Air-to-Ground Surveillance Using Predictive Pursuit

Sourav Dutta
Department of Computer Science,
University at Albany
Albany, NY 12203
Email: sdutta2@albany.edu

Chinwe Ekenna
Department of Computer Science,
University at Albany
Albany, NY 12203
Email: cekenna@albany.edu

***Abstract*— This paper introduces a probabilistic prediction model based on Markov decision process to improve tracking time and location detection accuracy in an air-to-ground robot surveillance scenario. While most surveillance algorithms focus mainly on controls of an unmanned aerial vehicle (UAV) and camera for faster tracking of an unmanned ground vehicle (UGV), this paper proposes a way of minimizing detection and tracking time by applying a prediction model to the initial observed path taken by the UGV. A novel tracking algorithm has been designed that applies Markov prediction model on the trajectories of a UGV derived from some unknown path planning algorithm, to predict chosen motion planning algorithm used by the UGV. We present a pursuit algorithm that addresses the problem of target localization by combining prediction of used planning algorithm by the target, and application of the same planning algorithm to predict its future trajectories. Our results show a high predictive accuracy based on a final position attained by the UGV and the location predicted by our model.**

Keywords: Localization and Mapping, Machine Learning, Motion and Path Planning, Probabilistic Reasoning, Markov Decision Process.

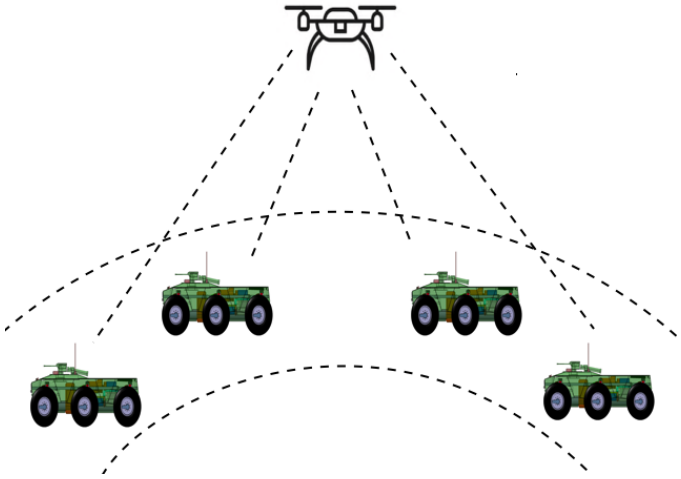

Fig. 1: **Air-to-Ground Surveillance** (UAV and UGV design referenced from [1], [5])

I. INTRODUCTION

Surveillance by drones is an emerging field of study due to the reduction in the cost of agile and intelligent drones in the market. Previously, surveillance was being performed by a network of stationary security cameras [13]. Recent advances have surveillance done by drones with onboard cameras, having some computer vision algorithm running to detect invaders or track specific objects [17]. The applications of drone surveillance are practically limitless including police supervision, security surveillance, military, agriculture, etc. They can also be used to land an autonomous aerial vehicle on an autonomously moving aircraft carrier [11].

In a surveillance scenario, there are always two parties, one invader and another pursuer. The motive of the invader is to reach a certain goal point without getting caught. The motive of the pursuer is to stop the invader before it reaches the goal point. In some previous work, the invader was either operated manually or human [14], [24]. But now, with sophisticated robot platforms and algorithms, this practice of sending manually operated robots or humans in a high-security zone is being replaced by sending autonomous vehicles. In this paper, we assume that only an autonomous ground vehicle is used as an invader, and that, it is running and learning what planning algorithm to use based on the environment.

Upgrading the camera and the flight controls of the UAV (pursuer) is not always sustainable. The pursuer will have to keep updating its camera and controls, just to keep up with advances in the technology of the UGV (invader). This calls for a change at the algorithmic level. Most of the existing surveillance algorithms require partitioning of the environment into grids and maximizing the probability of the existence of the target in one single grid by using sensor data. This partitioning is not scalable if the environment is of a larger size since the individual partitions will also be of a considerably large surface area, thereby increasing search space. Our motivation for this algorithm is from a standpoint where both cost-effectiveness and low maintenance can be achieved for the pursuer.

In general, the air-to-ground surveillance problem can be considered as a variation of a pursuit-evasion game. In [6], a new approach to a pursuit-evasion game has been studied from a mathematical standpoint. Nash equilibrium [16] has been used to show that in a pursuer-invader scenario, the safety of both the invader and the pursuer can be achieved using equal payoffs for every pair of adopted strategies by the two parties (invader and pursuer).

In this paper, we discuss a new way of surveillance, where pursuing an invader involves predicting its trajectory in

the future from its observed initial trajectories. Assuming that the invader is using an adaptive reinforcement learning algorithm [7], [8] to reach from start to goal location in the environment, we apply our algorithm based on Markov decision process (MDP) on a set of possible motion planning algorithms that the invader may use to determine the one strategy that best suits the environment. We know that this is a restrictive assumption in practice, nonetheless, we are interested in seeing if it is possible to predict the strategies of the invader as an important first step towards learning to predict movement without such a restrictive set. As a relaxation to this restriction, we have the option of including any motion planning algorithm in the set as and when we need to.

We have developed a novel predictive pursuit algorithm by applying MDP on observable trajectories of the invader, that involves predicting the motion planning strategy of the invader, and thereby, predicting its future trajectories to prevent it from reaching its goal, by following the invader during its learning phase, as shown in Figure 1.

Our results show that the predictive pursuit algorithm has high accuracy in determining the trajectory of the invader, because, it can predict the motion planning strategy of the invader, and the invader is localized by applying the predicted strategy on the given environment.

II. Related Work

In the following sections, we will discuss some generic problems in using sensors to track a ground target from the air, a brief description of Markov chain and Markov processes, and exploitation of some properties of adaptive motion planning strategies to pro-actively determine the position of a moving target from air to ground.

A. Air-to-Ground Surveillance Using Sensors and Tracking Algorithms

In some recent research experiments to track a UGV using a UAV [11], onboard cameras and sensors have been used, and the current position of the UGV has been estimated to land the UAV. A monocular vision-based approach for cooperation has been used between a UAV and a UGV. The images taken by the UAV were captured by an onboard low-cost camera which was processed by a computer on the ground after being transmitted over a wireless channel.

In [19], usage of radar technology in an air-to-ground surveillance has been discussed. A transmitting antenna has been used to illuminate a region of interest with a noise waveform. The illuminated scene has been treated by the receiver as a linear system which filtered the transmitted signal, and adaptively estimated the unknown filter coefficients.

In [15], a generic problem in trajectory planning for drone has been discussed which arises due to faulty sensors or flight control. A risk-aware trajectory has been generated to ensure the safety of the drone. In actual practice, the UAV needs time to find a safe landing position, and by that time, the UGV may shift to a new position. A preemptive prediction of the location of the UGV is necessary, so that, the UAV finds sufficient time to land safely, yet it doesn't lose the UGV in its pursuit.

In [23], a cooperative air to ground surveillance has been discussed that introduces path planning using sensor data from a UGV and a UAV. Visual occlusions due to obstacles in the environment have been taken into account. A dynamic grid to break down the problem environment and localize the target has been used. The sum of probabilities of detection of a moving target over a finite look-ahead horizon has then been maximized using a Bayesian filter over sensor data from the UGV and the UAV.

In [18], another cooperative air to ground surveillance has been introduced that applies a second-order Markov chain on sensor data from a UGV and a UAV. The environment has been broken down into occupancy grid, and a path planning algorithm has been used to maneuver the UAV and the UGV to configurations where the target could have been detected with high probability.

B. PRM and Adaptive Strategies

In this section, we discuss the possible strategies that the UGV may adopt to move from start to goal within the environment. A class of motion planning algorithms called sampling-based motion planning [9] exists for a number of planning strategy problems. This algorithm has two phases, the learning phase, and the query phase. In the learning phase, it applies an efficient and fast local planner to construct a roadmap graph consisting of collision-free nodes within the environment. In the query phase, any given start and goal nodes within the environment are connected by joining the nodes within the roadmap using an adaptively selected strategy.

Probabilistic Roadmap Algorithm (PRM) [12] is a new-generation of motion planning algorithms that have been widely used for sampling and mapping robot configuration spaces. Some PRM variants exist [2], [4], [22], where choosing an appropriate strategy is dependent on the given problem environment. For heterogeneous spaces [20], it is almost infeasible to hand-select the best-suited strategy for the environment.

Based on all popularly known PRM algorithms, a general connection framework called adaptive neighbor connector (ANC) [7] has been developed using reinforcement learning, which adaptively selects a neighbor finder in conjunction with a distance metric from a candidate set of options. We will be referring to these combinations as $connectors$ from here on in this paper. In another variant of PRM [8], a method called Hybrid PRM has been introduced (referred to as adaptive PRM in this paper), where, instead of learning feasible neighbor connection methods, the Sampling Strategies were selected from an adaptive learning framework.

The ANC [7] can be configured to run with either multiple options for connector methods or multiple options for distance metrics or both. For our UGV, we chose to use the most widely used connection method, K-Closest with a brute force approach, which returns the K closest neighbors to a node based on some distance metric, where K is some small constant. In general, the following distance metrics are used for ranking each connection method based on the cost and

profit from the learning phase of the ANC:

- **Euclidean**: This metric gives equal weighting for all dimension.

$$\delta(p,q) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+...+(p_n-q_n)^2} \tag{1}$$

  where n is the number of dimensions of the points p and q.
- **Scaled Euclidean**: This metric is a variant of the Euclidean metric:

$$\delta(p,q) = \sqrt{s*(posMag)^2+(1-s)*(oriMag)^2} \tag{2}$$

  where $posMag$ is the Euclidean distance of the positional dimensions between p and q, $oriMag$ is the Euclidean distance of orientation dimensions between p and q, and s is a weighing parameter.
- **Center of Mass**: This metric is the Euclidean distance between the centers of mass of the robot at positions p and q.
- **Manhattan**: This metric gives us an overhead distance along a horizontal line or vertical line or both combined. The Manhattan distance is simply the sum of the components in each dimension.
- **Minkowski**: This metric is a generalization of both the Euclidean distance and the Manhattan distance in a normed vector space.

In a similar fashion to ANC, Hybrid PRM also has the option to be configured to run with multiple candidate options for sampling strategies. We have used these following sampling strategies as options in the HybridPRM:

- **Obstacle-Based PRM (OBPRM)**: In OBPRM [2], the roadmap nodes are sampled on or near obstacle surfaces, which in turn, improves the quality of roadmaps for crowded environments.
- **Medial Axis PRM (MAPRM)**: In MAPRM [22], some randomly generated configurations are retracted onto the medial axis of the free space which increases the number of nodes found in small volume corridors.
- **Gaussian PRM**: The Gaussian PRM [4] is based on the concept of blurring, used in image processing. It generates sampled nodes in difficult regions using simple intersection tests in the workspace. It is suitable for many different motion planning problems.

In this paper, we use the ANC and the adaptive PRM alternatively. Collectively, we refer to these two methods as *Adaptive Method* from here on. We mention Adaptive Method where either of the adaptive PRM or the ANC will be used, but not both together.

C. Markov Chain and Partially Observable Markov Decision Process

A Markov chain is a stochastic model of events, where the probability distribution of the next system state is dependent on the current state. In probability theory, an MDP is a stochastic process that satisfies the Markov property. A discreet time-homogeneous MDP is one where each system state is a random variable in discrete state space and the transition probabilities to move from one system state to the next is independent of time.

In [3], a generic problem in the discretization of configuration space and time for testing motion planning algorithms have been discussed. This discretization has led to inaccurate trajectories. As a solution to this generic problem, a partially observable Markov decision process (POMDP) approach was used. It helped in making decisions where sensor readings were either limited or inaccurate.
In [21], another approach to motion planning using MDP under continuous domain has been discussed. A POMDP framework has been followed by extending a Kalman filter [10] to approximate a nominal trajectory through belief space.
Inspired by the potential benefits of MDP in the field of motion planning, we have developed an algorithm for drone surveillance that can predict future trajectories of an invader by observing the initial strategies adopted by the invader.

III. Methodology

We assume that the invader (UGV) is running an Adaptive Method to reach from start to goal within the environment. Initially, we observe the path assumed by the UGV, and separately run all the planning strategies that the Adaptive Method may use. Then we create a Markov chain of probabilities and some initial probabilities to predict the one sampler or connection method being used by the Adaptive Method, and in turn, the invader.

A. Problem Description

In an MDP, we have a set of given possible states $S = (S_0, S_1, S_2, ..., S_n)$ with the Markov property, where n is a finite number. Markov property states that the probability of moving to the next state depends only on the present state and is independent of the previous states.

$$\begin{aligned} &p(max-index(S_{n+1}) = s|max \\ &\quad -index(S_1) = s_1, max \\ &\quad -index(S_2) = s_2, ..., max \\ &\quad -index(S_n) = s_n) = p(max \\ &\quad -index(S_{n+1}) = s|max - index(S_n) = s_n) \end{aligned} \tag{3}$$

where $(s_1, s_2, ..., s_n)$ are the indices of individual samplers or connectors in our list of samplers or connectors that may have been used by the Adaptive Method at each timestamp. The function $max - index(S_i)$ gives the index of the maximum value of S_i.

In a homogeneous discreet-time Markov chain, like the one used in our method, the probability of the transition is independent of n.

$$\begin{aligned} &p(max-index(S_{n+1}) = x|max \\ &-index(S_n) = y) = p(max \\ &-index(S_n) = x|max - index(S_{n-1}) = y), x, y \in (s_1, s_2, ..., s_n) \end{aligned} \tag{4}$$

In our prediction model, each state of the system is a probability distribution of the possible samplers or connectors used by the Adaptive Method. The number of samplers

or connectors used by the Adaptive Method decides the number of states we have in our Markov chain ($n = 2 * NS$, where NS = the number of samplers or connectors used by the Adaptive Method). The initial state probabilities are determined by the occurrences of each individual method during the learning phase of the predictive pursuit method. We use $n = 2 * NS$ observations to train our prediction model to make sure that sufficient timestamps have passed after the learning phase of the Adaptive Method that is being used by the UGV. This is because, in the learning phase of the Adaptive Method, it may switch between the samplers or connectors as per the nature of the environment (since, each sampler or connector is suitable for certain type of environments).

$$S_0(j) = \frac{occurrence[j]}{n}, \forall j \in \{1, 2, ..., NS\} \tag{5}$$

In order to determine the state which can confirm the usage of a particular sampler or connector by the Adaptive Method, we try to maximize the probability of occurrence of the one single sampler or connection method that has been used by the Adaptive Method for the entire environment after its learning phase.

$$S_i = S_{i-1} * P, i \in \{1, 2, ..., n\} \tag{6}$$

or

$$S_i = S_0 * P^{i-1}, i \in \{1, 2, ..., n\} \tag{7}$$

where,

$$\begin{aligned} p(x, y) &= p(max(S_i) \\ &= y|max(S_{i-1}) \\ &= x), i \\ &\in \{1, 2, ..., n\}, x, y \\ &\in \{1, 2, ..., NS\} \end{aligned} \tag{8}$$

Our target is to minimize the number of times P is multiplied to S_0 in order to get a probability value greater than a certain threshold. This will reduce the time required for learning phase of our predictive pursuit, and eventually, lead to a faster target localization.

Note that, since each of the system state is a probability distribution, we have

$$\forall i \sum_{j=1}^{NS} S_i(j) = 1, i \in \{0, 2, ..., n\} \tag{9}$$

Algorithm 1 Predictive Pursuit

Input. $Samplers$ is a list of all Sampling Strategies or Connectors used in learning phase of the Adaptive Method.
Input. $observedPath$ is the observed path file from the Adaptive Method.

1: **for** $i = 1$ to NO **do**
2: $observedNode \leftarrow followUGV(observedPath)$
3: **Initialize:** $dist(1, 2, ..., NS) \leftarrow \infty$
4: **for** $j = 1$ to NS **do**
5: $seeds \leftarrow PRM(Samplers(j), observedNode)$
6: $dist(j) \leftarrow distance(seeds, observedNode)$
7: $idx \leftarrow$ index of $minimum(dist(1, 2, ..., NS))$
8: $samplerSequence(i) \leftarrow Samplers(idx)$
9: $samplerOccurrence(i) \leftarrow samplerOccurrence(idx) + 1$
10: $[S_0, P] \leftarrow constructProbability()$
11: **while** $countIter \leq maxObservations$ **do**
12: $S_{next} \leftarrow S_0 * P$
13: **for** $k = 1$ to NS **do**
14: **if** $S_{next}(k) \equiv threshold$ **then**
15: $flag \leftarrow True$
16: $samplerIndex \leftarrow k$
17: **break**
18: $countIter \leftarrow countIter + 1$
19: **if** $flag \equiv True$ **then**
20: **break**
21: **if** $flag \equiv True$ **then**
22: **for** $l = 1$ to $countIter$ **do**
23: $seeds \leftarrow PRM(Samplers(samplerIndex), observedNode)$

Output. $seeds$

B. Algorithm 1: Predictive Pursuit

Our predictive pursuit algorithm (Algorithm 1) predicts the one sampler or connector that is being selected by the Adaptive Method by creating a Markov chain of probabilities, and then, applies that sampler or connector to predict the next two coordinates where the UGV is supposed to be at. This forward prediction is to give the UAV sufficient time to land from a considerable height so that the UGV doesn't shift to a new location.

At the beginning of the predictive pursuit algorithm, we initialize a few parameters as listed in Table I.

At each timestamp, the robot position from each individual connection method is observed, and the robot position from an adaptive connection strategy is also recorded. In Step 13 (algorithm 1), it runs a basic PRM algorithm with the individual sampler or connection method $Sampler(j)$ and stores the robot position obtained from it in $seeds$. Euclidean distances between each robot position from the individual connectors or samplers ($seeds$), and that from the Adaptive Method ($observedNode$) is calculated and stored in a distance matrix $dist$. In Step 3 (algorithm 1), we initialize $dist$ with values that are large enough to not fit in the boundaries of the given environment (∞). In case a particular sampler can't find a solution in the given environment and the PRM code doesn't

TABLE I: **Initialized Parameters for Algorithm 1**

Parameter Name	**Value**	**Description**	**Justification**
$threshold$	0.5	Used to determine the samplers or connectors that the invader uses	Number of samplers/connectors being greater than 3, this value is sufficient for a prediction.
NS	$length(Samplers)$	Number of samplers or connectors used in the learning phase of Adaptive Method	Since, $Samplers$ is the list of samplers/connectors, the length of the list is the desired value.
NO	$NS * 2$	Number of suggested observations before making a prediction	This value gives the Adaptive Method sufficient timestamps to try all samplers/connectors in the $Samplers$ list.
S_0	$1/NS$	List of initial state probabilities	This value sets equal probabilities to all the samplers or connectors before the learning phase of the predictive pursuit.
S_{next}	$1/NS$	List of next state probabilities	This value sets equal probabilities to all the samplers or connectors before the learning phase of the predictive pursuit.
P	0	A $2D$ transition matrix of dimensions $NS * NS$ that is a map of state transitions of our Markov chain	A 0 value indicates a no transition between two states which will change as the predictive pursuit completes its learning phase.
$samplerSequence$	null	List that holds the sequence of samplers that we have estimated has been used during the learning phase of the Adaptive Method	Null value indicates that initially no sampler/connector is there in the list.
$samplerOccurrence$	0	List that holds the number of times each sampler is estimated to have been used during the learning phase of the Adaptive Method	0 value indicates that initially no sampler/connector has been used.
$samplerIndex$	-1	Predicted index of the sampler that the invader uses	This value indicates that initially no sampler/connector has been predicted.
$flag$	$False$	Boolean variable that indicates whether or not the sampler or connector used by the Adaptive Method has been determined or not	This value indicates that initially sampler/connector has not been determined.
$maxObservations$	10	Indicates the maximum number of attempts to find the sampler with a threshold probability by multiplying $S_0 * P$	Since we have already exhausted the timestamps for the learning phase of the Adaptive Method, we can expect the probability of one particular sampler/connector to be approaching the threshold.
idx	-1	Holds the index of the minimum distance in $dist$ at each iteration during the learning phase of the predictive pursuit algorithm	This value indicates that none of the candidate options among the samplers or connectors have been selected yet.

return any coordinate in Step 13, that particular sampler will receive a zero probability in the Markov chain. From this $dist$ matrix, an estimation is made as to which strategy is closest to the one observed from the Adaptive Method. This process is repeated until a fixed number of observations have been made. We have initialized this fixed number as twice the number of individual strategies, considering the fact that the Adaptive Method runs all individual strategies in its learning phase.

We record the count and sequence of occurrences of the individual strategies in the Adaptive Method by comparing the coordinates from the individual samplers or connectors and that from the Adaptive Method at each timestamp. Then the transition matrix P and the initial state probabilities S_0 are created using Algorithm (2).

The construct probability algorithm returns the transition matrix P and the predictive pursuit algorithm then matrix multiplies the state matrix S_0 of the previous iteration (timestamp) to create a new state matrix S_{next}. Whenever we find a probability beyond a certain threshold for any element in the prediction matrix S_{next}, we are certain that this strategy is the one that has been used so far and will be used henceforth. Then we can predict the next move of the UGV and track it down.

The predictive pursuit algorithm stops at a point when the predicted probabilities of one of the strategies in the current state S have at least a $threshold$ value and one particular sampler may be used for making the prediction. The variable $countIter$ gives the possible timestamp at which the state probabilities of our Markov chain will certainly be able to predict the sampler or connector used by the adaptive learning framework (by comparing with the threshold). When at Step 24, the predictive pursuit has already made a decision as to which sampler has been used by the Adaptive Method being run by the invader (UGV). Then the pursuer goes for forward predicting the next $countIter$ possible locations of

the invader by running the predicted sampler or connector on the same environment using the last $observedNode$ of the invader as start location.

Algorithm 2 Construct Probability

Input. $samplerSequence$ is the sequence of estimated samplers from observing the Adaptive Method.

Input. $samplerOccurrence$ is the count of occurrences of each sampler from observing the Adaptive Method.

Input. $Samplers$ be a list of all Sampling Strategies or Connectors used in learning phase of the Adaptive Method.

1: **for** $i = 1$ to $NO - 1$ **do**
2: $fromIdx \leftarrow$ index of sampler $samplerSequence(i)$ in $Samplers$.
3: $toIdx \leftarrow$ index of sampler $samplerSequence(i+1)$ in $Samplers$.
4: $P(fromIdx, toIdx) \leftarrow P(fromIdx, toIdx) + 1$
5: **for** $i = 1$ to NS **do**
6: $sumRow \leftarrow \sum_{j=1}^{i}(P(i,j))$
7: **for** $j = 1$ to NS **do**
8: $P(i,j) \leftarrow P(i,j)/sumRow$
9: **return** [S0,P]

C. Algorithm 2: Construct Probability

The construct probability algorithm (Algorithm 2) receives the sampler occurrences and sampler sequence from the predictive pursuit algorithm and generates the transition matrix P and the initial state matrix $S0$. Each row i in the transition matrix P represents the occurrence of sampler i in the current system state. Each column j in the transition matrix P represents the occurrence of sampler j in the next system state. The initial state matrix $S0$ is determined by equation 5.

IV. Experiments

A. Experimental Setup

The input to the predictive pursuit algorithm is robot locations at each timestamp from ANC or adaptive PRM and individual connection methods or samplers run on the same environment. We run ANC or adaptive PRM and all individual connection methods or samplers separately on C++ motion planning library. These algorithms are executed on a Dell Optiplex 7040 desktop machine running OpenSUSE operating system. Each robot location obtained from the ANC is considered as an $observedNode$ in the Pursuit Algorithm. Euclidean distance calculated between the $observedNode$ and the robot location obtained from each individual connector or sampler is stored in the $dist$ matrix, which is later used in the $constructProbability$ method. We have implemented the predictive pursuit algorithm in Python 2.7.14. We have performed tests on the following environments:

- **KukaYouBot**: In this environment, an 8 DOF robot has been placed in an environment containing four rooms as shown in Figure 2. We chose this environment because it has a ground robot (UGV), which is ideal for a research on air-to-ground surveillance.
- **Serial Walls**: In this environment, a cylindrical has been placed in a room containing serial walls with windows on each one of them which is the only point of exit as shown in Figure 3. We chose this environment because it has visual occlusions from the invader's (UGV) perspective and tracking the UGV will be a challenging task in this environment.
- **Zigzag**: In this environment, a 3D Hexagonal robot is placed with a start location in a free space and goal location in another free space. In order to reach between the two free spaces, the robot has to travel through a narrow passage that has a zigzag shape as shown in Figure 4. This is a 2D environment. We chose this environment because, considering the perspective of a UAV, the UGV will seem to be roaming in a 2D environment which will give us a workspace specification of the performance of our method.
- **3D House**: In this environment, a table-shaped robot is placed in a 3D house with start location in a free space in one room and goal location in another room. In order to reach between the start and goal locations, the robot has to travel through two narrow doors as shown in Figure 7.
- **2D Maze**: In this environment, the robot has to travel through a maze-like structure to reach from start to destination. The start and destinations are in two different free spaces separated by this maze in Figure 5. This is another 2D environment and it has been chosen for the same reason as that of the Zigzag environment.
- **Non-Convex S**: In this environment that is similar to Zigzag, a cuboid robot is placed with a start location in a free space and goal location in another free space. In order to reach between the two free spaces, the robot has to travel through a narrow passage that has a curve shape that looks like an inverted 'S' (hence the name) as shown in Figure 6. This is also a 2D environment and it has been chosen for the same reason as that of the 2D Maze and Zigzag environments.

It is important to note here that in our experiments, not all the invaders are of the shape of a UGV. This is because, we are concerned about the location of the UGV, and not its shape or orientation.

B. Experimental Results

The results of running the Predictive Pursuit algorithm on various heterogeneous environments is given in Table II. The first column shows the environment used from the experimental section. Second columns lists the connection method. Third column lists the sampling strategies. In case the Adaptive Method is trying different samplers, we have used the Hybrid PRM method along with the K-Closest neighbor finder. If the Adaptive Method is using ANC, we have used one particular sampler as listed in Table 1.

The percentage accuracy of the prediction is inversely proportional to the size of the environment. So we have divided the Euclidean distance between the observed position

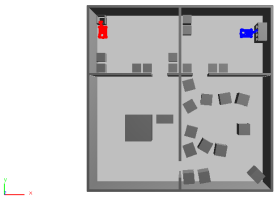

Fig. 2: **Kuka-You-Bot**

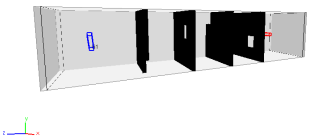

Fig. 3: **Serial Walls**

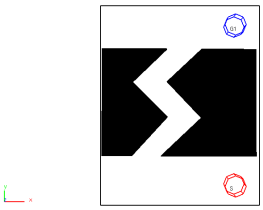

Fig. 4: **Zigzag**

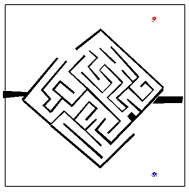

Fig. 5: **2D Maze**

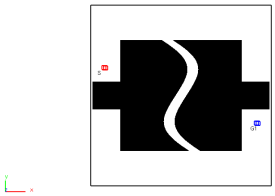

Fig. 6: **Non-Convex S**

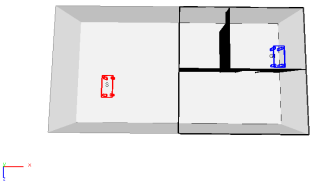

Fig. 7: **3D House**

Fig. 8: Heterogeneous Environments Studied

TABLE II: **Predictive Pursuit Run On Different Environments**

Env	Connector	SS	OP	PP	PA
KukaYouBot	ANC [7]	OBPRM	(-18.12, 27.06, -0.54)	(-20.17, 27.56, 0.62)	98.94
KukaYouBot	ANC [7]	URF	(-19.11, 30.21, -0.96)	(-18.68, 30.32, 0.35)	97.52
SerialWalls	ANC [7]	URF	(1.87, 1.70, 0.76)	(2.05, 1.65, 1.78)	99.32
ZigZag	K-Closest	H-PRM [8]	(-1.04, 0.15)	(-0.86, -0.02)	99.03
2DMaze	K-Closest	H-PRM [8]	(0.76, 11.36)	(0.75, 11.41)	99.88
NonConvexS	K-Closest	H-PRM [8]	(-0.56, 3.71)	(-0.82, 3.87)	98.21
3D House	K-Closest	H-PRM [8]	(5.03, 2.91, 6.69)	(4.50, 2.48, 7.98)	93.58

Env Environment used.
SS Sampling Strategy used by the invader (UGV).
OP Observed position of the invader (UGV).
PP Predicted position of the invader (UGV).
PA Percentage Accuracy of the predictive pursuit.
URF Uniform Random Free sampler.
H-PRM HybridPRM sampler.

and predicted position by the length of the diagonal of the environment. If it is a 2D environment, we calculate the diagonal as follows:

$$diagonal = \sqrt{(length)^2 + (breadth)^2} \tag{10}$$

If it is a 3D environment, we calculate the diagonal as follows:

$$diagonal = \sqrt{(length)^2 + (breadth)^2 + (height)^2} \tag{11}$$

Then we calculate the percentage accuracy as follows:

$$PA = (1 - (\delta(OP - PP)/diagonal)) * 100 \tag{12}$$

where PA is the percentage accuracy and $\delta(OP - PP)$ is the euclidean distance between the observed position (OP) and the predicted position (PP).
Following are the results we got for each environment:

- KukaYouBot: We ran ANC with OBPRM and Uniform Random Free samplers in two different experiments. For ANC with OBPRM, our predictive pursuit gave 98.94% accuracy. For ANC with Uniform Random Free, our predictive pursuit gave 97.52% accuracy. Both these percentages are really high considering the scaled size and complexity of the environment.
- Serial Walls: We ran ANC with Uniform Random Free sampler on this environment. Again we have a very high accuracy in prediction.
- Zigzag: We ran Hybrid-PRM with K-Closest neighbor finder on this environment and get a pretty high accuracy (99.03%).
- 2D Maze: Another environment where we ran Hybrid-PRM with K-Closest neighbor finder. Our predictive

pursuit algorithm gives a pretty high accuracy (99.88%).

- Non-Convex S: In another 2D environment which is quite similar to the Zigzag environment, we ran Hybrid-PRM with K-Closest neighbor finder and get a high accuracy (98.21%).
- 3D House: We ran Hybrid-PRM with K-Closest neighbor finder on a 3D environment and our predictive pursuit gave us an accuracy of 93.58%.

As we can see, the predictive pursuit algorithm gives better results when the invader is in a 2D environment than when it is in a 3D environment. This is because, in a 2D environment, the invader has only 2 degrees of freedom with just x-axis and y-axis coordinates to predict. But, in a 3D environment, the invader has higher (generally 6) degrees of freedom with at least x-axis, y-axis and z-axis to predict. In some cases of a 3D environment, the invader may also need to fly from one position to another because of obstacles on the ground. Therefore, it becomes more difficult to predict the location of a non-planar object. Ideally, the predictive pursuit will be run by a UAV and it will always have a 2D (bird's eye) view of the environment that the invader is in.

It is to be noted here that the choice of connectors and sampling strategies used by the invader (UGV) is arbitrary. We have chosen ANC and Hybrid PRM as an alternative to randomizing the connectors and sampling strategies, which are integral parts of motion planning for an autonomous vehicle like the invader described in our paper. The predictive pursuit algorithm can be configured to be applied on any random motion planning algorithm without any prior knowledge of what algorithm the invader is using. The novelty of our approach limited the methods we could directly compare with.

We have assumed that the pursuer is surveying the same environment as a routine task and it has already run all the possible motion planning algorithms that the invader may use. So every time it detects an invasion, it doesn't have to run the motion planning algorithms on the environment in order to predict the one used by the invader. We consider this as a one time pre-processing overhead. Our algorithm is adaptive and can be applied to real-world pursuit-evasion scenarios or on cooperative air-to-ground robots.

V. Conclusion

In this paper, we have discussed a new approach to Pursuit-Evasion in an air-to-ground surveillance scenario. This algorithm brings scalability to surveillance drones, as it is low maintenance and independent of hardware changes. In our experimental section, we have assumed that the UGV is running an Adaptive PRM or Adaptive Neighbor Connector. However, without any modification, this predictive pursuit algorithm can be further used to predict any unknown motion planning strategy used by the invader. In a future work, we can extend this algorithm to be applied to a team of pursuers and a team of invaders. This algorithm can also be applied on a cooperative team of robots to determine their relative positions in the environment. It can also be applied for collision detection in a dynamic environment.

References

[1] Shutterstock website. https://www.shutterstock.com/image-vector/drone-icon-vector-783816757

[2] Amato, N.M., Bayazit, O.B., Dale, L.K.: Obprm: An obstacle-based prm for 3d workspaces (1998)

[3] Bai, H., Hsu, D., Lee, W.S.: Integrated perception and planning in the continuous space: A pomdp approach. The International Journal of Robotics Research 33(9), 1288–1302 (2014)

[4] Boor, V., Overmars, M.H., Van Der Stappen, A.F.: The gaussian sampling strategy for probabilistic roadmap planners. In: Robotics and automation, 1999. proceedings. 1999 ieee international conference on. vol. 2, pp. 1018–1023. IEEE (1999)

[5] Bugár, M., Stanak, V., Ferencey, V.: Hybrid powertrain conceptual design for unmanned ground vehicle. Science & Military Journal 6(1), 13 (2011)

[6] Chen, H., Chen, J., Zhang, W., Liu, H.: Analysis of a new pursuit-evasion game based on game theory. In: Natural Computation (ICNC), 2015 11th International Conference on. pp. 875–880. IEEE (2015)

[7] Ekenna, C., Jacobs, S.A., Thomas, S.L., Amato, N.M.: Adaptive neighbor connection for prms: A natural fit for heterogeneous environments and parallelism. In: IROS. pp. 1249–1256 (2013)

[8] Ekenna, C., Uwacu, D., Thomas, S., Amato, N.M.: Improved roadmap connection via local learning for sampling based planners. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. pp. 3227–3234. IEEE (2015)

[9] Elbanhawi, M., Simic, M.: Sampling-based robot motion planning: A review. Ieee access 2, 56–77 (2014)

[10] Grewal, M.S.: Kalman filtering. In: International Encyclopedia of Statistical Science, pp. 705–708. Springer (2011)

[11] Hui, C., Yousheng, C., Xiaokun, L., Shing, W.W.: Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision. In: Control Conference (CCC), 2013 32nd Chinese. pp. 5895–5901. IEEE (2013)

[12] Kavraki, L.E., Kolountzakis, M.N., Latombe, J.C.: Analysis of probabilistic roadmaps for path planning. In: Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on. vol. 4, pp. 3020–3025. IEEE (1996)

[13] Ko, T.: A survey on behavior analysis in video surveillance for homeland security applications (2008)

[14] Lee, D.J., Zhan, P., Thomas, A., Schoenberger, R.B.: Shape-based human detection for threat assessment. In: Visual Information Processing XIII. vol. 5438, pp. 81–92. International Society for Optics and Photonics (2004)

[15] Müller, J., Sukhatme, G.S.: Risk-aware trajectory generation with application to safe quadrotor landing. In: Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. pp. 3642–3648. IEEE (2014)

[16] Nash, J.F., et al.: Equilibrium points in n-person games. Proceedings of the national academy of sciences 36(1), 48–49 (1950)

[17] Oliveira, T., Encarnaçao, P.: Ground target tracking for unmanned aerial vehicles. In: AIAA Guidance, Navigation, and Control Conference. p. 8082 (2010)

[18] Owen, M., Yu, H., McLain, T., Beard, R.: Moving ground target tracking in urban terrain using air/ground vehicles. In: GLOBECOM Workshops (GC Wkshps), 2010 IEEE. pp. 1816–1820. IEEE (2010)

[19] Rigling, B.D.: Adaptive filtering for air-to-ground surveillance. In: Algorithms for Synthetic Aperture Radar Imagery XI. vol. 5427, pp. 53–62. International Society for Optics and Photonics (2004)

[20] Upadhyay, A., Ekenna, C.: Investigating heterogeneous planning spaces. In: Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2018 IEEE International Conference on. pp. 108–115. IEEE (2018)

[21] Van Den Berg, J., Patil, S., Alterovitz, R.: Motion planning under uncertainty using iterative local optimization in belief space. The International Journal of Robotics Research 31(11), 1263–1278 (2012)

[22] Wilmarth, S.A., Amato, N.M., Stiller, P.F.: Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. In: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. vol. 2, pp. 1024–1031. IEEE (1999)

[23] Yu, H., Meier, K., Argyle, M., Beard, R.W.: Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles. IEEE/ASME Transactions on Mechatronics 20(2), 541–552 (2015)

[24] Zhou, J., Hoang, J.: Real time robust human detection and tracking system. In: null. p. 149. IEEE (2005)