Group Following in Monotonic Tracking Regions *

Christopher Vo Jyh-Ming Lien †

Abstract

For a 2-d pursuit-evasion game called group following, we study a data structure called *monotonic tracking regions* (MTR). An MTR has a support path so that the points along the path can collectively see every point in the MTR. An MTR can be considered as a generalization of a star-shaped region. Using an MTR, we can plan online the path of a bounded-speed camera tracking a group of n agents to a planning horizon h in time $O(hn^2)$.

1 Introduction

Monotonic tracking regions (MTRs) [Vo and Lien 2010] are a data structure to guide a single camera following multiple coherent targets in a 2D workspace. Intuitively, in an MTR, the camera can maintain visibility of targets by moving along a trajectory. This MTR data structure allows us to generate the camera's motion online by solving a *linear programming problem*.

We assume that the camera C has a bounded linear velocity v_C^{max}. The exact configuration of this view 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 of the camera is simply: $x_C(t+\triangle t) = x_C(t) + \triangle t \cdot v_C(t)$, where $v_C(t)$ is the camera's velocity at time t. The target T comprises a group of coherent *non-adversarial* members, whose trajectories are not known in advance. We also assume that the size of T and T's maximum (linear) velocity v_T^{max} are known (by the camera). The position of $x_\tau(t)$ a target $\tau \in T$ at time t is known only if τ is visible by the camera. We attempt to maximize the number of visible targets:

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

where $\text{card}(\mathcal{X})$ is the number of elements in $\mathcal{X}$.

2 Monotonic Tracking Regions (MTRs)

We let a 2D region $\mathcal{M}_\pi$ be a generalized cylinder defined w.r.t a *supporting path* π. We say π is a supporting path of $\mathcal{M}_\pi$ if every point $x \in \mathcal{M}_\pi$ can see a subset of π. Consequently, the visibility of π spans $\mathcal{M}_\pi$.

Definition 2.1. *$\mathcal{M}_\pi \subset \mathcal{F}$ is a region supported by a path π if $\mathcal{M}_\pi = \{x \mid \exists y \in \pi \text{ s.t. } \overline{xy} \subset \mathcal{F}\}$, where $\overline{xy}$ is an open line segment between x and y, and $\mathcal{F}$ is the free space (i.e., the area without obstacles).*

Furthermore, we define the subset of π visible by x as: $\mathcal{V}_\pi(x) = \{y \in \pi \mid \overline{xy} \subset \mathcal{F}\}$. Finally, we define MTR.

*This work is supported in part by NSF IIS-096053, NSF EFRI-1240459, AFOSR FA9550-12-1-0238.

†Both authors are with Department of Computer Science, George Mason University, Fairfax, VA 22030 USA.

Definition 2.2. *A region $\mathcal{M}_\pi \in \mathcal{F}$ is an MTR supported by π if $|\mathcal{V}_\pi(x)| = 1, \forall x \in \mathcal{M}_\pi$, where $|\mathcal{X}|$ is the number of connect components in a set $\mathcal{X}$.*

Because each $x \in \mathcal{M}_\pi$ can see only an interval of π, we can compactly represent the visible region (called visibility interval) of x as a tuple $\mathcal{V}_\pi(x) = (s,t), 0 \le s \le t \le 1$, if we parameterize π from 0 to 1.

2.1 Follow a single target

Let $x_\tau(t)$ be the position of the target τ at time t. Since we know the maximum speed of the target, we can estimate the positions $x_\tau(t+\triangle t)$ in the next time step, i.e., the intersection of $\mathcal{F}$ and a disc with radius $\triangle t \cdot v_T^{max}$. In order to keep the target in the view, the camera's next position $x_C(t+\triangle t)$ must be:

$$x_C(t+\triangle t) \in \mathcal{V}_\pi\left(x_\tau(t+\triangle t)\right) = \bigcap_{x \in x_\tau(t+\triangle t)} \mathcal{V}_\pi(x) \ .$$

Let $I_i = \mathcal{V}_\pi(x_\tau(t+i\cdot\triangle t)) = (s_i, t_i)$. Here i is an integer from 1 to h, where h is the user-defined time horizon. 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 parameterized camera locations x_i from a sequence of intervals such that every point x_i is in its corresponding interval I_i. In addition, one may desire to minimize the distance travelled by the camera. Taking all these into consideration, this problem can be formulated as an h-dimensional linear programming (LP) problem:

$$\begin{array}{ll} \min & x_h - x_1 \\ \text{s.t.} & s_i \le x_i \le t_i, \forall i \in \{1, 2, \cdots, h\} \end{array}$$

We call the above LP problem the *canonical following problem*. Solving a canonical following problem can be done efficiently since h is usually small (≤ 20).

2.2 Follow multiple targets

Now, we will extend the 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

$$x_C(t+\triangle t) \in \bigcup_{\tau \in T} \mathcal{V}_\pi(x_\tau(t+\triangle t)) = \bigcup_{\tau \in T} \left(\bigcap_{x \in x_\tau(t+\triangle t)} \mathcal{V}_\pi(x) \right) .$$

To simplify our notation, let $I_i = \bigcup_{\tau \in T} \mathcal{V}_\pi(x_\tau(t) + i \cdot \triangle t) = (s_i, t_i)$. 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 subintervals I_i^j, each of which can see n_i^j

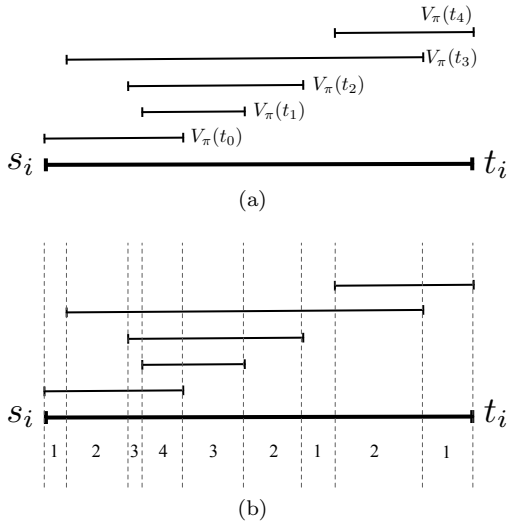

Figure 1: (a) The interval $I_i = (s_i, t_i) = \cup_{\tau \in T} \mathcal{V}_\pi(x_\tau)$. (b) The interval I_i is segmented into 9 subintervals, each of which is a set of points in π that can see the same number of targets, which is shown below each interval.

targets. Fig. 1(a) shows an example of I_i defined as the union the all the visibility intervals $\mathcal{V}_\pi(x_\tau)$ of the targets τ. Note that I_i may contain multiple connected components. Fig. 1(b) shows the subdivision of I_i (i.e., subintervals I_i^j) bounded by the end points of $\mathcal{V}_\pi(x_\tau)$. Each I_i^j is associated with the number of visible targets n_i^j. When the velocity of the camera is unlimited, then the optimal strategy is to pick the subinterval I_i^j with the largest n_i^j in each I_i, i.e., I_i is shrunk to I_i^j. Thus, instead of solving the the following problem using I_i, the subintervals I_i^j will be used. See Fig. 2.

From Fig. 2, one can also see that the distance that the camera has to travel from x_2 to x_3 is quite long, thus the camera will need to move very fast to maintain the maximum visibility. When the camera speed is bounded, this may not always be possible. Therefore, we need a way to select a subinterval from each I_i so that the total number of visible targets is maximized while still maintaining the constraint that the minimum distance between $I_i^j \subset I_i$ and $I_{i+1}^k \subset I_{i+1}$ is smaller than $\triangle t \cdot v_T^{max}$. More specifically, we would like to find a solution to the following problem:

$$\arg\max_{\{j_i\}} \left(\sum_{i=1}^{h} n_i^{j_i} \right) \text{ s.t. } \text{dist}(I_i^{j_i}, I_{i+1}^{j_{i+1}}) \leq \triangle t \cdot v_T^{max}, \forall i \ ,$$

where j_i is the index of the j_i-th subinterval in interval I_i, and $\text{dist}(x, y)$ is the closest distance between two subintervals x and y. Although, at the first glance, this problem seems to be another LP problem, fortunately, Lemma 2.3 shows that the optimal subintervals can be found in $O(hn^2)$ time, where n is the number of targets and h is the time horizon.

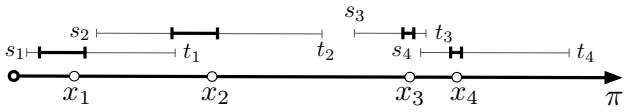

Figure 2: Make predictions in subintervals with maximum targets visibility for the next $h = 4$ future steps.

Lemma 2.3. *Finding all $I_i^{j_i}$ will take $O(hn^2)$ time for n targets and h planning time horizon.*

Proof. The main idea is to construct a directed graph from these subintervals and the current position of the camera, and show that this graph must be a DAG with $O(hn)$ vertices and $O(hn^2)$ edges. Then the problem of find a sequence of optimal subintervals become the longest path search problem in the DAG, which can be solved in time linear to the size of the graph.

To construct such a graph, we first define the idea of *reachability*. Given two subintervals u and v from two consecutive intervals, the reachable interval $r(u, v) \in v$ is a set of points in v that the camera can reach from u in one step without violating the speed constraint. If $r(u, v)$ is not empty, then we say v is reachable from u. Note that the reachability can be nested, i.e., given three subintervals u, v, and w, we say that w is reachable from u if $r(u, w) = r(r(u, v), w)$ is not empty. In the graph that we will construct, we ensure that every node in the graph is reachable from the current position x_C of the camera. Finally, we say that a subinterval v is reachable by the camera if $r(x_C, v)$ is not empty.

Specifically, we let the current position x_C of the camera be the source of the graph and let the subintervals be the rest of the nodes in the graph. The source are then connected to the subintervals in I_1 that are reachable by the camera. The each reachable subinterval in I_1 are connected to the subintervals in I_2 that are reachable by the camera. The process repeats until the reachable intervals in I_h are connected by those in I_{h-1}. Note that since we only need to pick one subinterval from each interval, the subintervals within each interval are not connected. Finally, we let the edge weight be the number of visible targets in the destination node. The graph constructed this way must be a DAG since there is no back edge. Any path the connects the source to a sink will contain a sequence of valid subintervals. Thus, finding the maximize number of targets visible from these subintervals is equivalent to finding the longest path in the DAG, which can be solved in linear time using topological sort. Since each interval will have $\Theta(2n)$ subintervals and two consecutive intervals will have $4n^2$ edges, this DAG has $O(hn)$ vertices and $O(hn^2)$ edges. □

References

Vo, C., and Lien, J.-M. 2010. Following a large unpredictable group of targets among obstacles. In *The Third International Conference on Motion in Games*.