematographic preferences).

References

[1] D. D. Seligmann and S. Feiner, "Automated generation of intent-based 3d illustrations," *SIGGRAPH Comput. Graph.*, vol. 25, no. 4, pp. 123–132, 1991. 1

[2] W. H. Bares, L. S. Zettlemoyer, D. W. Rodriguez, and J. C. Lester, "Task-sensitive cinematography interfaces for interactive 3d learning environments," in *IUI '98: Proceedings of the 3rd international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 1998, pp. 81–88. 1

[3] S. Hughes and M. Lewis, "Robotic camera control for remote exploration," in *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2004, pp. 511–517. 1

[4] T.-Y. Li and C.-C. Cheng, "Real-time camera planning for navigation in virtual environments," in *SG*

'08: Proceedings of the 9th international symposium on Smart Graphics. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 118–129. 1, 2

[5] M. Christie, P. Olivier, and J.-M. Normand, "Camera control in computer graphics," *Computer Graphics Forum*, vol. 27, no. 8, pp. 2197 – 2218, 2008. 1

[6] N. Courty and E. Marchand, "Computer animation: a new application for image-based visual servoing," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation, 2001.*, vol. 1, 2001, pp. 223– 228. 1, 2

[7] L. E. Parker, "Distributed algorithms for multi-robot observation of multiple moving targets," *Auton. Robots*, vol. 12, no. 3, pp. 231–255, 2002. 1

[8] S. LaValle, H. Gonzalez-Banos, C. Becker, and J.-C. Latombe, "Motion strategies for maintaining visibility of a moving target," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 1, Apr 1997, pp. 731–736 vol.1. 1, 2

[9] T.-Y. Li and T.-H. Yu, "Planning tracking motions for an intelligent virtual camera," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, 1999, pp. 1353–1358 vol.2. 1

[10] B. Jung and G. Sukhatme, "A region-based approach for cooperative multi-target tracking in a structured environment," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 3, 2002, pp. 2764–2769 vol.3. 1

[11] R. Murrieta-Cid, B. Tovar, and S. Hutchinson, "A sampling-based motion planning approach to maintain visibility of unpredictable targets," *Auton. Robots*, vol. 19, no. 3, pp. 285–300, 2005. 1, 2

[12] C.-Y. Lee, H. González-Baños, and J.-C. Latombe, "Real-time tracking of an unpredictable target amidst unknown obstacles," 2002. 1, 2

[13] H. Gonzalez-Banos, C.-Y. Lee, and J.-C. Latombe, "Real-time combinatorial tracking of a target moving unpredictably among obstacles," in *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, vol. 2, 2002, pp. 1683–1690 vol.2. 1, 2

[14] T. Bandyopadhyay, Y. Li, M. Ang Jr., and D. Hsu, "A greedy strategy for tracking a locally predictable target among obstacles," in *Proc. IEEE Int. Conf. on Robotics & Automation*, 2006, pp. 2342–2347. 1, 2

[15] M. Christie and J.-M. Normand, "A semantic space partitioning approach to virtual camera composition," *Computer Graphics Forum*, vol. 24, no. 3, pp. 247 – 256, 2005. 1, 2

[16] A. Butz, "Anymation with cathi," in *In Proceedings of the 14th Annual National Conference on Artificial Intelligence (AAAI/IAAI)*, 1997, pp. 957–962. 1

[17] N. Halper, R. Helbing, and T. Strothotte, "A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence," *Computer Graphics Forum*, vol. 20, no. 3, pp. 174 – 183, 2002. 1, 2

[18] J. Blinn, "Where am I? What am I looking at?" *IEEE Computer Graphics and Applications*, vol. 8, no. 4, pp. 76 – 81, July 1988. 1, 2

[19] F. Jardillier and E. Languénou, "Screen-space constraints for camera movements: the virtual cameraman," *Computer Graphics Forum*, vol. 17, no. 3, pp. 175 – 186, 2001. 1, 2

[20] C. Ware and S. Osborne, "Exploration and virtual camera control in virtual three dimensional environments," *SIGGRAPH Comput. Graph.*, vol. 24, no. 2, pp. 175–183, 1990. 1

[21] D. B. Christianson, S. E. Anderson, L.-W. He, D. H. Salesin, D. S. Weld, and M. F. Cohen, "Declarative camera control for automatic cinematography," in *Proc. of the National Conference on AI*, 1996, pp. 148–155. 1

[22] P. Coleman, K. Singh, L. Barrett, N. Sudarsanam, and C. Grimm, "3d screen-space widgets for non-linear projection," in *GRAPHITE '05: Proc. of the 3rd international conference on Computer graphics and interactive techniques*, 2005, pp. 221–228. 1

[23] W. H. Bares, J. P. Grégoire, and J. C. Lester, "Realtime constraint-based cinematography for complex interactive 3d worlds," in *AAAI '98/IAAI '98*. AAAI, 1998, pp. 1101–1106. 1, 2

[24] M. Gleicher and A. Witkin, "Through-the-lens camera control," in *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1992, pp. 331–340. 1, 2

[25] C. Becker, H. González-Ba nos, J.-C. Latombe, and C. Tomasi, "An intelligent observer," in *The 4th International Symposium on Experimental Robotics IV*. London, UK: Springer-Verlag, 1997, pp. 153–160. 1, 2, 9.1

[26] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross, "Visibility transition planning for dynamic camera control," in *SCA '09: Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. New York, NY, USA: ACM, 2009, pp. 55–65. 1, 2, 9.1

[27] O. Goemans and M. Overmars, "Automatic generation of camera motion to track a moving guide," in *In International Workshop on the Algorithmic Foundations of Robotics*, 2004, pp. 187–202. 2

[28] R. Murrieta-Cid, T. Muppirala, A. Sarmiento, S. Bhattacharya, and S. Hutchinson, "Surveillance Strategies for a Pursuer with Finite Sensor Range," *The International Journal of Robotics Research*, vol. 26, no. 3, pp. 233–253, 2007. 2

[29] S. Bhattacharya, S. Candido, and S. Hutchinson, "Motion strategies for surveillance," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007. 2

[30] S. Bhattacharya and S. Hutchinson, "On the existence of nash equilibrium for a two-player pursuit-evasion game with visibility constraints," *Int. J. of Rob. Res*, vol. 57, pp. 251–265, 2009. 2

[31] T. Bandyopadhyay, Y. Li, M. H. A. A. Jr, and D. Hsu, "Stealth tracking of an unpredictable target among obstacles," *Algorithmic Foundations of Robotics VI*, vol. 17, pp. 43–58, October 2005. 2

[32] T. Bandyopadhyay, M. Ang Jr., and D. Hsu, "Motion planning for 3-d target tracking among obstacles," in *Proc. Int. Symp. on Robotics Research (ISRR)*, 2007. 2

[33] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People Tracking with Mobile Robots Using Sample-Based Joint Probabilistic Data Association Filters," *The International Journal of Robotics Research*, vol. 22, no. 2, pp. 99–116, 2003. 2

[34] S. M. Drucker and D. Zeltzer, "Intelligent camera control in a virtual environment," in *In Proceedings of Graphics Interface '94*, 1994, pp. 190–199. 2

[35] M. Christie and E. Languénou, "A constraint-based approach to camera path planning," in *Smart Graphics*, 2003, pp. 172–181. 2

[36] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, vol. 2, 1999, pp. 1024–1031. 5, 5, 8