2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Uncertainty Measured Markov Decision Process in Dynamic Environments

Sourav Dutta[1], Banafsheh Rekabdar[2], and Chinwe Ekenna[1]

Abstract

Successful robot path planning is challenging in the presence of visual occlusions and moving targets. Classical methods to solve this problem have used visioning and perception algorithms in addition to partially observable markov decision processes to aid in path planning for pursuit-evasion and robot tracking.

We present a predictive path planning process that measures and utilizes the uncertainty present during robot motion planning. We develop a variant of subjective logic in combination with the Markov decision process (MDP) and provide a measure for belief, disbelief, and uncertainty in relation to feasible trajectories being generated. We then model the MDP to identify the best path planning method from a list of possible choices. Our results show a high percentage accuracy based on the closest acquired proximity between a target and a tracking robot and a simplified pursuer trajectory in comparison with related work.

I. Introduction

Planning paths for robots in narrow and uncertain spaces e.g, boulders falling causing blocked areas is still a difficult task. In all types of environments that a robot may encounter (if they're scalable), they are confronted with various kinds of decisions involving multiple choices and relative uncertainty. A clear understanding of these uncertainties becomes a prerequisite for effective decision making.

Recently an innovative logic variant method was developed [5], called Collective Subjective Logic (CSL), and the authors created models for minimizing uncertainty in a decision making process by using Probabilistic Soft Logic (PSL). They combined multiple opinions concurrently and provided high scalability and prediction accuracy while dealing with uncertain opinions. This work will take insight from the aforementioned work, make innovative advancements and apply them to path planning problems.

Interestingly, some important work in robotics that looks into belief states and partially observable processes have been developed in [4], [10], [17]. The models, however, see an exponential growth in their running time due to the increasing number of states needed as more unknown locations are explored. Improved variants of these models have been developed but, the measurement of uncertainty present and how it can mitigate longer planning times has not been investigated.

This research is supported in part by NSF awards CRII-IIS-1850319

[1]Sourav Dutta and Chinwe Ekenna - Department of Computer Science, University at Albany, SUNY, NY 12206, USA {sdutta2, cekenna}@albany.edu.

[2]Banafsheh Rekabdar - Department of Computer Science Southern Illinois University, IL 62901, U.S.A. banafsheh.rekabdar@siu.edu.

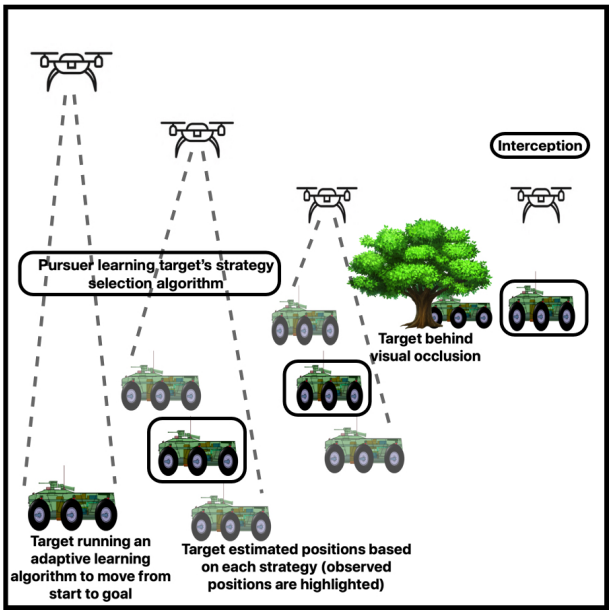


Fig. 1: **Process Overview**

In this paper, we introduce a novel combination of the CSL and MDP which we call Uncertainty- MDP (U-MDP). Our method has two main modules as illustrated in Figure 1 : the learning phase; where the algorithm uses a combination of PSL and Subjective Logic (SL) to learn the uncertainty representations of the available motion planning algorithms and the prediction phase; where the MDP uses the evidence gathered in the learning phase to predict the planning algorithm in use.

Compared to other MDP variants, the states in our stochastic process don't increase exponentially instead we utilise available planning strategies to induce on the states of the MDP. We make measurements of the calculated uncertainty in addition to belief and disbelief, to pick the best choice of planning strategy in a given environment. Using our approach thus limits the search space and reduces computational complexity.

The contributions of this paper include:

- an improved target tracking algorithm that takes observational and estimation uncertainties into account and uses them in the decision making process and
- a faster algorithm with reduced execution time with a reduction of an exponentially increasing belief space to a fixed number dependent on the number of planning strategies available.

II. Related Work

In this section, we discuss work related to logic systems, Markov decision process, and application to robotics.

A. Collective Subjective Logic

Subjective Logic (SL) is a variant of probabilistic logic with a primary focus on the existence of uncertainty or incomplete knowledge. SL offers a variety of operators like belief, disbelief, and uncertainty to update opinions. However, SL operators don't collectively deal with multiple opinions concurrently, rather, they sequentially combine two opinions. As a result, SL operators lack scalability to derive opinions from a large-scale network data. In an SL, an opinion is represented as in equation 1, where b is the belief factor (e.g. true), d is the disbelief factor (e.g false), a is a pre-known base rate inferred from domain knowledge, and u is the amount of uncertainty. Decisions are made by equations 2 and 3, where E_b is the expected value for belief probability, E_d is the expected value for disbelief probability.

$$w = (b, d, u, a) \tag{1}$$

$$E_b = b + a * u \tag{2}$$

$$E_d = d + (1 - a) * u \tag{3}$$

Probabilistic Soft Logic (PSL) [11], on the other hand, is a machine learning framework that develops probabilistic models using a concise logical syntax, and, solving them via fast convex optimization. PSL provides a formulation tool to determine unknown probabilities and requires probabilities to be point-valued. PSL provides collective reasoning with high scalability based on relationships between opinions but does not deal with uncertainty. PSL assigns a rule weight for each rule to indicate the level of confidence. PSL uses truth probabilities in a range of $[0, 1]$, instead of a binary decision. A given probability p_{x_i} is called an atom, where x_i is a random variable representing a certain relationship, details of which will be discussed in Section III-A.

$$p_{x_1} \wedge p_{x_2} = max\{0, p_{x_1} + p_{x_2} - 1\} \tag{4}$$

$$p_{x_1} \vee p_{x_2} = min\{p_{x_1} + p_{x_2}, 1\} \tag{5}$$

$$\neg p_{x_1} = 1 - p_{x_1} \tag{6}$$

By taking the merits of both SL and PSL, a hybrid probabilistic logic algorithm, called Collective Subjective Logic (CSL) was developed in [5], which provides high scalability and high prediction accuracy while dealing with uncertain opinions over a large-scale network dataset. The basic entities in CSL are subjective opinions as in SL while the structural relations between the variables are modeled using PSL rules. This helps in supporting the collective inference of unknown variables in large scale network data. Our new CSL variant allows the application of SL-like simplistic rules on probabilistic data without the use of an expensive bi-conditional between SL and PSL.
We introduce a variant to CSL that uses the basic rule for constructing principles of PSL with uncertainty properties included via the logic system of SL. On applying this on a set of states, we derive E_b and E_d as vectors instead of scalar values obtained from equations (2) and (3). These vectors are generated for each discreet timestamp, and each element of the vector represents the expected value of a state (strategy is our case) for that timestamp. This is quite similar to the CSL but derives its notations from SL.

B. Markov Decision Process

A Markov chain is a stochastic model of events, where a sequence of independent identically distributed (iid) random variables $\{X_n\}_{n=1}^{\infty}$ is ordered by time. MDP is a stochastic decision making process that satisfies the Markov property which states that the probability of moving to the next state depends only on the present state and is independent of the previous states [2].

In this paper we introduce a new MDP variant that doesn't take exponential time for self-learning using a reward-based approach. In our MDP variant, no static initial state or transition probabilities are used, instead, we derive them from the knowledge gathered from our CSL variant previously highlighted in this paper.

C. Motion Planning under uncertainties using Markov Decision Processes

Uncertainty in a robot's belief space is a general problem in both motion planning and target tracking. In recent research, the MDP model has been extended and updated to account for uncertainty in dynamic environments [6], [19], [22]. A tool belt of sensors gathers data on external variables such as obstacles, weather patterns and in certain environments; autonomous and independent entities [13], [15], [16], [26]. This provides the robot with a detailed description of the world. Sensor inaccuracies and noisy data, however, provide the robot with a fragmented world. To operate and navigate an environment under these conditions is uncertain. In recent years, several new variants of the MDP model have surfaced. SAFE-MDP [12], Mixed Observability-MDP [6], and CC-MDP [12]. However, POMDP tends to be at the root of these expansionist models. For example, the POMDP model in [21] exhibits the ability to model and reduce uncertainty for a common problem in aerospace, the presence of strong winds. Strong winds account for uncertain scenarios during motion planning and localization in the UAV's belief space and it's the ability to track a target.

In [10], the authors developed a target tracking algorithm that follows some basic principles of partially observable Markov decision process (POMDP) but reduces the computational complexity of POMDP. This method called SARSOP (Successive Approximations of the Reachable Space under Optimal Policies) uses a combination of target following and target searching by modeling target tracking as a POMDP.

In [20], the authors developed another UAV-based target tracking algorithm by modeling the problem as a POMDP and using a modular framework that runs on the Robotic Operating System (ROS). It computes a policy for executing actions instead of waypoints to navigate and avoid obstacles.

D. Motion Planning Primitives and SBMP

In this section, we discuss the different sampling strategies that a UGV may access and exploit. Sampling based planning is a class of motion planning algorithms that can be broken

down into two phases. The learning phase; where the algorithm applies a local planner to construct a graph of collision free nodes (sampling) within any given environment. In the query phase; the collision free nodes are connected within the graph (neighbor connection) to create a path from start to goal using an adaptively selected strategy [7]. Many different algorithms exist for sampling and neighbor connecting, a combination of which we refer to as a "strategy" throughout this paper.

Probabilistic Roadmap Algorithm (PRM) [14] is a sampling based motion planning technique that sample robot configurations (nodes) and connect them to form a graph (roadmap) containing feasible trajectories. It has been used to solve a number of planning strategy problems, where choosing an appropriate strategy (sampler, neighbor connector, etc) is dependent on the given problem environment. In its initial variants, the strategy was determined a priori without any knowledge of the environment. However, planning in heterogeneous environments necessitates dividing the problem into regions where these choices have to be made for each one. Simple hand-selection of the best strategy for each region becomes infeasible. Heterogeneous Planning Spaces (HPS) [18] a method developed recently that takes as input a list of sampling methods and runs them iteratively to determine a sampler for each of the regions in the environment. The regions are partitioned using a visibility-based cost function, and the sampler which suits a particular region is also determined by the visibility factor. Hybrid PRM [9] is another learning variant of sampling based techniques and, strategies are selected from a list of candidates (samplers and neighbor connectors) using an adaptive learning framework. Keeping the neighbor connector fixed, In this work we used these following sampling based strategies as candidate strategies in the HybridPRM:

- **Obstacle-Based PRM (OBPRM)** [1]: In OBPRM, the nodes are sampled on or near obstacle surfaces, which in turn, improves the quality of the graph for environments that are cluttered.
- **Medial Axis PRM (MAPRM)** [24]: In MAPRM, 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** [3]: The Gaussian PRM 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.

III. Methodology

Uncertainty CSL (U-CSL) and MDP (U-MDP)

We present 2 algorithms U-CSL and U-MDP that introduces the ability to measure the uncertainty present in the environment and exploit it to produce feasible trajectories. These algorithms can be generalized to work both during the sampling, connection and querying stage of sampling based planning methods. We make innovations during the sampling stage and produce results and comparisons with existing work.

Algorithm 1 U-CSL

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

1: **Initialize:** $threshold \leftarrow 0.8$
2: **Initialize:** $observedNode \leftarrow startNode$
3: **Initialize:** $NS \leftarrow size(Strategies)$
4: **while** all values in E_b or E_d is less than $threshold$ **do**
5: **for** $i = 1$ to NS **do**
6: $samples[i] \leftarrow$ generate nodes using $Strategies[i]$, with start node as $observedNode$ and goal node as $goalNode$.
7: $r[i] \leftarrow$ number of valid samples in $samples$ after collision check.
8: $s[i] \leftarrow$ number of invalid samples in $samples$ after collision check.
9: $observedNode \leftarrow followUGV()$
10: **if** target can not be observed due to low visibility **then**
11: $observedNode \leftarrow$ center of convex hull created by all samples from NS Strategies.
12: **for** $i = 1$ to NS **do**
13: $w[i] \leftarrow$ average of distances between all valid samples generated by $Strategies[i]$ and stored in $r[i]$, and the $observedNode$.
14: Normalize $r[i], s[i]$ and $w[i]$ to get $b[i], d[i]$ and $u[i]$
15: Calculate $E_b[i]$ and $E_d[i]$ from Equations (7) and (8), respectively.
16: **if** any value in E_b is greater than $threshold$ **then**
17: $idx \leftarrow$ index of $max(E_b)$
18: **else**
19: $idx \leftarrow$ index of $max(E_d)$
20: Add $Strategies[idx]$ to $StrategySequence$
21: $StrategyOccurrance[idx] \leftarrow StrategyOccurrance[idx] + 1$
22: **return** [$StrategySequence$,$StrategyOccurrance$]

A. U-CSL

In our U-CSL model we consider strategies as entities, at each timestamp, each strategy is assigned positive evidence r, negative evidence s, and an amount of uncertainty w. These parameters are all vectors as they represent multiple strategies. They are then normalized into b, d and u respectively from r, s and w.

At each timestamp, an attempt to visually track the target robot is made and a new observation is recorded. The estimated location of this robot is stored in $obsevedNode$ (Algorithm (1), step 9). r and s are calculated for each strategy by the number of valid and invalid samples, respectively, generated by that strategy after collision check for that timestamp. In case of a missing observation, where the target robot can not be visually observed due to obstacles in the environment, the location is calculated from the center of the convex hull created by valid samples for all the strategies. w is calculated for each strategy by using the average of euclidean distances between the valid samples for that strategy and the location. For each timestamp, the strategy with the highest expected value in E_b or E_d is stored in $StrategySequence$, and the $StrategyOccurance$

for that strategy is incremented by 1. This process is repeated until we find E_b or E_d for one strategy that gives us value over a threshold. At that point, the CSL algorithm is stopped, and the $StrategySequence$ and the $StrategyOccurance$ are forwarded to the prediction model MDP. The list of strategies used in this paper was discussed in section II-D of our related work. Equations (7) and (8) are calculation for E_b and E_d in U-CSL, where a gives the base rate. Initially, we assign equal base rates to all the strategies, assuming uniform distribution, to give a fair chance to all probable strategies during the learning phase.

$$E_b[i] \leftarrow b[i] + a[i] * u[i] \tag{7}$$

$$E_d[i] \leftarrow d[i] + (1 - a[i]) * u[i] \tag{8}$$

B. U-MDP

The U-MDP model, Algorithm (2), predicts the estimated location of the target. Generally, MDP's are characterized by a base state S_0 and a transition probability matrix P. These two parameters are derived from domain knowledge or by previous observations. In U-MDP, we use the learning phase to determine the base state S_0 and a transition probability matrix P dynamically, as opposed to static state transitions used by [7], [25] works. After receiving the $StrategySequence$ and the $StrategyOccurance$ from the CSL, the MDP constructs the transition probability matrix by using the $constructProbability$ algorithm implemented in our current work [7]. Once the transition matrix has been created, Algorithm (2) starts calculating the expected probabilities of the strategies, and records which strategy tends to 1 first in a number of recorded timestamps. We do this via implementing equation 9.

$$S_i = S_0 * P^i, i \in \{1, 2, ..., n\} \tag{9}$$

Once a strategy has been predicted, and the number of timestamps required has been determined, U-MDP uses this predictive strategy to predict the location of the target in conjunction with a local planner and builds a predicted path for the target robot.

One of the contributions of our algorithm is to reduce the computational cost of using any MDP. Generally in all MDPs, the belief space grows exponentially as the algorithm starts exploring unknown spaces. In POMDP or regular MDP, the number of iterations required to reach probability 1 is uncertain [4]. Given that, our goal is to minimize the number of iterations U-CSL takes to learn and U-MDP takes to predict. This can only be done if one of the E_b or E_d attains probability 1 with least possible number of iterations.
Since b, d and w are normalized factors, we can write equation (10).

$$b[i] + d[i] + u[i] = 1 \tag{10}$$

From equations (8) and (10), we derive equation (11).

$$E_b[i] = 1 - (d[i] + u[i]) + a[i] * u[i] \tag{11}$$

$d[i]$ and $u[i]$ range between [0,1], and since $a[i]$ is initialized to be $1/NS$, where NS is the number of strategies and remains constant, we can safely assume that E_b tends to 1 as the uncertainty u decreases. The value of uncertainty w depends on the average of some euclidean distances and as we explore more regions of the environment, eventually, it will tend to zero as the samples generated get merged to the goal location. Therefore, the ability to make a decision under belief or disbelief tends to 1 for a particular strategy and the U-CSL is guaranteed to converge.

Algorithm 2 U-MDP

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

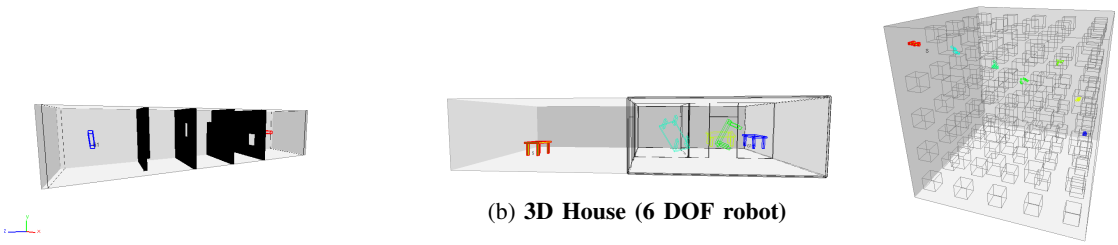
$[StrategySequence, StrategyOccurrance] \leftarrow GetStrategySequence()$
$[S_0, P] \leftarrow constructProbability()$
while $countIter \leq maxObservations$ **do**
  $S_{next} \leftarrow S_0 * P^{countIter}$
  for $k = 1$ to NS **do**
    if $S_{next}(k) \equiv threshold$ **then**
      $flag \leftarrow True$
      $strategyIndex \leftarrow k$
      break
  $countIter \leftarrow countIter + 1$
  if $flag \equiv True$ **then**
    break
if $flag \equiv True$ **then**
  for $l = 1$ to $countIter$ **do**
    $seeds \leftarrow LocalPlanner(Strategies(strategyIndex), observedNode)$

Output. $seeds$

IV. Experiments and Results

A. Experimental Setup

The U-MDP algorithm was trained using HPS. During the training phase of our algorithm (U-CSL), we used one strategy at a time along with the HPS framework.
Different experiments was performed using both HPS and HybridPRM [8]. The HybridPRM algorithm works in a similar way as the HPS, except that, its cost function is evaluated for the entire environment, and once a node generator has been determined as a suitable one for the environment, it is used throughout the environment. This creates a level of uncertainty as to which strategy is suited for any given environment. Another uncertainty factor comes from the fact that the two paths created by a HybridPRM or HPS strategy with different lists of local planners, samplers and node connectors, will never be the same. There will always be some difference between the euclidean distances between the observed coordinates connecting each path at each timestamp. We utilize these uncertainties to train our U-CSL Algorithm.
Both the training and testing algorithms were executed on a Dell Optiplex 7040 desktop machine running OpenSUSE operating system. We have performed tests on the following environments - Serial Walls (Fig. 2a), 3D House (Fig. 2b), Cluttered (Fig. 2c), and KukaYouBot (Fig. 3a).



(a) **Serial Walls (6 DOF robot)**

(b) **3D House (6 DOF robot)**

(c) **Cluttered environment (6-9 DOF robot)**

Fig. 2: Heterogeneous Environments Studied

TABLE I: **U-CSL and U-MDP Run On Different Environments**

Env	PS	IS	OP	PP	DIP	PA	Dist_Avg	IT	PT
Kuka-You-Bot	HPS	HPS	(-17.07,21.68,0.67)	(-18.2,20.37,0.5)	1.74	99.23	4.68	49	9
Kuka-You-Bot	HPS	HybridPRM	(-20.6,20.13,0.59)	(-18.2,20.37,0.5)	2.41	98.94	3.75	49	9
Serial Walls	HPS	HPS	(0.65,3.08,2.67)	(0.7,3.13,2.39)	0.29	98.54	1.86	26	7
Serial Walls	HPS	H-PRM	(3.12,0.64,9.54)	(2.96,0.65,9.36)	0.24	98.78	1.01	28	26
House	HPS	HPS	(6.18,1.58,4.66)	(5.73,2.03,4.39)	0.69	96.96	2.65	14	12
House	HPS	H-PRM	(6.35,1.62,4.66)	(6.35,1.5,4.52)	0.19	99.19	1.4	8	5
Cluttered Cube	HPS	HPS	(3.61,2.27,4.36)	(4.18,2.34,4.43)	0.58	96.66	4.11	13	4
Cluttered Cube	HPS	H-PRM	(2.1,2.54,3.75)	(2.75,2.01,4.36)	1.03	94.01	4.42	17	1
Cluttered 7DOF	HPS	HPS	(1.55, 1.73, 6.55)	(1.33, 1.47, 6.02)	0.63	96.36	2.28	12	3
Cluttered 7DOF	HPS	H-PRM	(6.35, 1.62, 4.66)	(6.35, 1.49 , 4.52)	0.19	98.9	1.40	8	5
Cluttered 9DOF	HPS	HPS	(5.89,7.93,8.75)	(8.13,4.49,7.83)	4.21	75.71	5.77	6	1
Cluttered 9DOF	HPS	H-PRM	(6.05,7.91,7.63)	(3.77,6.25,7.9)	2.83	83.64	5.51	7	1

[Env] Environment used.
[PS] Sampling Strategy used by the pursuer.
[IS] Sampling Strategy used by the invader.
[OP] Observed position of the invader at the point of interception.
[PP] Predicted position of the invader at the point of interception.
[DIP] Distance between invader and pursuer at point of interception.
[PA] Percentage Accuracy of the U-MDP.
[Dist_Avg] Average distance between the paths of the invader and the pursuer.
[IT] Number of timestamps taken by Invader to reach from start to goal.
[PT] Number of timestamps taken by Pursuer to intercept the invader.
[HPS] Heterogeneous Planning Spaces strategy.
[H-PRM] HybridPRM strategy.

B. Results

Table (I) shows the results of running the U-MDP algorithm on different heterogeneous environments. We compared the paths of the target and the pursuer to determine an average distance between the two (column Dist_Avg). We also determined a point of interception (columns OP and PP) which shows the closest position of the target and the pursuer in their individual paths. Based on the positions of the target and the pursuer, we calculate a percentage accuracy (column PA) that demonstrates the distance between the target and the pursuer with respect to the size of the environment. Column IT gives the number of timestamps taken by the target to reach its goal, while the column PT gives the number of timestamps taken by the pursuer to intercept the target.
The percentage accuracy was calculated in Equation (12), where $diagonal$ is the diagonal of the bounding box of the environment, PI is the position of the invader, and PP is the position of the pursuer.

$$\%Accuracy = (1 - (\delta(PI - PP)/diagonal)) * 100 \quad (12)$$

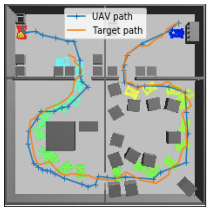
Based on the parameters explained above, we have the following observations for each of the environments:

- **Serial Walls:** We ran 2 differennt experiments with HPS and HybridPRM got a percentage accuracy of more than 98% in both cases. The average distance recorded between the path of the invader and the pursuer is small(1.86 and 1.01) compared to the size of the environment (of dimensions $4 \times 4 \times 19$) and its complexity. The number of timestamps the pursuer takes while the invader uses the HybridPRM (26) is significantly higher than when it uses the HPS (7). This is because of the difference in the cost functions used in both the algorithms. But in both cases, the pursuer is able to intercept the invader before it reaches the goal.

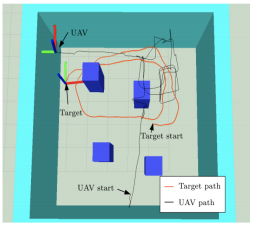
TABLE II: **U-CSL and U-MDP Run On KukaYouBot Environment with different seeds**

Invader trajectory time from start to goal (milliseconds)	Time to intercept (milliseconds)	Success(S)/Failure(F)
1322.430	723.677	S
91.644	746.532	F
4162.000	737.314	S
8213.810	674.919	S
34937.600	664.714	S
1863.090	751.444	S
629.816	726.422	F
9533.800	703.242	S
98517.799	554.380	S
146814.000	531.326	S

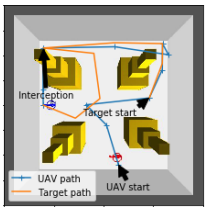
- **3D House:** We ran HPS and HybridPRM in two different experiments for this environment and got a percentage accuracy of more than 96% and 99%, respectively. The average distance between the path of the invader and the pursuer is also pretty low (2.65



(a) **Path comparison between invader and pursuer in KukaYouBot environment using U-CSL and U-MDP [20]**

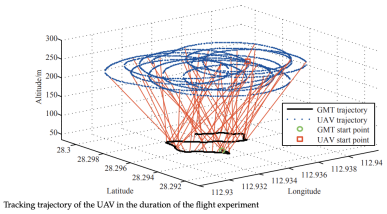


(b) **Path comparison between invader and pursuer using POMDP [20]**

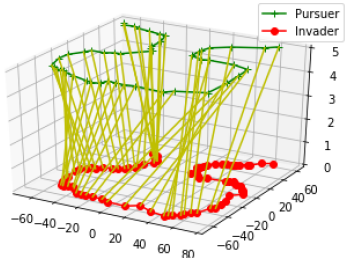


(c) **Path comparison between invader and pursuer using U-CSL and U-MDP in similar environment**

Fig. 3: Path comparison between invader and pursuer



(a) **Path comparison between invader and pursuer using adaptive filter in vision-based tracking system [23]**



(b) **Path comparison between invader and pursuer using U-CSL and U-MDP in similar environment**

Fig. 4: Visual tracking vs U-CSL and U-MDP run on 3D Environments

and 1.4) as compared to the size of the environment (of dimensions $20 \times 4 \times 10$) and its complexity. The number of timestamps the pursuer takes while the invader is using the HPS (12) is higher than when it uses the HybridPRM (5), for this environment.

- **Cluttered:** We ran HPS and HybridPRM in six different experiments for this environment with different degrees of freedom of the robot. From the results table, we can see that with higher DOFs, the percentage accuracy gets low (96.36%-75.71%), but the time to intercept decreases (5-1 timestamps). The average distance between the path of the invader and the pursuer is also low (4.11, 4.42, 2.28, 1.40, 5.77 and 5.51) as compared to the size of the environment (of dimensions $10 \times 10 \times 10$) and its complexity.

- **KukaYouBot:** We ran HPS and HybridPRM in two different experiments for this environment and got a percentage accuracy of 98% and 99%, respectively. The average distance between the path of the invader and the pursuer is small (4.68 and 3.75) as compared to the size of the environment (of dimensions $160 \times 160 \times 17.6$) and its complexity. The number of timestamps the pursuer takes while the invader is using the HybridPRM is the same as when it uses the HPS (9). In both cases, the pursuer is able to intercept the invader before it reaches the goal. On a different set of experiments, we ran U-CSL and U-MDP on the KukaYouBot environment with 10 different randomized seeds and recorded the time taken by the invader from start to goal (without being intercepted) and the time taken by the pursuer to intercept the invader. The results from the experiments have been recorded in Table II and the trajectory comparison from one single seed is shown in Figure 3a. The experiment was considered to be a success if the pursuer intercepts the invader in a lesser amount of time than that taken by the invader to reach from start to goal. We record a success rate of 80%.

C. Comparison with Related Work

We also compared our algorithm with other competitive algorithms from existing literature [20]. The path comparison between the invader (target/UGV) and the pursuer (UAV) using U-CSL and U-MDP is shown in Figure 3c. When compared to the trajectory comparison of a similar algorithm using POMDP (Figure 3b), we observe that the UAV trajectory generated using U-CSL and U-MDP follows the trajectory of the target more closely than the POMDP variant. When we compared it with a visual tracking-based algorithm [23] (Figure 4a), we observe that the UAV trajectory using our method gives a more simpler trajectory Figure 4b).

V. Future Work

In future work, we plan on investigating replacement for the strategies during the learning phase including considering the degree of freedom information of the robot during the learning phase. We also plan on performing real world applications to test our algorithm in robot retrieval and monitoring scenarios. As an immediate plan, we intend to apply this new learning method to adaptively select a strategy for an environment and build a path planning algorithm.

REFERENCES

[1] Nancy M Amato, O Burchan Bayazit, and Lucia K Dale. Obprm: An obstacle-based prm for 3d workspaces. 1998.

[2] Frank Bickenbach and Eckhardt Bode. Evaluating the markov property in studies of economic convergence. *International Regional Science Review*, 26(3):363–392, 2003.

[3] Valérie Boor, Mark H Overmars, and A Frank Van Der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Robotics and automation, 1999. proceedings. 1999 ieee international conference on*, volume 2, pages 1018–1023. IEEE, 1999.

[4] Krishnendu Chatterjee, Martin Chmelík, and Jessica Davies. A symbolic sat-based algorithm for almost-sure reachability with small strategies in pomdps. In *AAAI*, pages 3225–3232, 2016.

[5] Feng Chen, Chunpai Wang, and Jin-Hee Cho. Collective subjective logic: Scalable uncertainty-based opinion inference. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 7–16. IEEE, 2017.

[6] Jean-Alexis Delamer, Yoko Watanabe, and Caroline P. Carvalho Chanel. Solving path planning problems in urban environments based on a priori sensors availabilities and execution error propagation. In *AIAA Scitech 2019 Forum*, page 2202, 2019.

[7] Sourav Dutta and Chinwe Ekenna. Air-to-ground surveillance using predictive pursuit. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8234–8240. IEEE, 2019.

[8] Chinwe Ekenna, Sam Ade Jacobs, Shawna L Thomas, and Nancy M Amato. Adaptive neighbor connection for prms: A natural fit for heterogeneous environments and parallelism. In *IROS*, pages 1249–1256, 2013.

[9] Chinwe Ekenna, Diane Uwacu, Shawna Thomas, and Nancy M Amato. Improved roadmap connection via local learning for sampling based planners. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3227–3234. IEEE, 2015.

[10] David Hsu, Wee Sun Lee, and Nan Rong. A point-based pomdp planner for target tracking. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2644–2650. IEEE, 2008.

[11] Bert Huang, Angelika Kimmig, Lise Getoor, and Jennifer Golbeck. Probabilistic soft logic for trust analysis in social networks. In *International Workshop on Statistical Relational Artificial Intelligence (StaRAI 2012)*. Citeseer, 2012.

[12] Xin Huang, Ashkan Jasour, Matthew Deyo, Andreas Hofmann, and Brian C Williams. Hybrid risk-aware conditional planning with applications in autonomous vehicles. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3608–3614. IEEE, 2018.

[13] Cheng Hui, Chen Yousheng, Li Xiaokun, and Wong Wing Shing. Autonomous takeoff, tracking and landing of a uav on a moving ugv using onboard monocular vision. In *Control Conference (CCC), 2013 32nd Chinese*, pages 5895–5901. IEEE, 2013.

[14] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 4, pages 3020–3025. IEEE, 1996.

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

[16] Brian D Rigling. Adaptive filtering for air-to-ground surveillance. In *Algorithms for Synthetic Aperture Radar Imagery XI*, volume 5427, pages 53–62. International Society for Optics and Photonics, 2004.

[17] Mária Svoreňová, Martin Chmelík, Kevin Leahy, Hasan Ferit Eniser, Krishnendu Chatterjee, Ivana Černá, and Calin Belta. Temporal logic motion planning using pomdps with parity objectives: case study paper. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 233–238. ACM, 2015.

[18] Aakriti Upadhyay and Chinwe Ekenna. Investigating heterogeneous planning spaces. In *Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2018 IEEE International Conference on*, pages 108–115. IEEE, 2018.

[19] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[20] Fernando Vanegas, Duncan Campbell, Markus Eich, and Felipe Gonzalez. Uav based target finding and tracking in gps-denied and cluttered environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2307–2313. IEEE, 2016.

[21] Fernando Vanegas, Duncan Campbell, Nicholas Roy, Kevin J Gaston, and Felipe Gonzalez. Uav tracking and following a ground target under motion and localisation uncertainty. In *Aerospace Conference, 2017 IEEE*, pages 1–10. IEEE, 2017.

[22] Fernando Vanegas, Jonathan Roberts, and Felipe Gonzalez. Uav tracking of mobile target in occluded, cluttered and gps-denied environments. In *2018 IEEE Aerospace Conference*, pages 1–7. IEEE, 2018.

[23] Xun Wang, Huayong Zhu, Daibing Zhang, Dianle Zhou, and Xiangke Wang. Vision-based detection and tracking of a mobile ground target using a fixed-wing uav. *International Journal of Advanced Robotic Systems*, 11(9):156, 2014.

[24] Steven A Wilmarth, Nancy M Amato, and Peter F Stiller. 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*, volume 2, pages 1024–1031. IEEE, 1999.

[25] Leonore Winterer, Sebastian Junges, Ralf Wimmer, Nils Jansen, Ufuk Topcu, Joost-Pieter Katoen, and Bernd Becker. Motion planning under partial observability using game-based abstraction. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2201–2208. IEEE, 2017.

[26] Huili Yu, Kevin Meier, Matthew Argyle, and Randal W Beard. 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.