Model-Based Probabilistic Pursuit via Inverse Reinforcement Learning

Florian Shkurti, Nikhil Kakodkar and Gregory Dudek

***Abstract*— We address the integrated prediction, planning, and control problem that enables a single follower robot (the photographer) to quickly re-establish visual contact with a moving target (the subject) that has escaped the follower's field of view. We deal with this scenario, which reactive controllers are typically ill-equipped to handle, by making plausible predictions about the long- and short- term behavior of the target, and planning pursuit paths that will maximize the chance of seeing the target again. At the core of our pursuit method is the use of predictive models of target behavior, which help narrow down the set of possible future locations of the target to a few discrete hypotheses, as well as the use of combinatorial search in physical space to check those hypotheses efficiently. We model target behavior in terms of a learned navigation reward function, using Inverse Reinforcement Learning, based on semantic terrain features of satellite maps. Our pursuit algorithm continuously predicts the latent destination of the target and its position in the future, and relies on efficient graph representation and search methods in order to navigate to locations at which the target is most likely to be seen at an anticipated time. We perform extensive evaluation of our predictive pursuit algorithm over multiple satellite maps, thousands of simulation scenarios, against state-of-the art MDP and POMDP solvers. We show that our method significantly outperforms them by exploiting domain-specific knowledge, while being able to run in real-time.**

I. INTRODUCTION

This paper proposes a model-based predictive pursuit algorithm that enables a robot photographer to maintain visual contact with a subject whose motion is independent of the photographer. 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 scenario of intermittent loss of visibility, which is inevitable in practice[1].

We make use of navigation demonstrations by the target, which we exploit via Maximum Entropy Inverse Reinforcement Learning (IRL) [1] in order to estimate a reward function that expresses their preferences over the terrain features of the map. For example, the target might prefer to take paved roads, as opposed to grass or sand, in order to navigate to its destination. We can integrate that information in a behavior model for more accurate predictions, thus increasing the chance of re-establishing visual contact. Most importantly, feature preferences can be learned in a data-efficient way from as few as ten demonstrations, on a single map, and are typically transferable to other maps that share the same appearance and semantic classes. Using the reward learned from IRL and treating the target as an efficient navigator with respect to that reward, we identify a class of stochastic policies that can very efficiently simulate target behavior without having to perform planning at runtime for the purpose of prediction. This learned predictive and generative model for plausible paths marks an improvement over our prior work on this topic [2], which did not make use of prior knowledge of the target's navigation behavior. We assume that the robot photographer receives noisy observations of the target within a limited-range and field of view sensor, but has no other means of communication. We also assume that the target navigates purposefully while heading for a destination, and its motion is independent of the actions of the photographer, as opposed to being evasive or cooperative.

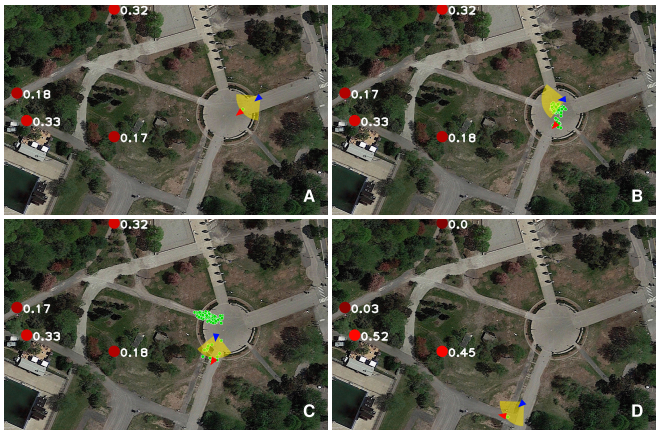

Fig. 1. Red denotes destinations. (A) The follower sees the target. (B) Visual contact is lost. Destinations are predicted and particles (green) start to diffuse. (C) Particles split on two different roads. The follower chooses one group of particles as more promising. (D) The follower re-establishes visual contact. Destinations on the top side of the map become unlikely.

Our method operates on both positive and negative observations and maintains a belief distribution on the target's possible locations. While the target remains unseen, the belief expands spatially and uncertainty grows. Our planning algorithm generates actions that shrink uncertainty or even completely *clear* it. The key enabling insight that allows us to plan over the vast array of potential target paths is the use of the learned navigation reward, which we use to reduce the set of possible hypotheses into a few possibilities that are compatible with the target's demonstrated navigation behavior and the set of available destinations.

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the NSERC Canadian Field Robotics Network

The authors are affiliated with the Mobile Robotics Lab, the Center for Intelligent Machines (CIM), and the School of Computer Science at McGill University in Montreal, Canada. {florian, nikhil, dudek}@cim.mcgill.ca

[1]In many variants of pursuit-evasion games, once line-of-sight is broken, the game ends.

II. RELATED WORK

Our problem is related to the large body of existing work on the theoretical bounds related to maintaining persistent tracking 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 [3].

Lavalle *et al.* [4] consider the problem of planning the trajectory of a single observer trying to maximize visibility of a fully-predictable or partially-predictable target. Generally, continuous visibility maintenance has been considered mostly for the case of adversarial targets. Such is the case for example in [5], [6], [7], [8]. 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. Nevertheless, it is worth mentioning 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 ([5], [6]).

The persistent tracking problem has also been addressed by approximate sampling-based POMDP solvers such as SARSOP [9], which extends a Monte Carlo Search Tree and maintains both the upper and lower bounds of the state-action value function. This enables pruning of provably suboptimal actions, and has been an idea that has been adopted in multiple subsequent solvers. The main limitation, however, when applied to our problem is that it works only on small-sized grids and runs in the order of minutes. Recent POMDP solvers such as DESPOT [10] alleviate some of these problems, by sampling a finite set of plausible future scenarios.

III. BACKGROUND

A. Maximum Entropy Inverse Reinforcement Learning

Making long-term and short-term predictions about the future behavior of a purposefully moving target requires that we know the instantaneous reward function that the target is trying to approximately optimize. Given a set of demonstration paths that trace the target's motion on a map, we can infer which parametric transformation of features of the map explains them, using the framework of Maximum Entropy Inverse Reinforcement Learning (IRL) [1]. In this framework the target is treated as planning in a Markov Decision Process (MDP) whose parametric reward function $r_\theta(s,a)$ is unknown, and to be estimated from the demonstrated paths.

We assume the existence of feature vectors $f(s) = [f_1(s), f_2(s), ..., f_N(s)]^\top$ where s is a location on the map. In our case, these N feature maps are boolean images that indicate semantic labels over satellite maps, such as roads, vegetation, trees, automobiles, buildings, as well as dilations of these boolean maps. We also assume that the instantaneous reward is a linear transformation of the features, $r_\theta(s,a) = \theta^\top f(s)$, parameterized by the weight vector θ. Maximum Entropy IRL factors the probability distribution over trajectories $\tau = (s_0, a_0, s_1, ..., a_{T-1}, s_T)$ as an instance of the the Boltzmann distribution:

$$p(\tau|\theta) = \frac{1}{Z(\theta)} \exp\left(\sum_{t=0}^{T} \theta^\top f(s_t)\right) \tag{1}$$

where $Z(\theta)$ is the partition function that acts as a normalization term. In this formulation, paths that have accumulated equal reward will be assigned the same probability mass, and differences in cumulative reward will be amplified exponentially. This distribution arises from the constrained optimization problem of maximizing entropy subject to first-moment matching constraints, which specify that the expected feature count at each state should be the same as the empirical mean feature count in the demonstration dataset. It is worth mentioning that Eqn. 1 assumes deterministic dynamics for the system being modeled, as well as no noise in the observation of the demonstrated trajectories.

Learning in the Maximum Entropy IRL model can be done by maximizing the log-likelihood function of the trajectories in the demonstration dataset S, as follows:

$$\theta^* = \underset{\theta}{\text{argmax}} \sum_{i=1}^{|S|} \log p(\tau_i|\theta) \tag{2}$$

$$= \underset{\theta}{\text{argmax}} \sum_{i=1}^{|S|} \sum_{t=0}^{T_i} \left(\theta^\top f(s_t^{(i)}) - \log Z(\theta)\right) \tag{3}$$

This optimization problem is convex for deterministic dynamics, so gradient-based optimization methods are sufficient to solve it. In our case we use the exponentiated gradient ascent method from [11] to obtain an estimate of the optimal reward weight parameters. The gradient of the scaled log likelihood function $\mathcal{L}(\theta) = \frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{t=0}^{T_i}\left(\theta^T f(s_t^{(i)}) - \log Z(\theta)\right)$ is

$$\nabla_\theta \mathcal{L}(\theta) = \frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{t=0}^{T_i} \left(f(s_t^{(i)}) - \nabla_\theta \log Z(\theta)\right) \tag{4}$$

$$= \frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{t=0}^{T_i} \left(f(s_t^{(i)}) - \frac{dZ(\theta)/d\theta}{Z(\theta)}\right) \tag{5}$$

$$= \frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{t=0}^{T_i} f(s_t^{(i)}) - \sum_{\tau} p(\tau|\theta) f_\tau \tag{6}$$

$$= \bar{f} - \sum_{s} p(s|\theta,\pi) f(s) \tag{7}$$

where $f_\tau = \sum_{s\in\tau} f(s)$ is the vector of features accumulated by the trajectory τ, $p(s|\theta,\pi)$ is the probability of visiting state s from a policy that solves the MDP with the induced reward function r_θ, and $\bar{f}$ is the empirical average of features in the dataset. The Maximum Entropy IRL algorithm for estimating the optimal reward weights is shown in Alg. 1.

Algorithm 1 MaxEnt IRL

1: $S:$ set of demonstrated trajectories
2: $\theta \leftarrow \theta_0$
3: **while** not converged **do**
4: Use value iteration to solve MDP(θ) for the optimal policy $\pi(a|s,\theta)$
5: Compute state visitation frequencies $p(s|\theta,\pi)$ using dynamic programming as in [1]
6: Compute gradient as in Eqn. 7
7: $\theta \leftarrow \theta \exp(\alpha \nabla_\theta \mathcal{L}(\theta))$ [11]

IV. TERRAIN-BASED PREDICTION MODEL FOR NAVIGATION

Once we learn the parameters of the target's reward function from a set of example paths we can make predictions about the target's actions. We treat the target as an efficient navigator through the environment, approximately optimizing its trajectory while trying to reach its destination. We want to account for the fact that the target's optimization process is approximate. In existing literature [12], [13], one way to perform approximate planning on MDPs, and also in particular in step 4 of Alg. 1, is done through *softmax value iteration*. Instead of the standard update rules in value iteration:

$$Q^*_\theta(s,a) = r_\theta(s,a) + V^*_\theta(s') \quad (8)$$

$$V^*_\theta(s) = \max_a Q^*_\theta(s,a) \quad (9)$$

where $(s,a) \rightarrow s'$ is the next state (from deterministic dynamics), the max operation is replaced by a softmax:

$$\widetilde{Q}_\theta(s,a) = r_\theta(s,a) + \widetilde{V}_\theta(s') \quad (10)$$

$$\widetilde{V}_\theta(s) = \operatorname*{softmax}_a \widetilde{Q}_\theta(s,a) \quad (11)$$

In order to sample paths on which the target can move towards one of the available destinations, we perform offline computation of one approximate value function per destination d, denoted here by $\widetilde{V}^{(d)}_\theta(s)$. This denotes the approximate value of the optimal path from location s to destination d on the map. We use similar notation for the state-action value function $\widetilde{Q}^{(d)}_\theta(s,a)$.

Given a value function for each destination, we can model the distribution of paths, conditioned on their endpoints, by comparing a path's accumulated value to the value of the optimal path on those endpoints:

$$p(\tau_{A\rightarrow B}|\text{start } A, \text{dest } C) \propto \frac{\exp\left(R_\theta(\tau_{A\rightarrow B}) + \widetilde{V}^{(C)}_\theta(B)\right)}{\exp\left(\widetilde{V}^{(C)}_\theta(A)\right)} \quad (12)$$

This model is useful in order to perform destination prediction, given the history of a path so far:

$$p(\text{dest } C|\text{start } A, \tau_{A\rightarrow B}) \propto p(\tau_{A\rightarrow B}|\text{start } A, \text{dest } C)p(\text{dest } C) \quad (13)$$

where $p(\text{dest } C)$ is the initial prior distribution on the destinations. In our case this is uniform, and the set of possible destinations is already known and of small size.

Offline pre-computation of the value function for each destination d also gives us access to a destination-conditional stochastic policy:

$$\pi_\theta(a|s,d) = \exp(\widetilde{Q}^{(d)}_\theta(s,a) - \widetilde{V}^{(d)}_\theta(s)) \quad (14)$$

which amplifies the advantage of action a at the given state. If iteratively applied from any location in the map, this stochastic policy typically leads to an attractor[2] at the given destination d. Examples of this are shown in Fig. 2. In effect

Fig. 2. 100 paths sampled by iterative application of the stochastic policy of Eqn. 14 at test time. Homotopically-distinct samples are possible, as long as the approximate value function is comparable between the two homotopy classes. In practice, we observed that this is not a typical event.

this gives us a fast and direct way of sampling paths from any location to a specific destination, as opposed to performing MCMC or other sampling methods on Eqn. 12. This has a crucial effect on our probabilistic pursuit algorithm, because it enables the generation of possible paths without any planning at runtime for the purposes of prediction, which allows the pursuer to dedicate its resources to formulating a navigation plan that will re-establish visual contact.

V. MODEL-BASED SINGLE-FOLLOWER PROBABILISTIC PURSUIT

We want to enable a single agent to persistently follow a target that is independent of the follower and which moves purposefully in an environment with obstacles. We assume the follower is equipped with: (a) knowledge of the map and its pose in it (b) demonstrated trajectories that the target has executed in the past in similar environments, revealing its preferences with respect to both potential destinations and routes that lead to them.

We pose the probabilistic pursuit problem as an integrated combination of planning and prediction of the future long-term behavior of the target's state. One way to see this is as a reinforcement learning problem, where the discrete state s_t of the pursuer-target system is defined as $s_t = [x_t, y_t, d_t]$ where x_t and y_t denote the planar poses (row, column, and yaw) of the target and the pursuer respectively, and d denotes the latent destination of the target. The instantaneous reward

[2] As long as the softmax value iteration has converged and the feature maps in the given environment are not conflicting.

is the indicator function of whether the follower sees the target or not. In particular, if we let FOV(y_t) denote the field of view of the follower at time t, then the reward function is $r(s_t, a_t) = \mathbb{1}_{[x_t \in \text{FOV}(y_t)]}$, where a_t is the action that the follower takes at that time.

The follower does not always have full information about the pose of the target, so it needs to maintain a belief $bel(s_t) = p(s_t|h_t)$ about the system's state s_t, given a history vector $h_t = [z_{1:t}, a_{1:t}]$, where $z_{1:t}$ is the history of sensor observations [3]. We assume that the follower has full knowledge of its own state as well as of the map, so the $bel(s_t)$ is really[4] $bel(x_t, d_t)$, which factors into $p(x_t|h_t)p(d_t|h_t)$. Therefore, this formulation allows us to describe the problem as a Partially-Observable Markov Decision Process (POMDP), more specifically as a Belief MDP, for the target's pose and a classification problem for its future destination. We assume that the initial distribution over destinations is uniform.

The POMDP transition model $\mathcal{T}_\theta(s_{t+1}, s_t, a_t) = p(s_{t+1}|s_t, a_t, \theta)$ is stochastic, and depends on the behavior of the target, parameterized here by the vector θ of reward weights, which we learn through inverse reinforcement learning. The transition model factors into $p(s_{t+1}|s_t, a_t, \theta) = p(x_{t+1}|x_t, d_t, \theta)p(y_{t+1}|y_t, a_t)p(d_{t+1}|d_t, x_t)$. We assume for simplicity that the follower's dynamics model is deterministic[5], so $p(y_{t+1}|y_t, a_t) = \mathbb{1}_{[(y_t, a_t) \to y_{t+1}]}$. We also assume that the destination of the target is a discrete latent variable that remains constant throughout the duration of the experiment, so $p(d_{t+1}|d_t, x_t) = \mathbb{1}_{[d_t = d_{t+1}]}$. This formulation is related to the POMDP-lite [14] and the Hidden Parameter MDP [15] formulations. The POMDP transition model is therefore $\mathcal{T}_\theta(s_{t+1}, s_t, a_t) = p(x_{t+1}|x_t, d_t, \theta)$ which is the learned stochastic policy in Eqn. 14.

One of the advantages of using behavior prediction models based on reward parameters learned through IRL, is that we have access to a very fast generative model $(x_{t+1}, y_{t+1}, z_{t+1}, r_{t+1}) \sim \mathcal{G}_\theta(x_t, y_t, d_t, a_t)$ that simulates the system. The more representative this simulator is of the target's actual behavior, the more certain the follower is about the value of each available action.

The POMDP transition model that we use accounts for the possibility of false negative detection errors (not recognizing the target when it is in the field of view of the follower), which is a common case for visual object detectors, particularly at large distances. On the other hand, we assume no false positive detections (recognizing the target when it is in fact not there). The observation model we use has false negative detection probability of q:

$$p(z_t = \emptyset | s_t) = \begin{cases} q & \text{if} \quad x_t \in \text{FOV}(y_t) \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

The pursuit problem of finding the target after it has escaped the follower's field of view can be formulated as finding a pursuit policy $\pi(a_{t+1}|h_t)$ that maximizes the follower's value function[6] $V^\pi(h_t) = E_{\pi, \mathcal{T}_\theta}[R_t|h_t]$, where $R_t = \sum_{i=t}^{t+T} \gamma^{i-t} r(s_t, a_t)$ is the discounted cumulative reward function. Solving this problem exactly is in general intractable, so sampling-based POMDP solvers, such as [16], [17] have been introduced in the literature.

Instead of resorting to general-purpose POMDP solvers that need to maintain some level of suboptimal exploratory behavior, we design an algorithm that exploits domain knowledge about the problem and uses a greedy planning behavior to determine which location to navigate to next, in order to maximize the chance of re-establishing visual contact with the moving target. We demonstrate in the results section that this approach outperforms state-of-the art POMDP solvers and provides a better solution to the problem. In the sections that follow we present an analysis of the different components of this algorithm, which is described in detail in Alg. 2.

A. Particle filter and Bayesian updates of the belief

We represent the belief $bel(x_t)$ about the target's location as a set of particles $x^{(i)}$, whose weights $w^{(i)}$ are updated by a particle filter. When a particle is created we associate it with a fixed speed, a fixed destination $d^{(i)}$ and a single path $\gamma^{(i)}$ to that destination. It travels along this path without deviation. The destination of the particle is sampled according to Eqn. 13. The sequence of states on the path are sampled iteratively through actions from the learned policy in Eqn. 14.

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. This is a crucial element because it allows us to deterministically predict where each particle is going to be at a specific time.

The transition model: The design choice of fixed speed and fixed path makes the problem of predicting where the particle is going to be in a few time steps purely deterministic. In our case the only stochasticity for a particle's transition model arises from the initial choice of destination $d^{(i)}$, the reference path $\gamma^{(i)}$ to it, and the fixed speed $v^{(i)}$ at which the particle traverses the path. After those stochastic samples have been drawn the motion of the particle is deterministic. So, the particle dynamics model, $p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t) = \sum_{\gamma, v} p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t, \gamma, v)p(\gamma, v|x_t^{(i)}, d^{(i)}, a_t)$,
which after the independence assumptions simplifies to $p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t, \gamma^{(i)}, v^{(i)})p(\gamma^{(i)}|d^{(i)})p(v^{(i)})$, depends entirely on the sampled path and speed. The conditional transition model $p(x_{t+1}^{(i)}|x_t^{(i)}, d^{(i)}, a_t, \gamma^{(i)}, v^{(i)})$ is completely deterministic [18] in our system.

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

[3] $z_t = \emptyset$ indicates not seeing the target. $z_t = x_t$ indicates it was seen

[4] This assumption allows us to decouple the tracking problem from that of Simultaneous Localization and Mapping, although this work is not affected should one need to model localization uncertainty.

[5] Note that the methods presented here do not rely on this assumption.

[6] Note that the follower's value function and policy are different than the target's value function and policy.

being evasive; second, that no particles are too slow because they might cause the follower to stay behind and try to clear up hypotheses that move very slowly, while the target progresses to its destination. Thus, we set:

$$p(v) = \begin{cases} \text{Uniform}[v_{\max}/2,\ v_{\max}] & b = 1 \\ \mathbb{1}_{[v=v_{\max}]}, & \text{otherwise} \end{cases} \quad (16)$$

where $b \sim \text{Bernoulli}(0.5)$ 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 negative errors as well as dependence to viewpoint, which is becoming less crucial given recent advances in object detection using supervised deep learning. The observation model we use in the particle filter is the one presented in Eqn. 15.

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 most other filters cannot provide.

B. Navigation

Our algorithm assumes for simplicity that navigation is done on a 2D plane. 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 [19]. 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 takes place.

One of the main advantages of using the structure of the GVG in order to facilitate topological and shortest path queries is that for realistic environments its size is usually small enough to be suitable for real time reasoning. The path obtained from executing a shortest-path algorithm on the GVG is not suitable for fast navigation, 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. This is essentially the Douglas-Peucker algorithm, referred to here as REFINE.

We use Yen's K-shortest paths algorithm [20] on the GVG to compute multiple topologically-distinct paths to a given goal, and then we iteratively refine each of the paths to reduce their length. We use this as a heuristic in order to avoid the scenario where the shortest path found on the GVG belongs to a homotopy class that does not contain the globally optimal path to the goal. After executing Yen's algorithm and refining the paths, we return the one with shortest length as an estimate of the shortest path.

Once we have the shortest path, we compute the optimal velocity and linear acceleration controls that will traverse the path in minimum time. We assume that the follower dynamics is omnidirectional $y' = \phi(y, a) = y + a\delta t$, which simplifies significantly optimal control for path execution.

C. 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 or model-predictive 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 obstacles. This is illustrated in Fig. 3. When visual contact is lost, planning replaces

Fig. 3. Paparazzi frames around the current target pose. They describe configurations from which the follower can observe the target. Trees are treated as obstacles. Yellow denotes the field of view. Better seen in color.

reactive following. We sample destinations and paths for the particles, as explained in the previous section, and we start the sampling and resampling steps in the particle filter according to incoming observations. During a sequence of incoming observations with no detections, the distribution of the destinations $p(d_t|h_t)$ is not updated; it is only updated when the target is detected.

The follower plans a path that will lead to a high-value location, as outlined in Alg. 2, 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 and the particles 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 to be able to compute the possible times and places at which the follower could visually intercept it. To this end we assume the functionality of a function called

T, τ = MIN-TIME-TO-VIEW(GVG, y_t, p), which is a trajectory planner in space and time that enables the follower to navigate from its current state y_t to the paparazzi frame p in minimum time, so as to bring itself in view of a potential target location, as quickly as possible. T is the time it will take to navigate to p, and τ is the trajectory that is planned. We implemented MIN-TIME-TO-VIEW for the case where the follower is an omnidirectional robot, by using iterative refinement of the shortest path obtained from the GVG, as described in the previous section[7].

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 as the next navigation waypoint the paparazzi frame that sees the particle with the highest ratio of probability over distance , and heads over to reach that paparazzi frame. This is a greedy nearest neighbor algorithm, and we term it NNm for "nearest neighbor pursuit along multiple homotopy classes", or more simply, "topological pursuit."

Algorithm 2 NNm($z_t,\ bel(x_t) = \{(x_t^{(i)}, w^{(i)})\}_{i=1...M},\ \ y_t$)

z_t : the follower's observation
$bel(x_t)$: the follower's belief about the target's pose
y_t : the follower's pose
D : set of potential destinations
if $z_t = x_t$ i.e. the target is visible **then**
  $bel(x_{t+1}) \leftarrow$ PF-UPDATE(GVG, $bel(x_t), z_t, y_t, D$)
  $\bar{x}_t \leftarrow$ desired paparazzi frame based on x_t (e.g. behind the target)
  $\delta\theta, \delta r \leftarrow y_t - \bar{x}_t$
  $a_t \leftarrow$ feedback control from $\delta\theta, \delta r$
  $y_{t+1} \leftarrow \phi(y_t, a_t)$
  Update $p(d_t|h_t)$ as in Eqn. 13
  NNm($z_{t+1},\ bel(x_{t+1}),\ y_{t+1}$)
else
  for $i = 1...M$ **do**
    Sample destination $d_t^{(i)} \sim p(d_t|h_t)$ as in Eqn. 13
    Sample path $\gamma^{(i)} \sim p(\gamma|d_t^{(i)})$ as in Eqn. 14
    Sample speed $v^{(i)} \sim p(v)$ as in Eqn. 16
    Assign path $\gamma^{(i)}$ and speed $v^{(i)}$ to particle $x_t^{(i)}$
  $T_{ji}, \tau_{ji} \leftarrow$ MIN-TIME-TO-VIEW(GVG, y_t, p_{ji}), $\forall j, i$ for paparazzi frames p_{ji} sampled along $\gamma^{(i)}$
  $\Delta l_{ji}, \Delta\theta_{ji} \leftarrow$ total length and rotation in the path π_{ji}
  $I \leftarrow (i,j)$ such that particle $x_t^{(i)}$ is predicted to reach $p_{ji} \in \gamma^{(i)}$ close to follower's arrival time T_{ji}
  $i^*, j^* \leftarrow \underset{(i,j)\in I}{\text{argmax}}\ \frac{w^{(i)}}{\Delta l_{ji} + \Delta\theta_{ji}}$
  while not reached $p_{j^*i^*}$ **do**
    $bel(x_{t+1}) \leftarrow$ PF-UPDATE(GVG, $bel(x_t), z_t, y_t, D$)
    $a_t \leftarrow$ next action in follower's trajectory $\tau_{j^*i^*}$
    $y_{t+1} \leftarrow \phi(y_t, a_t)$
    $t \leftarrow t + 1$
  Go to step 5

VI. RESULTS

A. Setup

We set up a simulation environment in order to benchmark our algorithm against existing MDP and POMDP solvers. This environment includes 20 different aerial images, with top-down view, shown in Fig. 7, each of which covers areas where the dominant semantic labels of the terrain are: roads, vegetation, trees, buildings, and vehicles.

Each map was annotated offline by human annotators, who also provided examples of target trajectories to pre-specified destinations, from various starting points on the aerial image. The 280 target trajectories, which were demonstrated across the range of all maps, expressed preference for roads and vegetation, avoiding trees and buildings, which were treated as obstacles.

We compared our algorithm with variations on the three following baseline methods: (i) full-information pursuit, (ii) UCT [21], and (iii) POMCP [16]:

(i) Full-information pursuit refers to the variant of the problem, where the follower knows *a priori* the trajectory of the target, so it can make use of techniques similar to Model Predictive Control (MPC), which recompute a control sequence at each time step, based on known dynamics, and select the best action at each time step, discarding the rest of the plan. Following similar rationale, at each time step, our full-information pursuit algorithm replans a trajectory from the current state of the follower to the closest valid paparazzi frame for the target, and executes the first action prescribed by that trajectory, similarly to MPC. The performance of the full-information pursuer provides an upper bound on the performance of pursuit methods, in which the state of the target may be latent.

(ii) UCT is a sampling-based MDP solver that grows a Monte Carlo Search Tree, using the Upper Confidence Bound (UCB1) rule to trade off exploration vs exploitation. We apply UCT to the variant of our problem in which the state of the target is revealed to the follower at each time step, but the future states of the follower-target system are stochastic, according to the latent preferences of the target, such as destination, route, type of terrain etc.

(iii) POMCP is one of the state-of-the-art sampling-based POMDP solvers. Like UCT, it also performs Monte Carlo Tree Search. It differs from UCT by the fact that each state node in the tree contains action edges that lead to observation nodes, as opposed to other state nodes. It avoids propagating and updating the particle filter belief at each simulation path through the tree, by sampling a particle from the belief according to its weights in the filter, and simulating

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

its evolution through the tree. POMCP addresses the same version of the problem that our method does. It is worth noting that both in our method as well as in POMCP we use the same particle filter settings, and the same learned IRL reward weights, so the prediction mechanism for the target behavior is identical among the two methods[8].

We compared our methods against these baselines across approximately 4500 experiment scenarios in total, each of which was repeated under the exact same settings across 5 different episodes, to get an estimate of variance in the outcomes of each scenario. These experiment scenarios included variations on:

- the start and destination
- the follower's / target's maximum speed
- the time limit allowed to UCT and POMCP to generate a single action
- the probability of false negative detections
- the speed profile of the target, as it executes its trajectory

For the performance of UCT and POMCP baselines we report the success rate for planning each action within 5 seconds; this of course implies offline operation for these methods, whereas our topological pursuit method runs in real-time at 1Hz.

B. Findings

Our experiments demonstrate that our method outperforms both POMCP and UCT when the follower and the target share the same maximum speed. This is better shown in Fig. 4, which demonstrates the duration of successful pursuit (i.e. how long the follower sees the target) relative to the full-information pursuit, which achieves 100% success rate, managing to maintain visual contact with the target during the entire duration of the experiment. In Fig. 4 the probability of false negative observations is set to 0.5, which means that half the time, when the target is in the field of view of the follower, it is not detected, in addition to other times when visual contact is necessitated by the structure of the environment. Our method scores on average approximately 40%, whereas POMCP is at 15% and UCT is under 5%. The main reason why POMCP and UCT perform poorly in this regime of equal maximum speed for the follower and the target is that any suboptimal pursuit actions performed early on in an episode have critical consequences throughout its duration, in the sense that there is little time to recover visual contact if the target is moving at full speed. POMCP and UCT are prone to committing such suboptimal moves due to their tendency to explore and be optimistic in the face of uncertainty, whereas our algorithm is greedy and focused on navigating to a single destination at a time, so it does not make use of exploratory actions. Additionally, without an explicit penalty term in the design of the reward function, actions produced by UCT and POMCP tend to have high variance as well, which is problematic for deployment in a real vehicle.

[8]Also for both UCT and POMCP the Upper Confidence Bound exploration constant was set to $\sqrt{2}$, and the default leaf expansion policy is the uniform distribution.

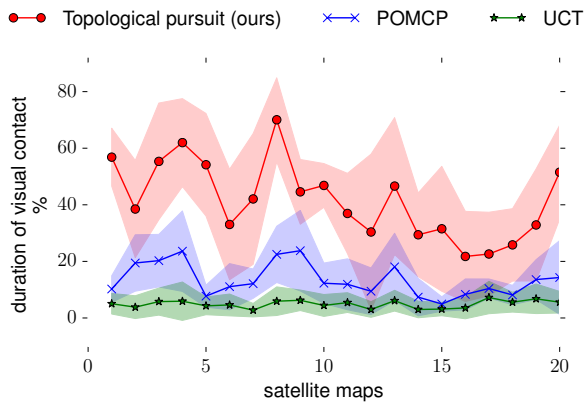

Fig. 4. Relative duration of visual contact across the available test maps. Bars denote 1σ standard deviation. Averages were taken with respect to target trajectories and episodes. Colors denote algorithm types: topological pursuit (red, our method), POMCP (purple) and UCT (green) algorithms (See text).

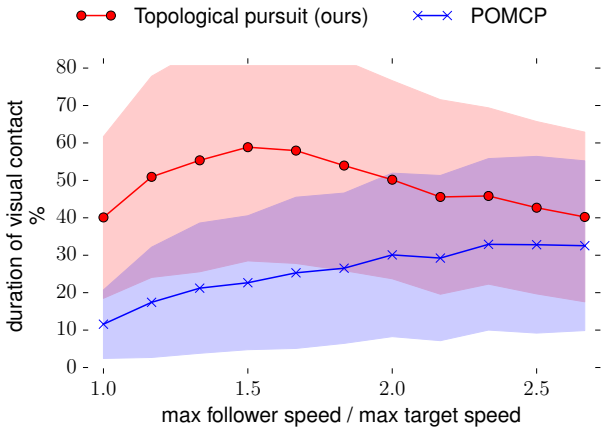

Fig. 5. Relative duration of successful pursuit as a function of maximum speed advantage of the follower. Bars denote 1σ standard deviation. Averages were taken with respect to maps, target trajectories, and episodes.

As the follower becomes faster relative to the target, suboptimal actions for exploration become less critical because there is time for the pursuer to recover visual contact with the target. This is better illustrated in Fig. 5, which shows that the tracking performance of POMCP gradually improves as the follower becomes twice as fast as the target. This is not surprising as POMCP is an anytime algorithm. Our method, however, does not demonstrate the same property. One hypothesis that might explain this has to do with step 21 of Alg. 2, where we are searching over points along the sampled particle paths, such that the particles are projected to arrive there approximately at the same time as the follower, which means that they are suitable to become rendezvous locations. As the relative speed of the follower increases, the range of locations that are reachable by the target at the same time as the follower becomes smaller and smaller, which means that the number of options for the next best paparazzi frame to navigate to is gradually reduced. As the number of good hypotheses for rendezvous locations becomes more limited the quality of pursuit deteriorates.

It is worth mentioning, however, that many practical pursuit scenarios belong in this regime, where the pursuer does not need to be more than twice as fast as the target (aerial photography with vehicles carrying heavy camera equipment, or diving assistant underwater robots). In this neighborhood of relative speed advantage our algorithm still outperforms Monte Carlo Tree Search-based methods.

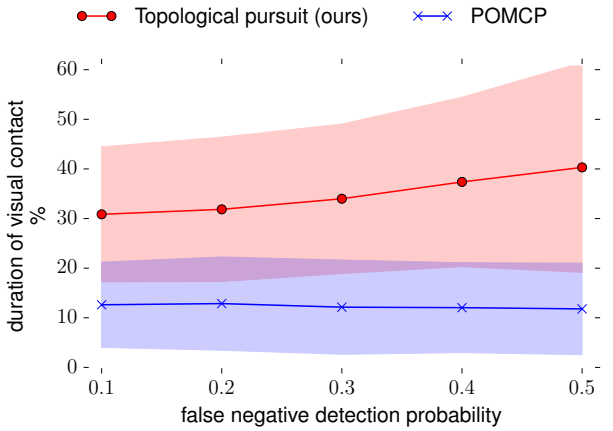

Fig. 6. Relative duration of visual contact as a function of the false negative detection rate. Bars denote 1σ standard deviation. Averages were taken with respect to maps, target trajectories, and episodes.

The third axis of variation that we examined is the false negative detection rate, whose effect is shown in Fig. 6. Our method is more robust compared to POMCP, with more than 3x longer tracking on average, when the observation model is prone to false negative errors with probability up to 0.5, which is significant, particularly in vision-based tracking and detection at large distances, where the target occupies a small part of the image.

VII. CONCLUSIONS

We presented a real-time predictive pursuit algorithm that handles the case of loss of visual contact during tracking, and plans to re-establish it. Our method uses a behavior model of the target, learned from navigation demonstrations and Inverse Reinforcement Learning, in order to predict the target's short-term and long-term actions. This model is easy to sample from, and we use it to maintain predictions of the target's latent destination, as well as to forecast the route it will take to said destination, thereby being able to generate plausible hypotheses about where the target is going to be and when. Our physical search method exploits this information to devise a search plan, which aims to recover the target in the follower's field of view. We validate our method by comparing it to state-of-the art MDP and POMDP solvers, and we show that it outperforms them by significant factors.

Fig. 7. Some of the satellite maps and the human-annotated target paths that we used in our test set. Red denotes destinations.

REFERENCES

[1] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *23rd National Conference on Artificial Intelligence - Volume 3*. AAAI, 2008, pp. 1433–1438.

[2] F. Shkurti and G. Dudek, "Topologically distinct trajectory predictions for probabilistic pursuit," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5653–5660, September 2017.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP Planning with Regularization," in *Advances in Neural Information Processing Systems 26*, 2013, pp. 1772–1780.

[11] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, *Activity Forecasting*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 201–214.

[12] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3931–3936.

[13] A. Tamar, S. Levine, and P. Abbeel, "Value iteration networks," *CoRR*, vol. abs/1602.02867, 2016. [Online]. Available: http://arxiv.org/abs/1602.02867

[14] M. Chen, E. Frazzoli, D. Hsu, and W. S. Lee, "POMDP-lite for Robust Robot Planning under Uncertainty," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1602.04875

[15] F. Doshi-Velez and G. Konidaris, "Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations," in *International Joint Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 1432–1440.

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

[17] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *AAMAS*, vol. 27, no. 1, pp. 1–51, Jul 2013.

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

[19] H. Choset, "Incremental Construction of the Generalized Voronoi Diagram, the Generalized Voronoi Graph, and the Hierarchical Generalized Voronoi Graph," in *CGC Workshop on Computational Geometry*, October 1997.

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

[21] L. Kocsis and C. Szepesvári, *Bandit Based Monte-Carlo Planning*. Springer Berlin Heidelberg, 2006, pp. 282–293.