Topologically distinct trajectory predictions for probabilistic pursuit

Florian Shkurti and Gregory Dudek

***Abstract*— We address the integrated planning and control problem that enables a single follower robot (the "photographer") to maintain a moving target (the "subject") in its field of view for as long as possible. We propose a real-time pursuit algorithm that seamlessly handles the often neglected, yet unavoidable, scenario in which the target escapes the follower's field of view; a scenario that simple, reactive controllers are ill-equipped to handle. Our algorithm aims to minimize the expected time until visual contact is re-established, which enables the photographer to track the subject for as long as possible, even in the presence of loss of visibility. At the core of our pursuit algorithm is an efficient method for sampling plausible trajectories from different homotopy classes. We do this by generating topologically distinct shortest paths by using the Voronoi diagram. We use these paths to make informed, model-based predictions of the likely future locations of the target, given a history of observations. Given these predictions, our algorithm produces pursuit trajectories that approximately minimize the expected time to recover visual contact. We show that constraining the predictive pursuit problem to the space of homotopy classes condenses the expanse of possibilities that our algorithm must consider, which enables target tracking in large occupancy grids, as opposed to many POMDP methods that are constrained to small environments. We benchmark the tracking behavior of our algorithm against the baseline of human subjects who performed the same set of pursuit tasks in simulation, as well as against two other pursuit algorithms that only take into account paths from a single homotopy class. We show that considering homotopy alternatives in 2D pursuit improves the tracking performance and that our algorithm does at least as well as humans in most pursuit scenarios.**

I. INTRODUCTION

This paper addresses a class of strategies that can be used by a robot photographer to maintain visual contact with an oblivious subject. We explicitly address the case where visual contact is lost, and propose an algorithm that aims to minimize the expected time to recover it. This is novel with respect to previous work, particularly pursuit-evasion games, which do not consider the (inevitable in practice) scenario of loss of visibility. We assume that the robot photographer makes plans using observations on the target, but no other means of communication. We also assume that the target is purposeful, to the extent that it is heading for some goal and is oblivious to the actions of the photographer, as opposed to being deliberatively evasive.

What should a photographer robot do when its subject escapes the field of view (i.e. is no longer in visual contact)? A number of common phenomena could lead to this, including tracker failure, occlusions due to the structure of the environment and moving objects, or even the complexity of the scene. In many of these scenarios, simple, reactive controllers are insufficient for re-establishing visual contact with the subject, because they require continuous feedback of the target's state. In this work we propose an effective pursuit algorithm that is able to handle loss of visual contact. Our method operates on both positive and negative observations received from the target tracker and maintains a belief distribution on the target's possible locations. While the target remains unseen, the belief expands spatially, uncertainty grows. Our planning algorithm generates actions that shrink uncertainty or even completely *clear* it, which means that the target has been found. The key enabling insight that allows us to plan over the vast array of potential target paths is that we first group paths into homotopically consistent sets, thereby reducing the continuous optimization problem into a small discrete set of options. The potential applications of visibility-maximizing planners are diverse. One such application is in the rising field of robot cinematography where automated videographers [1], [2], often aerial vehicles, aim to ensure as long a coverage of a particular visual event as possible. Maintaining the target in the field of view is a prerequisite for optimizing for aesthetically pleasing shots. Another application includes robot assistants that follow human professionals, such as divers or scientists, and record their experiment sessions or provide support. Cooperative localization based on mutually-observing cameras is another application of maximizing for visibility and recovering from its loss [3].

The main challenge for re-establishing visual contact with a target is the expanding horizon of possible locations that

The authors are with the School of Computer Science and the Center for Intelligent Machines at McGill University in Montreal, Canada {florian, dudek}@cim.mcgill.ca

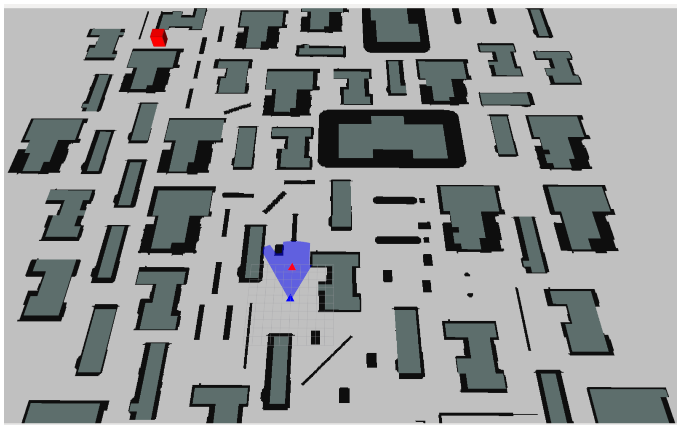


Fig. 1. A view of the initial configuration of the simulator. The blue robot is the follower, with an associated limited field of view. The red robot in the field of view is the target. The red cube denotes the destination of the target. Videos and more info about the project can be found at `http://www.cim.mcgill.ca/~mrl/topological_pursuit`

grow as time passes; the uncertainty about the target's location can grow at a rate at which the follower is physically unable to clear the belief, due to its speed being insufficient to keep up with the rate of expansion. In existing literature, most clearing algorithms assume *a team* of robots that coordinate in order to guarantee finding the target. It is an NP-hard problem to determine the minimum size of this team. Similarly, we have shown in prior work [4] that computing the minimum speed required for the follower to guarantee clearing its belief is NP-hard, but it is straightforward to show that under reasonable assumptions the problem gets harder as the target has more time to escape. As a result, the follower often has to consider a very large number of hypotheses.

Existing work on this subject can be found in the literature on pursuit-evasion games and reinforcement learning. Pursuit-evasion games formulate the tracking problem as a minimax optimization: the evader wants to minimize the escape time, while the follower wants to maximize it, and they provide solutions in terms of Nash equilibria. These games, however, end at the time of escape and provide no recommendations for what the follower should do afterwards. On the other hand, several existing reinforcement learning agents that operate on Partially-Observable Markov Decision Processes (POMDP), such as [5], do not constrain the predicted motion of the target hypotheses sufficiently, and allow for trajectories that would not be plausible for goal-directed navigation.

Our approach addresses both of these issues. Our main contribution is an efficient algorithm for making predictions of target motions that makes use of generated shortest paths that are topologically distinct in the sense of belonging to different homotopy classes[1]. Our method relies on having access to the Voronoi Diagram of the (known) world and on iteratively refining the K-shortest paths computed on the Generalized Voronoi Graph (GVG) representation of that Diagram. We use these refined, topologically distinct paths: (i) to guide the evolution of the follower's belief about the target's location, and (ii) to plan follower trajectories that optimize for the likelihood of re-establishing visual contact.

Using topological information to predict long-term target trajectories allows the follower to efficiently control the pruning of the possible paths considered within each homotopy class. In fact we show that even in cluttered environments our planner computes pursuit trajectories in real-time. We also show that, despite the pruning, the performance of our pursuit algorithm is comparable to human baseline.

II. RELATED WORK

Our problem is related to the large body of existing work on the theoretical bounds related to maintaining continuous surveillance of a moving target. This problem has received extensive consideration in the literature of pursuit-evasion games. For most work in this category, issues related to the optimization of visibility based on the structure of the environment are critical, so there are natural connections with the classic "art gallery" problem and its variants [6].

In the work of Lavalle *et al.* [7] the problem of planning the trajectory of a single observer trying to maximize visibility of a fully-predictable or partially-predictable target is considered. The problem setting is augmented in [8] which examines planning the paths of several observer robots whose goal is to provide continuous visual coverage of several unpredictable targets.

Generally, continuous visibility maintenance has been considered mostly for the case of adversarial targets. Such is the case for example in [9], [10], [11], [12]. The question of identifying whether a target is being cooperative or evasive and modifying the pursuer's plans in response to this degree of information is something that currently lacks a precise characterization in the literature. It is worth mentioning nevertheless, that if the target is adversarial there exist settings under which the evader can escape the follower's field of view, no matter what the follower does ([9], [10]).

The tracking problem has also been addressed by approximate sampling-based POMDP solvers such as SARSOP [13], which extends a Monte Carlo Search Tree and maintains both the upper and lower bounds of the state-action value function, which enables pruning of provably suboptimal actions. The proposed method, however, is demonstrated to work on small grids and runs in the order of minutes.

The papers that are most similar to ours are [14], [15], which relied on the same idea of using the GVG to generate K-shortest paths along the graph, in order to explore topological structure. However, the main application there is socially compliant navigation and local, shared autonomy for wheelchairs, whereas in this work we rely on the GVG for predicting the utility of long-term plans for pursuit and tracking in the absence of visual contact. Our work has also been influenced by existing ideas on topological and path diversity considerations in planning, such as [16], [17], [18].

III. MODEL-BASED PROBABILISTIC PURSUIT

In the present work we pose the probabilistic pursuit maintenance problem as a combination of classical planning and reinforcement learning where the instantaneous reward is the indicator random variable of whether the follower sees the target or not.

In this problem we are constrained by the fact that, in general, instead of having complete information of the location of the target, we only have access to a belief $bel(x_t) = p(x_t|z_{1:t}, u_{1:t})$ about its state x_t, given a history h_t that consists of: partial observations $z_{1:t}$ that the follower has experienced, and the sequence of controls and actions $u_{1:t}$ that it has executed[2]. Therefore, this formulation allows us to describe the problem as a Partially-Observable Markov Decision Process (POMDP).

The problem of finding the target after it has escaped the follower's field of view can be formulated as finding

[1]If a path is continuously deformable to another path, and they share the same endpoints, they are said to belong to the same homotopy class

[2]Note here that we are assuming the follower knows its own state as well as the map

the policy $\pi(u_{t+1}|h_t)$ that maximizes the value function $V^\pi(h_t) = E_\pi[R_t|h_t]$, where $R_t = \sum_{i=t}^{t+T} \gamma^{i-t} r_t$ is the (discounted) cumulative reward function, and r_t is the indicator function of whether the follower sees the target. Trying to solve this problem exactly is in general intractable, so sampling methods, such as [5], [19] are necessary.

For sampling to turn out to be helpful, we need to have a simulator for how our system behaves. In other words, we need a way to generate representative samples $(x_{t+1}, z_{t+1}, r_{t+1}) \sim \text{Sim}(x_t, u_t)$. The more representative this model is to the target's actual behavior the more certain the follower is about what the value of each available action is. For instance, modeling the target as a random walk is a bad idea because of lack of any destination and impractically long (in many cases infinite[3]) hitting times for a given location. On the other extreme, assuming perfect information about which route the target is going to follow and which speed it is going to use, is also not a realistic assumption in many cases.

We take the middle ground and we assume extra information about the long-term behavior of the target, namely that we know its final destination, but we do not know the route it is going to take to get there, nor the speed at which it will be travelling. We assume that the target is trying to efficiently reach its destination, even if it does not choose the optimal route. So, it tries to make monotonic progress towards reaching it. This precludes looping behavior, or strange intermediate stops, and it is an assumption inspired by the concept of *navigation functions* in potential fields and control theory. It is also worth mentioning that in this behavior model the target is completely oblivious to the actions of the follower, unlike in pursuit-evasion games in which the target has to come up with best responses to the possible actions of the follower.

Since we are to model the target as an efficient navigator, we need to predict or sample routes along which it will be travelling. Given the target's destination, and lacking any other information, for instance about visual landmarks along the route, or preferred routing information about the target, the only actionable information that the follower has in order to rank some routes more highly than others is the topology and visibility structure of the environment. Intuitively, we want to generate plausible paths that do not deviate from the length of the shortest path by a lot, but manage to explore a rich set of alternatives about how the target might want to go to its destination. In this work we search among these alternatives, and make predictions about the target's short- to long-term behavior, by using the concept of topologically distinct shortest paths.

A. Topologically distinct short paths via the GVG

We say that two paths are topologically distinct if they belong to different homotopy classes. Two paths that share the same endpoints are *homotopic* if there is a continuous function that deforms one into the other, without hitting any obstacles in the environment. It is important to clarify at this point that all the upcoming sections make the following two assumptions: (a) the pursuit is being done purely in 2D worlds[4], and (b) the homotopy classes considered contain simple paths, i.e. paths without self-intersections, having a winding number of zero for any obstacle. The concept of enumerating simple homotopy classes as a way of reasoning about the set of paths that they represent is promising because for many 2D environments these classes represent sufficiently many paths to cover the free space. They also allow planning at a level of abstraction that is more robust to changing paths, or small changes in the environment, than other representations such as occupancy grids.

We start by using the Voronoi diagram of the environment to find points that are equidistant from at least two obstacles, and on top of that structure we build the Generalized Voronoi Graph. Its edges include points that are equidistant to exactly two objects, while its nodes are at the remaining points of the Voronoi diagram, either as meetpoints of at least three edges, or as endpoints of an edge into a dead-end in the environment. These two structures provide a roadmap for navigation in the environment on which graph-based path planning and navigation takes place.

One of the main advantages of using the structure of the GVG in order to facilitate topological queries is that its size is usually small enough for realistic environments, which makes it suitable for real-time reasoning. The other advantage is that shortest path queries on the GVG yield simple (no loops) paths which suit the notion of progress to the destination that we mentioned above.

The path obtained from a shortest-path algorithm on the GVG is likely not plausible for prediction in terms of length, curvature and appearance. Shortest paths on the GVG are not globally optimal paths in the rest of the environment. We partially address this issue by using an iterative refinement procedure, where we replace parts of the path that are joinable by a straight line with the points on that line, until little improvement is possible[5]. Thus, we get plausible short paths from the GVG.

We use Yen's K-shortest paths algorithm [22] on the GVG to compute multiple paths to a given destination and then we iteratively refine each of the paths to reduce their length. An example outcome of this process is depicted in Fig. 2, which shows three topologically distinct short paths. While this heuristic does not guarantee global optimality with respect to length, it produces plausible alternatives for predictions in real time. For example, computing up to 50 distinct short paths requires less than a second on a modern machine.

1) Computational complexity: The computational complexity of Yen's K-shortest path algorithm is $O(K|V|S)$ where $|V|$ is the number of nodes of the GVG graph, and $O(S)$ is the computational complexity of the shortest path algorithm used as a subroutine for Yen's. In our case we

[3]For example, the expected time for Brownian motion starting at 0 to hit a horizontal line is infinity.

[4]In 3D, one can reason about persistent topological features [20], [21]

[5]This is essentially the Douglas-Peucker algorithm.

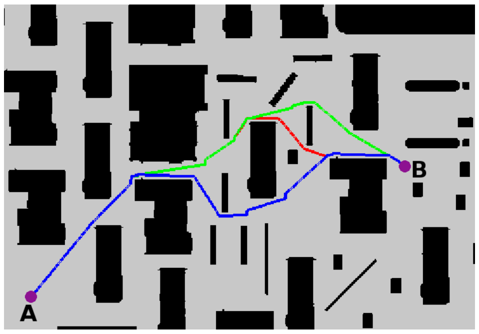


Fig. 2. Three homotopically distinct paths found by applying Yen's 3-shortest paths on the GVG and then refining the paths to reduce their length.

use Dijkstra, which has complexity $O(|E| + |V|\log|V|)$. GVG is a planar graph, which implies according to Euler's formula that $|V| - |E| + F = 2$, where F is the number of faces corresponding to the graph (regions enclosed by edges – in our case obstacles – and the outer region). Since for any undirected graph $\sum_{v \in V} \deg(v) = 2|E|$ we have $2F \geq 2|E| - 2|V| = \sum_{v \in V} \deg(v) - 2|V|$. We separate GVG nodes into the set V_1 of endpoints (degree 1) and the set $V - V_1$ of meetpoints (degree ≥ 3). We conclude that $2F \geq |V_1| + 3|V - V_1| - 2|V| = |V| + 2|V - V_1| - 2|V| = |V| - 2|V_1|$, or equivalently $2F + |V_1| \geq |V - V_1|$. This means that the number of meetpoints is at most linear in the number of obstacles and the number of endpoints.

That said, we are counting on the fact that the reduced form of the GVG [23] is a roadmap of a smooth version of the environment, and thus an efficient representation of the world, with a bounded number of endpoints per obstacle. In particular, it is more efficient than occupancy grid or pointclouds. For example, the cluttered world shown in Fig. 2 contains about 90 obstacles and is represented by about 200 GVG nodes.

In practice, the follower could get away with generating fewer alternatives because physically searching all the route hypotheses could require a lot of time, or very high speed. In the next few sections we describe how to use these paths as hypotheses for prediction.

2) Representation of the belief: When visual contact is lost, the probability of the target's location spreads outwards in the environment. We approximate this diffusion as a set of particles. The alternative option of using discretized 2D histograms of probabilities does not work well on obstacle boundaries. Empirically, the probabilities often get concentrated around these boundaries if propagated naively. In order to address this issue one could treat this as a fluid flow problem, in which fluid particles are going to bounce back from the boundaries. The main problem with these representations, however, is that it is not straightforward how to bias the flow in order to make it diffuse towards a certain destination, without solving differential equations to compute the entire vector field from source to sink.

3) Particle filter and Bayesian updates of the belief: Due to these reasons, we preferred the much simpler and efficient representation of the belief b about the follower's location as a set of particles $x^{(i)}$, whose weights are updated by a particle filter. When a particle is created we associate it with a fixed speed and a single reference topologically distinct path along which it travels. We model the motion of the particle as always traveling at that speed, and we rely on including enough particles in the belief to represent a wide range of speed combinations, without modeling variable speed on any single particle.

The transition model: The design choice of fixed speed makes the problem of predicting where the particle is going to be in a few time steps either purely deterministic or almost completely predictable, depending on design choice. In our case the only stochasticity for a particle's transition model arises from the initial choice of reference path $\gamma^{(i)}$ and speed $v^{(i)}$ for the particle. So, the particle dynamics model, $p(x_{t+1}^{(i)}|x_t^{(i)}, u_t) = \sum_{\gamma,v} p(x_{t+1}^{(i)}|x_t^{(i)}, u_t, \gamma, v)p(\gamma, v|x_t^{(i)}, u_t)$, which after the independence assumptions simplifies to $p(x_{t+1}^{(i)}|x_t^{(i)}, u_t, \gamma^{(i)}, v^{(i)})p(\gamma^{(i)})p(v^{(i)})$, depends almost entirely on the path and the speed. We construct the conditional transition model $p(x_{t+1}^{(i)}|x_t^{(i)}, u_t, \gamma^{(i)}, v^{(i)})$ to have very little noise or to be completely deterministic [24]. We describe below how we choose $p(\gamma)$ and $p(v)$.

In order to induce some ranking between the available topologically distinct paths γ we use their total length as their differentiating factor. $l(\gamma)$ denotes the length of that path. We use the softmin operator to turn these lengths into probabilities:

$$p(\gamma) = \exp(-l(\gamma)/\tau) / \sum_{\gamma_i} \exp(-l(\gamma_i)/\tau) \tag{1}$$

where τ is the softmin temperature parameter. In the limit, when τ reaches 0 softmin becomes the hard minimum, in other words it assigns full probability to the minimum length path. Higher values of τ encourage more exploration among paths of nearby lengths.

We model the speed distribution for particles using two criteria: first, that we guarantee that the maximum possible target velocity is assigned to many particles, so that their diffusion can catch up with the target's motion, even if it is being evasive; second, that no particles are too slow because they might induce the follower to stay behind and try to clear up hypotheses that have very low probability. With those requirements in mind we set:

$$p(v) = \begin{cases} \text{Uniform}[v_{\max}/2,\ v_{\max}] & b = 1 \\ \delta_{Dirac}(v - v_{\max}), & \text{otherwise} \end{cases} \tag{2}$$

where $b \in \text{Bernoulli}(0.5)$ and $v_{\max}$ is the maximum speed of the target. Approximately half the particles have full speed under this model while the remaining half will be at least half as fast as the target. Again, this encourages progress towards the known destination.

The observation model of the particle filter incorporates a maximum range and field of view limitation to model depth sensors such as the Kinect, but also to reflect the fact that today's target trackers are not generally reliable at large distances. The observation model may also encode particularities of the detector, such as its susceptibility to false positives and negatives as well as dependence to viewpoint [25]. The observation model we use assumes no false positives or negatives:

$$p(z_t|x_t^{(i)}) = \begin{cases} 1 & \text{iff} \quad x_t^{(i)} \in \text{FOV(follower at time t)} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

One of the advantages of this model is that particles disappear as soon as the follower finds them in its field of view, so they do not need to be included in future plans. The disadvantage is that if all the particles happen to be within the field of view, and the follower does not detect the target, they all become invalid at the same time, and we need to repopulate the belief. We handle this case by advancing each particle's motion to as far along its reference path as the time elapsed since the last positive target observation. Again, the assumption that each particle moves at its full speed along its path is critical here, too.

The choice of the particle filter as the main Bayesian filtering mechanism was made because it is able to incorporate *negative information*, in other words, not being able to currently see the target makes other locations outside the field of view more likely. This is something that other filters cannot provide.

B. Pursuit algorithm

Our pursuit algorithm incorporates two modules, depending on whether the target is currently in view or not. If it is, the follower uses reactive feedback PID control to reach a so called "paparazzi" reference frame behind the target (or whichever the desired viewing pose happens to be). This frame of reference gets updated as the target moves in compliance with the surrounding environment, so as to not hit any walls.

When visual contact is lost, planning replaces reactive following. We generate topologically distinct paths from the point where we last saw the target to its likely destination; we assign a reference path to each particle; and we start the sampling and resampling steps in the particle filter according to incoming observations. The follower plans a path that will lead to a high-value location, as outlined in Alg. 1, and executes that path. When the end of path is reached, if unsuccessful, the follower plans another one. If the target re-enters the field of view, the path hypotheses are discarded and reactive control resumes its operation.

In the previous section we insisted on being able to deterministically query where a particle is going to be at a particular time in the future. The main reason behind this was that we wanted to be able to compute the possible moments at which the follower could visually intercept it. To this end we assume the functionality of a function called MIN-TIME-TO-VIEW, which plans a trajectory in space and time that will enable the follower to navigate so as to bring itself in view of a location by a particular time. MIN-TIME-TO-VIEW yields a trajectory whose endpoint is a paparazzi frame that views a given location, together with a time estimate of when it is going to arrive at that frame. We implemented MIN-TIME-TO-VIEW for the case of omnidirectional robots using iterative refinement of the shortest path obtained from the GVG, as described in the previous section[6].

We restrict the set of candidate locations for visual interception to be points along the reference paths of the particles, for the sake of computational efficiency. Our pursuit algorithm computes the minimum times to reach paparazzi frames for a set of waypoints along these reference paths. The possible waypoints include the final destination of the target. So, heading directly to the destination without intermediate stops is one of the considered strategies. The algorithm then selects the paparazzi frame that sees the particle with the highest ratio of probability in the particle filter over interception time, and heads over to reach that paparazzi frame. This is a greedy nearest neighbor algorithm, and we term it NNm for "nearest neighbor pursuit with target prediction along multiple homotopy classes".

Algorithm 1 NNm($z_t,\ bel(x_t) = \{(x_t^{(i)}, w_i)\}_{i=1...M},\quad y_t$)

```
1:  $z_t$ : the follower's observation
2:  $bel(x_t)$ : the follower's belief about the target's pose
3:  $y_t$ : the follower's pose
4:  $g$ : the target's destination
5:  if $z_t = x_t$ i.e. the target is visible then
6:      $bel(x_{t+1}) \leftarrow$ PF-UPDATE$(\text{GVG}, bel(x_t), z_t, y_t, g)$
7:      $\bar{x}_t \leftarrow$ desired paparazzi frame based on $x_t$
            (e.g. behind the target)
8:      $\delta\theta, \delta r \leftarrow y_t - \bar{x}_t$
9:      $u_t \leftarrow$ feedback control from $\delta\theta, \delta r$
10:     $y_{t+1} \leftarrow f(y_t, u_t)$
11:     NNm$(z_{t+1},\ bel(x_{t+1}),\ y_{t+1})$
12: else
13:     $\lambda_{1:K} \leftarrow$ K-SHORTEST-PATHS$(\text{GVG}, x_t, g, \text{K})$
14:     $\gamma_{1:K} \leftarrow$ REFINE$(\lambda_{1:K})$
15:     $T_{jk}, \pi_{jk} \leftarrow$ MIN-TIME-TO-VIEW$(\text{GVG}, y_t, p_{jk}), \forall j, k$
            for waypoints $p_{jk}$ sampled along paths $\gamma_k$
16:     for $i = 1...M$ do
17:         Sample path $\gamma_{k_i} \sim p(\gamma)$ as in Eq. 1
18:         Sample speed $v_i \sim p(v)$ as in Eq. 2
19:         Assign path $\gamma_{k_i}$ and speed $v_i$ to particle $x_t^{(i)}$
20:     $I \leftarrow (i, j)$ such that particle $x_t^{(i)}$ is predicted to reach
            $p_{jk_i} \in \gamma_{k_i}$ close to follower's arrival time $T_{jk_i}$
21:     $i^*, j^* \leftarrow \underset{(i,j)\in I}{\text{argmax}} \frac{w_i}{T_{jk_i}}$
22:     while not reached $p_{j^*k_{i^*}}$ do
23:         $bel(x_{t+1}) \leftarrow$ PF-UPDATE$(\text{GVG}, bel(x_t),$
                $z_t, y_t, g)$
24:         $u_t \leftarrow$ next control in follower's trajectory $\pi_{j^*k_{i^*}}$
25:         $y_{t+1} \leftarrow f(y_t, u_t)$
26:         $t \leftarrow t + 1$
27:     Go to step 5
```

[6]Trajectory optimization and following is required for other types of dynamical systems, but that is outside the scope of this paper.

IV. RESULTS

We validate the performance of our pursuit algorithm, as well as the benefit added from involving topological information, in two substantial ways: first, by comparing our algorithm's performance to pursuit policies of expert humans; second, by comparing our algorithm to two other tracking algorithms, called NNs and NNr, that do not reason about multiple homotopy classes when they predict the motion of the target. Instead, they use the same pursuit algorithm as NNm. NNs stands for "nearest-neighbor pursuit with target prediction along the shortest path to the destination." It predicts feasible and likely interception points along the shortest path to the known destination. NNr stands for "nearest-neighbor pursuit with target prediction along a short path in a randomly-chosen homotopy class." The random choice among available homotopy classes in NNr is done according to the softmin rule in Eq. 1, which favors short paths.

A. Experimental Setup

In order to enable both of these types of comparisons we created a simulator which includes two representations of omnidirectional robots, one of which is equipped with a camera sensor, with a limited field of view (about 50 degrees) and a limited viewing range (8 meters). These limits were set based on sensors like the Kinect.

Our simulator indicates that the target is in view with probability 1 if and only if it is indeed, so it models only true positives and true negatives. At each point in time the robot has access to its own pose, as well as the relative pose of the target, provided it is visible. If not, then the follower does not get to see where the target is on the map – it needs to make informed guesses. The only helpful information it has is the eventual destination of the target. We selected worlds that are quite complex, with many sharp turns across small distances, as is shown in Fig. 1, in order to make it easier for visibility to be lost, and likely for early mistakes to be penalized highly in terms of performance in the pursuit.

We set up 40 pursuit tasks in total, all with the same starting configuration for the two robots, but with different destinations for the target, one per task. In all initial configurations the follower could see the target. The follower was controllable by joystick for the human users, or by position and velocity control by our planners. We decided to cluster these 40 tasks into eight separate groups, each of which consisted of 5 pursuit tasks at different destinations. Groups 1-4 were used for the comparison between NNm and human pursuit policies, while groups 5-8 were used for comparing algorithmic policies, namely NNm against NNr and NNs. The maximum speeds of the follower vs the target are shown here:

Group	1	2	3	4	5	6	7	8
Follower Max Speed	2	2.1	3	4	3	3.4	3.8	4.2
Target Max Speed	2	2	3	4	3	3	3	3

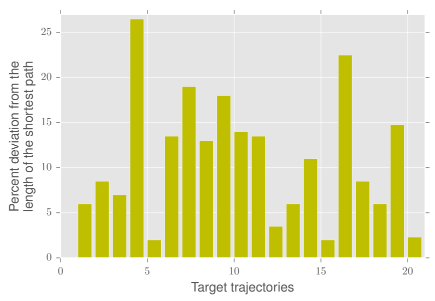


Fig. 3. The percentage of deviation in length of the target's trajectories compared to the optimal path from the initial configuration to the final destination.

We found that allowing the target to be faster than the follower produced low success rates both for humans and our algorithms, particularly in the range of high speeds mentioned above, so we decided against further examining it as a viable scenario.

The 20 pursuit tasks in the comparison of humans vs NNm were done in a single map, while the remaining 20 comparisons between algorithms were performed in 5 different maps that presented various degrees of difficulty and challenges for pursuit.

Each of the 40 tasks lasted from 20 seconds up to about a minute. The trajectories of the target were prerecorded offline from human users, and they were played back for each task. These trajectories were chosen so that they took advantage of the full speed of the target. While they do not account for evasive behaviors such as hiding at a fixed place, moving backwards, or doing loops and self-intersections, they can be characterized as efficient, goal-directed trajectories that do not always go through the optimal route. This can easily be seen in Fig. 3, where we plot how much the 20 target trajectories used for Groups 1-4 deviated from the optimal paths. The fact that the target trajectories are not following optimal routes can even better be demonstrated through the concept of *homotopy signatures* [17]. A homotopy signature is a vector of size equal to the number of obstacles in the map. Each entry contains the winding angle of the trajectory with respect to a point on an obstacle in the environment. In our case, since we are not modelling targets that do loops, this signature simplifies to a binary vector. In this case, an entry in the homotopy signature vector is true if the trajectory passes "to the left" of the associated obstacle and false otherwise.

We used these concepts to compare how the signatures of each of the 20 target trajectories compare to the signatures of their respective optimal paths. The results are shown in Fig. 4, and they indicate that an intelligent follower will indeed need to make decisions that involve more homotopy classes than that of the shortest path.

B. Human baseline for probabilistic pursuit

We recruited 10 people, most of them expert roboticists in their late twenties, all of them with prior experience in using joystick control, and one third of them reporting experience in First-Person Shooter games. We allowed each user to

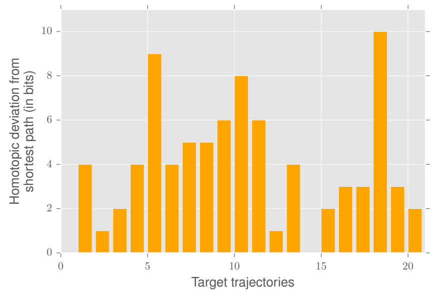


Fig. 4. Bit difference of homotopy signature vectors between each target trajectory that was recorded for Groups 1-4 vs. the shortest paths in each scenario. The median is 4, meaning that in half of the pursuit tasks, if we were to deform the target's trajectory to the shortest path, we would hit 4 out of the approximately 90 obstacles present in the map.

spend as much time as necessary to familiarize themselves with the controls of the simulator and the high speed behavior through practice sessions. Once each task began, the simulator showed them their current score, namely in how many camera frames they had visual contact with the target. It also showed them the time left for each task through a countdown timer. All users reported paying minimal attention to both of these numbers as they were focused on the game of pursuit. The results of this user study are shown in Fig. 5.

Humans exhibit the largest standard deviation in their pursuit performance compared to NNm. Our multi-homotopy exploring algorithm performs at least as well as the human baseline in 14/20 pursuit tasks, especially the ones of Groups 3,4, which require fast reaction. Our algorithm does better, in some cases even by 10%. This advantage is dependent on and indicative of the environment's complexity, because any mistake made early on in the pursuit can have a large impact on the user's final score for the task. It is worth mentioning that human performance on the closely related Traveling Salesman Problem is close to optimal for small size problems (of 10 to 20 cities) [26]. The planning component of some of these algorithms takes about a second to compute (when visual contact is lost), depending on the number of hypotheses being considered. In that time the target is allowed to move and might have disappeared. To account for this need for planning during the pursuit, we perform the comparisons between algorithms offline. NNm runs in real time if we allow lower resolution in the discretization of time during the planning step, and limit the number of hypotheses generated. The input for both the human participants and the three algorithms was identical: the same target trajectories were replayed to all agents in their respective comparisons.

C. *Benchmarking algorithmic performance*

In order to precisely quantify what the added benefit of exploring multiple homotopy classes was, we compared our algorithm against two other algorithms, called NNs and NNr, which use the same pursuit logic but differ in the way they predict the future motion of the target. NNs makes predictions in the homotopy class along the shortest path to the known destination, while NNr makes predictions along a randomly sampled homotopy class, according to the softmin distribution in Eq. 1, which favors short paths. We set the temperature for NNm and NNr to $\tau = 2$ for all the experiments shown above.

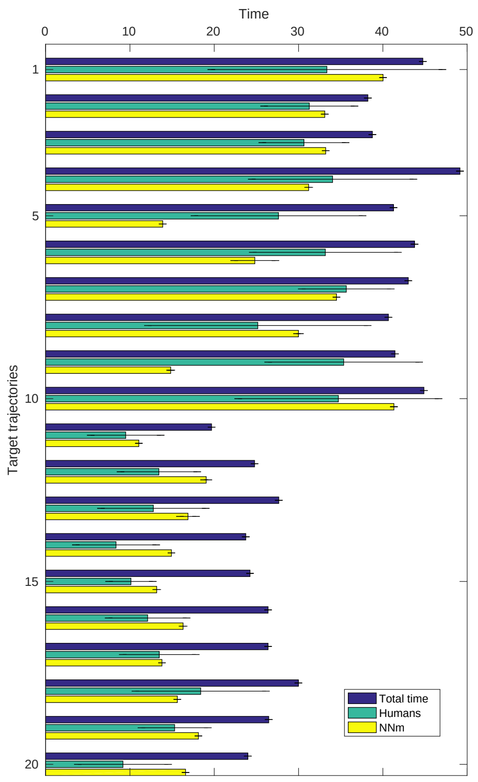
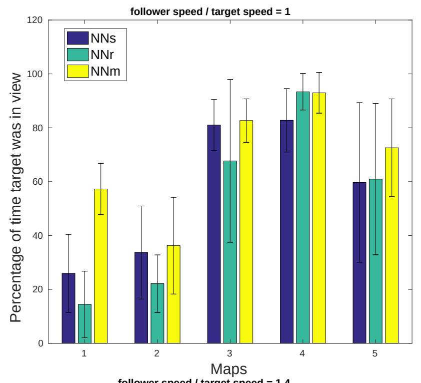


Fig. 5. Comparison between average human performance and the NNm pursuit algorithm (Nearest Neighbor with multi-homotopy prediction). The top bar in each triple is the average total time required for the completion of the trajectory. The middle is the average time humans managed to remain in visual contact with the target robot during that time. The bottom is the average time that NNm maintained visual contact. NNm performed at least as well as humans in 14/20 trajectories. (Seen better in color)

We compared these algorithms in 5 different maps under varying relative speed settings for the follower and the target, as outlined in Groups 5-8. We found that in challenging maps with small mean free path score, and obstacle layout that required many turns, our algorithm NNm outperformed NNs and NNr. In less challenging maps where the obstacle density is lower we found that the difference among the three algorithms was insignificant. The high density of obstacles in the challenging maps makes reasoning about multiple homotopies necessary because they can be seen as plausible perturbations of the shortest path, which the target might consider as valid routes to its destination. This effect was present across all relative speed variations in Group 5-8. NNm did at least as well as NNs and NNr in all 20 tasks and outperformed them in half.

V. CONCLUSIONS

In this paper we address probabilistic pursuit focusing on the issue of visibility maintenance in the presence of occasional interruptions of the visual contact between the target and the observer. We presented a core idea for predicting a moving target's motion using an enumeration of topologically distinct paths and how this enumeration can be applied to the problem of probabilistic pursuit to make



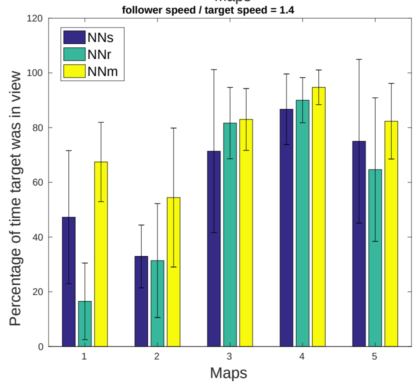


Fig. 6. Average percentage of time that the target was in view during pursuit in different maps. Higher is better. NNm is the third column, and it does at least as well or outperforms the other two methods.

long-term predictions about the target's behavior.

We showed that the nearest-neighbor pursuit algorithm that results from incorporating this idea, using efficient planning algorithms on the Generalized Voronoi Graph outperforms two similar pursuit algorithms that only consider paths in one homotopy class. This demonstrates the utility of reasoning about topology in pursuit problems, particularly in environments with high density of obstacles. Finally, we demonstrated through a user study that for a complex environment the resulting pursuer is competitive compared to human performance.

In future work we hope to better characterize the criteria for how many independent hypotheses (particles) should be maintained for each homotopy class. We are also working on a formal characterization of types of targets that are parameterized according to their degree of cooperation in the pursuit, interpolating between fully adversarial to fully cooperative. We believe that a broad range of practical problems can be addressed using these notions of probabilistic pursuit, especially problems where by natural processes need to be carefully documented by a robot observer.

ACKNOWLEDGMENT

This work was funded by the NSERC Canadian Field Robotics Network (NCFRN).

REFERENCES

[1] C. Schroeter, M. Hoechemer, S. Mueller, and H.-M. Gross, "Autonomous Robot Cameraman - Observation Pose Optimization for a Mobile Service Robot in Indoor Living Space," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 424–429.

[2] R. Bodor, A. Drenner, M. Janssen, P. Schrater, and N. Papanikolopoulos, "Mobile Camera Positioning to Optimize the Observability of Human Activity Recognition Tasks ," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2005, pp. 1564–1569.

[3] P. Giguere, I. Rekleitis, and M. Latulippe, "I see you, you see me: Cooperative localization through bearing-only mutually observing robots," in *2012 IEEE IROS*, Oct 2012, pp. 863–869.

[4] F. Shkurti and G. Dudek, "On the complexity of searching for an evader with a faster pursuer," in *IEEE ICRA*, 2013, pp. 4047–4052.

[5] D. Silver and J. Veness, "Monte-Carlo Planning in Large POMDPs," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 2164–2172.

[6] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[7] S. M. Lavalle, H. Gonzalez-Banos, C. Becker, and J.-C. Latombe, "Motion Strategies for Maintaining Visibility of a Moving Target," in *IEEE ICRA*, 1997, pp. 731–736.

[8] R. Murrieta-Cid, B. Tovar, and S. Hutchinson, "A Sampling-Based Motion Planning Approach to Maintain Visibility of Unpredictable Targets," *Autonomous Robots*, vol. 19, no. 3, pp. 285–300, Dec 2005.

[9] S. H. Sourabh Bhattacharya, "Approximation Schemes for Two-Player Pursuit Evasion Games with Visibility Constraints," in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[10] S. Bhattacharya and S. Hutchinson, "On the Existence of Nash Equilibrium for a Two-player Pursuit–Evasion Game with Visibility Constraints," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 831–839, Dec 2009.

[11] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion in a polygonal environment," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 875–884, Oct 2005.

[12] N. M. Stiffler and J. M. O'Kane, "Shortest paths for visibility-based pursuit-evasion," *IEEE ICRA*, pp. 3997–4002, May 2012.

[13] D. Hsu, W. S. Lee, and N. Rong, "A point-based POMDP planner for target tracking," in *IEEE International Conference on Robotics and Automation*, May 2008, pp. 2644–2650.

[14] M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard, "Online Generation of Homotopically Distinct Navigation Paths," in *International Conference on Robotics and Automation, ICRA*, 2014.

[15] L. Palmieri, A. Rudenko, and K. O. Arras, "A Fast Randomized Method to Find Homotopy Classes for Socially-Aware Navigation," in *IROS Workshop on Assistance and Service Robotics in a Human Environment*, 2015.

[16] M. S. Branicky, R. A. Knepper, and J. J. Kuffner, "Path and trajectory diversity: Theory and algorithms," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2008, pp. 1359–1364.

[17] S. Bhattacharya, V. Kumar, and M. Likhachev, "Search-Based Path Planning with Homotopy Class Constraints," in *AAAI*, 2010, pp. 1230–1237.

[18] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectory patterns with gaussian processes," in *IEEE 12th ICCV Workshops*, Sept 2009, pp. 1229–1234.

[19] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, Jul 2013.

[20] S. Bhattacharya, M. Likhachev, and V. Kumar, "Identification and Representation of Homotopy Classes of Trajectories for Search-based Path Planning in 3D," in *Robotics: Science and Systems (RSS)*, 2011.

[21] F. T. Pokorny, M. Hawasly, and S. Ramamoorthy, "Topological trajectory classification with filtrations of simplicial complexes and persistent homology," *The International Journal of Robotics Research*, vol. 35, no. 1-3, pp. 204–223, Jan 2016.

[22] J. Y. Yen, "Finding the K-Shortest Loopless Paths in a Network," *Management Science*, vol. 17, no. 11, pp. 712–716, jul 1971.

[23] H. Choset and J. Burdick, "Sensor-Based Exploration: The Hierarchical Generalized Voronoi Graph," *The International Journal of Robotics Research*, vol. 19, no. 2, pp. 96–125, 2000.

[24] A. Y. Ng and M. Jordan, "PEGASUS: A Policy Search Method for Large MDPs and POMDPs," in *Proceedings of 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 406–415.

[25] D. Meger, A. Gupta, and J. J. Little, "Viewpoint Detection Models for Sequential Embodied Object Category Recognition," in *IEEE ICRA*, 2010, pp. 5055–5061.

[26] J. N. Macgregor and T. Ormerod, "Human performance on the traveling salesman problem," *Perception & Psychophysics*, vol. 58, no. 4, pp. 527–539, 1996.