Uncertainty Measured Markov Decision Process

Sourav Dutta, Daniel Hug and Chinwe Ekenna

I. Abstract

In robot pursuit evasion scenarios, robot tracking or, multiple robots cooperatively performing tasks, the aim of one robot is to be able to stay in the vicinity of the other robot(s), following and interacting if needed. This becomes more difficult if the environment this occurs in is cluttered, uneven or having obstructing objects in it. One classical way to achieve these goals has been the development of visioning and perception algorithms in addition to partially observable Markov decision process to aid in path planning. In this work we present a predictive path planning process that measures and utilizes the uncertainty present in plannings spaces. We develop a variant of subjective logic in combination with Markov decision process (MDP) and provide a measure for belief, disbelief and uncertainty in relation to feasible trajectories being generated with path planning algorithms. We model the MDP to identify the best path planning method from a list based on these properties. In our experiments, we have compared the paths for both the invader and the pursuer. Our results show a high percentage accuracy based on the closest proximity of the invader and the pursuer.

II. Introduction

Planning paths for robots in narrow spaces or spaces that have uncertain situations e.g, boulders falling causing blocked areas is still a very 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. Although the topic of reasoning under uncertainty is well established and has been studied since the 1960s, many belief and reasoning models are unable to represent uncertainty in terms of its different manifestations.

The concept of belief theories has been earlier researched and applied for decision making under uncertainty [8], [9], [17]. This includes Fuzzy Logic [25], Dempster-Shafer Theory [19], and Probabilistic Soft Logic(PSL) [13] [4]. Subjective logic provides an assortment of operators to infer a new opinion based on observed motion. To mitigate ambiguous and unpredictable environments while learning the planning algorithm, an opinion is formed based on belief, disbelief, and uncertainty. Subjective logic is directly compatible with binary logic, probability calculus and classic probabilistic logic [15].

Recently an innovative logic variant method was developed [6], the authors invent models for minimizing uncertainty in a decision making process by using Probabilistic Subjective 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 that work, make innovative contributions and apply it to path planning problems.

Interestingly some important work in robotics that looks into belief states and partially observable processes have been developed e.g [5], [12], [21]. The models however have increasing number of states as more unknown locations are explored making its running time exponential. Variants to the models were developed and have shown improvements but, the notion and measurement of uncertainty and how it can mitigate longer planing times has not been investigated.

In this paper, we introduce a novel method with innovative variants of the SL and MDP which we call Uncertainty-MDP (U-MDP). Our method has two main modules: the learning phase; where the algorithm uses a combination of Probabilistic Soft Logic (PSL) and Subjective logic 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. Our algorithm as higlighted in Figure 1 is a specialization over traditional MDP, because, we innovatively use the planning strategies the MDP will learn from as states and utilizing the calculated uncertainty in addition to belief and disbelief measures determine what the best method to employ in a given scenario is. This limits the search space and reduces computational complexity. We have also eliminated the necessity of a fixed number of observations before making a prediction. The number of observations required are now dependent on the CSL algorithm.

The contributions of this research include:

- Taking observational and estimation uncertainties into account and using them in the decision making process.
- Reducing execution time by reducing belief space to a fixed number of strategies.
- Making learning and prediction processes deterministic through the combination of CSL and MDP.

From our experiments, we see that by using CSL for the learning phase, the MDP reaches its state of equilibrium much faster than using some uniform distribution for the initial state probabilities. Our results show a high percentage accuracy based on the path observed by the pursuer and the target. we also see a high percentage accuracy based on the possible point of interception of the target. Even for environment which is really varied and there can be multiple routes to the same goal, we can see that our algorithm has a very high percentage of accuracy.

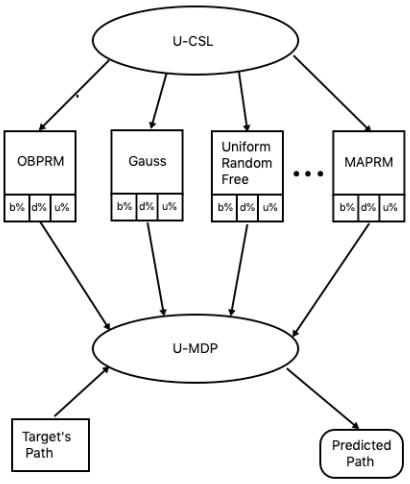


Fig. 1: **U-CLS MDP**

III. Related Work

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

A. Logic Systems

A temporal logic system is one that represents propositions that are defined in terms of time, and reduces the belief space by limiting the number of valid states for a particular timestamp. There have been studies on path planning in dynamic environments using temporal logic for obstacle detection and collision check. This same concept can be exploited for surveillance where the probability of the target being in a certain region depends on time. In [16], a path planning framework was implemented, that satisfy a temporal logic specification consisting of cosafety and safety components for partially unknown environments. The framework employs a "multilayered synergistic planner" to generate trajectories that satisfy the temporal logic specification and adopt an iterative replanning strategy to deal with unknown obstacles. An approach to ensure reachability of a state in a POMDP belief space to a boolean probability was generated in [5]. This method reduced the computational cost of the POMDP method without requiring a conversion to a MDP belief-support system .

However, target localization in an environment is not only time-dependent but is also affected by other factors like the size of the environment. One approach is using subjective logic (SL) which explicitly takes source unreliability and uncertainty into account. SL offers a variety of operators to update opinions consisting of belief, disbelief, and uncertainty. However, as pointed out in [26], SL has low scalability when the network is large-scale, and has high sensitivity to conflicting evidence. Therefore, a hybrid approach was used that includes deep learning (DL)-based dynamic opinion inference model and SL-based node-level opinions. In [6], the authors have discussed a way of minimizing uncertainty in a decision making process by using Probabilistic Subjective Logic (PSL). They have combined multiple opinions concurrently by developing new method called Collective Subjective Logic (CSL), that provides high scalability and prediction accuracy while dealing with uncertain opinions.

B. Motion Planning Primitives and SBMP

In this section we discuss the different sampling strategies that a UGV may use in an environment. 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 within any given environment. In the query phase; the collision free nodes are connected within the graph to create a path from start to goal using an adaptively selected strategy [10].

Probabilistic Roadmap Algorithm (PRM) is a new generation of motion planning algorithms that has been used for the mapping of configuration spaces. A Hybrid PRM is a variant of this algorithm, instead of learning the possible neighbor connection methods, the sampling strategies are selected from an adaptive learning framework. Hybrid PRM also has the option to be configured to run with different sampling strategies. We have used these following sampling strategies as options in the HybridPRM: Obstacle-Based PRM (OBPRM): In OBPRM [1], 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: The Gaussian PRM: [3] is based on the concept of blurring, used in image processing. It generates sampled nodes in difficult regions using simple intersection tests in the work space. It is suitable for many different motion planning problems.

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. A tool belt of sensors gather data on external variables such as obstacles, weather patterns and in certain environments; autonomous and independent entities. This provides the robot 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 [14], Mixed Observability-MDP [7], and CC-MDP [14]. However, POMDP tends to be at the root of these expansionist models. A generic POMDP model can be represented as a tuple of the following (S, A, O, T, Z, R). Where S is the set of states in the environment,

and A is the set of actions the agent can execute, O is the set of observations, T is the transition function, and R is the set of rewards. For example, the POMDP model in [23] 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 ability to track a target.

In [12], 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 modelling target tracking as a POMDP.

IV. Methodology

A. Collective Subjective Logic

Subjective Logic (SL) is a variant of probabilistic logic with a primary focus on the exsitence 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 equations 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), 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 require 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.

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

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

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

By taking the merits of both SL and PSL, a hybrid probabilistic logic algorithm, called Collective Subjective Logic (CSL) was developed in [6], that 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 collective inference of unknown variables in large scale network data.

We introduce a variant to CSL that uses the basic rule constructing principles of PSL, but includes uncertainty in it by using the same logic system as in SL. When we apply 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].

$$P(X_{n+1} = a|_{X_1=a_1, X_2=a_2, ..., X_n=a_n}) = P(X_{n+1} = a|_{X_n=a_n}) \tag{7}$$

In equation (7), $S = \{a_0, a_1, a_2, ..., a_n\}$ is a set of given possible states, where n may or may not be a finite number. These are the values that the random variables X_i may attain. In a discreet time Markov Chain, the subscript i represents timestamps.

This sequence of independent identically distributed random variables is called a $martingale$ [18], [20], where, at any particular time, the next conditional expected value in the sequence, given all prior values, is equal to the present value (see equation 8).

$$E(S_{n+1}|_{S_n}) = S_n \tag{8}$$

Therefore, by the Markov property, equation (9) can be derived, where P_{ij} is the probability to move from state i to state j in one timestamp.

$$P(X_{n+1} = j|_{X_n=i}) = P_{ij} \tag{9}$$

A transition probability matrix can constructed that takes the form $P = [P_{ij}]$. However, this can be generalized to equation (10), where t is the timestamp.

$$P_{ij}^t = [P^t]_{ij} \tag{10}$$

The policy here is to determine the P_{ij} in minimal time. In an optimal case, the number of timestamps t should be equal to the number of states n.
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.

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

We present 2 algorithms U-CSL and U-MDP that takes advantage of our ability to measure the uncertainty present in the environment and exploit it to produce path trajectories in a unique way for location prediction scenarios. These algorithms can be generalised to work both during the sampling, connection and querying stage of sampling based planning methods. We however discuss our algorithm and results for innovations made during the sampling stage.

In our U-CSL model we consider strategies as entities, at each timestamp, each strategy is assigned a positive evidence r, a 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. In order to complete the learning phase and start the prediction phase which utilizes the U-MDP to make a decision. This is determined by the ability to make a decision in either belief E_b or disbelief E_d. If for any sampler the probabilities exceed a certain threshold, we stop the learning phase and move to the prediction phase.

Equations (11) and (12) shows how E_b and E_d are calculated in our CSL variant, 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.

$$E_b[i] \leftarrow b[i] + a[i] * u[i] \tag{11}$$

$$E_d[i] \leftarrow d[i] + (1 - a[i]) * u[i] \tag{12}$$

A. U-CSL

Algorithm (1) uses a calculation of uncertainty, belief and disbelief. 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). The positive evidence r and the negative evidence s is 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 convex hull created by the valid samples for all the strategies. The amount of uncertainty 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 highest expected value in E_b or E_d is stored in $StrategySequence$, and the $StrategyOccurance$ for that strategy is incremented by 1.

Algorithm 1 U-CSL

Input. $Strategies$ is a list of all Sampling Strategies or 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 Samplers.
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 (8) and (9), 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** [$StrategySequence$,$StrategyOccurrance$]

This process is repeated until we find a E_b or E_d for one strategy that gives us a 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. It utilizes the sequence of strategies used by any SBMP method that adaptively selects suitable strategies in turn and applies different strategies with a reinforcement learning algorithm to determine the best suited strategy for any given region within the environment.

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, instead of using a static base state and transition probabilities. After receiving the

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.

1: $[StrategySequence, StrategyOccurrance] \leftarrow GetStrategySequence()$
2: $[S_0, P] \leftarrow constructProbability()$
3: **while** $countIter \leq maxObservations$ **do**
4: $S_{next} \leftarrow S_0 * P^{countIter}$
5: **for** $k = 1$ to NS **do**
6: **if** $S_{next}(k) \equiv threshold$ **then**
7: $flag \leftarrow True$
8: $strategyIndex \leftarrow k$
9: **break**
10: $countIter \leftarrow countIter + 1$
11: **if** $flag \equiv True$ **then**
12: **break**
13: **if** $flag \equiv True$ **then**
14: **for** $l = 1$ to $countIter$ **do**
15: $seeds \leftarrow LocalPlanner(Strategies(strategyIndex), observedNode)$

Output. $seeds$

$StrategySequence$ and the $StrategyOccurance$ from the CSL, the MDP constructs the transition probability matrix by using the $constructProbability$ algorithm previously implemented in our technical report [10]. Once the transition matrix has been created, Algorithm (2) starts predicting which strategy tends to a probability of 1 and after how many timestamps. This is done by using equation 13.

$$S_i = S_0 * P^i, i \in \{1, 2, ..., n\} \tag{13}$$

Once a strategy has been predicted, and the number of timestamps required has been determined, Algorithm 2 - U-MDP then uses this predictive strategy to determine the predicted location of the target using a local planner that takes in the selected strategy 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 [5]. Given that, our goal will be to minimize the number of iterations the U-CSL takes to learn and the U-MDP takes to predict. This can only be done if one of the E_b or E_d attains probability 1 in least possible number of iterations.

Since b, d and w are normalized factors, we can write equation (14). Therefore, from equations (12) and (14), we have equation (15).

$$b[i] + d[i] + u[i] = 1 \tag{14}$$

$$E_b[i] = 1 - (d[i] + u[i]) + a[i] * u[i] \tag{15}$$

$d[i]$ 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 gets merged to the goal location. Therefore, the ability to make a decision under belief or disbelief tends to 1 for a particular sampler and the U-CSL is guaranteed to converge.

C. Complexity Analysis

U-MDP depends on the iterative flow of control in Step 3 and Step 5. The value of $maxObservations$ determines the maximum number of timestamps we can give to make a prediction of the location of the target robot. This should not exceed the number of timestamps the target robot takes to reach the goal location. Therefore, we initialize the value of $maxObservations$ to $2 * NS$ where NS is the number of strategies used, which is twice the number of timestamps for the optimal case of a MDP. So this iteration runs for a maximum of $2 * NS$ times. Therefore, the algorithm runs for a maximum of $2 * NS^2$ times. Hence, the time complexity becomes O(n^2), where $n = NS$. The iteration in Step 14 is dependent on the iteration in Step 3; it runs as many times as the iteration in Step 3 will run, which is a maximum of $2 * NS$ times. Therefore, it doesn't contribute to the time complexity of the algorithm.

VI. Experiments and Results

A. Experimental Setup

The U-MDP algorithm was trained using Heterogeneous Planning Spaces (HPS) [22] strategy. This strategy takes as input a list of node generator methods (sampler) and runs them iteratively to determine a node generator 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. During the training phase of our algorithm (U-CSL), we used one sampler at a time along with the HPS strategy. This combination of HPS with one sampler forms our one strategy in the list of $Strategies$ in Algorithms (1) and (2). The testing was done using either HPS or HybridPRM [11]. 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. For testing our algorithm, we used the HPS or the HybridPRM along with all the samplers compatible with these two strategies. This creates a level of uncertainty as to which node generator 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 all the node generators simultaneously, and the one created by running these two strategies along

with one sampler, will never be the same and 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. Hence the name.
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**: This 3D environment is a room containing serial walls with windows on each one of them, which is the only point of exit, as shown in Figure 2a. In such a room, a cylindrical robot has been placed. We chose this environment because it has visual occlusions from the pursuer's perspective, thus creating more uncertainty and probably missing observations.
- **3D House**: This 3D environment is another room while the robot used is a table shaped one as shown in Figure 2b. The start location is in a free space in one room and goal location is in another room, while there are two narrow doors in between.
- **Cluttered Cube**: This is maze type environment (Figure 2c) where the obstacles are mostly air-borne. We have used three different robots with this environment. One has two degrees of freedom (DOF), one with one has seven DOF, and one has nine DOF. The reason we chose this as a test environment was because of the number of paths that can be generated from start to goal is way to varied for each different strategy. Therefore, it creates a challenge for the learning phase to determine one strategy that matches closely with the path of the target.

B. Results

Table (I) shows the results of running the U-MDP algorithm on different heterogeneous environments. In the results, 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 gives how far the two are with respect to the size of the environment. Column IT gives the number of timestamps taken by the target to reach it's goal, while the column PT gives the number of timestamps taken by the pursuer to intercept the target. Figures 3c, 4c and 5c describe the occurrences of each individual strategy during the learning phase (U-CSL) of our algorithm. Figures 2, 5 and 8 show how the expected values of each individual strategy changes over time. Figures 3b, 4b and 5b show how much time it takes for the state matrix to reach State Equilibrium.
The results table shows that the percentage accuracy is really high for most of the environments like the Serial Walls and the House. For the Cluttered environment, since we vary the degrees of freedom (DOF) of the robot, we have some variance in the paths. As a result, we get low accuracy for the 9 DOF robot. Otherwise, it is fairly high with low DOF. Also, from the columns IT (Invader time) and PT (Pursuer Time), we can see that the pursuer is able to intercept the invader before it reaches its goal. For some environments, it takes a longer amount of time, that is because of the time taken by the U-MDP to stabilize to a State Equilibrium.
Figures 5a and 5b show the two paths taken by the invader and the pursuer, respectively, in the 3D House environment. It show how similar they are and how closely the pursuer follows the invader. It has a 99.19% accuracy in terms of proximity at the point of interception.
The percentage accuracy was calculated as in Equation (16), 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.

$$\%Accuracy = (1 - (\delta(PI - PP)/diagonal)) * 100 \quad (16)$$

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

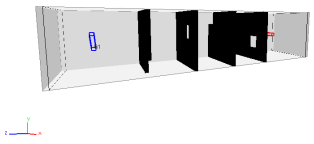
- **Serial Walls:** We ran HPS and HybridPRM in two different experiments for this environment and got a percentage accuracy of more than 98% for both the experiments. The average distance between the path of the invader and the pursuer is also pretty low (1.86 and 1.01) as compared to the size of the environment and its complexity. The number of timestamps the pursuer takes while the invader is using the HybridPRM is higher than when it uses the HPS. This is because of the difference in the cost functions used in both the algorithms. But in both the cases, the pursuer is able to intercept the invader before it reaches the goal.

- **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 and 1.4) as compared to the size of the environment and its complexity. The number of timestamps the pursuer takes while the invader is using the HPS is higher than when it uses the HybridPRM, for this environment.

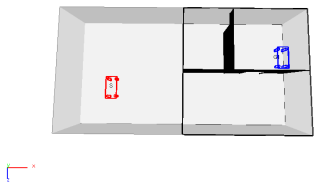
- **Cluttered:** We ran HPS and HybridPRM in six different experiments for this environment with different degrees of freedom of the robot. From the results table, we can see that with higher DOFs, the percentage accuracy gets low, but the time to intercept increases. The average distance between the 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 and its complexity.

VII. Future Work

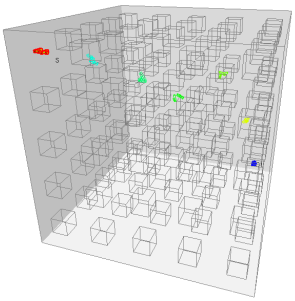
As a future work, we plan on investigating how to limit the list of strategies from the learning phase, because it limits



(a) **Serial Walls**



(b) **3D House**



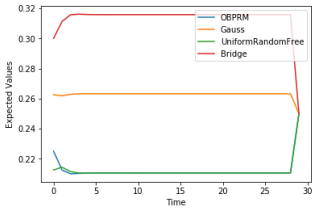
(c) **Cluttered environment**

Fig. 2: Heterogeneous Environments Studied

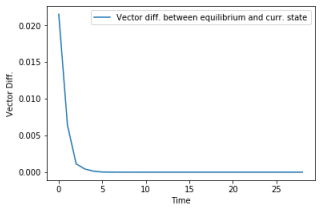
TABLE I: **U-MDP Run On Different Environments**

Env	**PS**	**IS**	**OP**	**PP**	**DIP**	**PA**	**Dist_Avg**	**IT**	**PT**
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

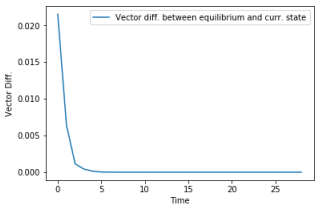
Env Environment used.
PS Sampling Strategy used by the pursuer.
IS Sampling Strategy used by the invader.
OP Observed position of the invader at the point of interception.
PP Predicted position of the invader at the point of interception.
DIP Distance between invader and pursuer at point of interception.
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.
IT Number of timestamps taken by Pursuer to intercept the invader.
HPS Heterogeneous Planning Spaces strategy.
H-PRM HybridPRM strategy.



(a) **All state expected values over time**

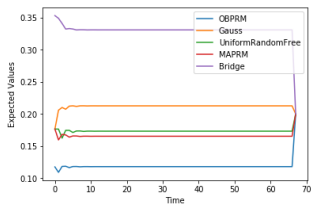


(b) **U-MDP State Equilibrium over time**

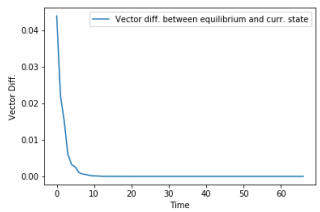


(c) **State occurrences after U-CSL**

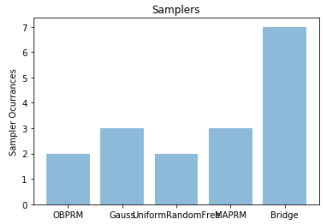
Fig. 3: Serial Walls environment



(a) **All state expected values over time**

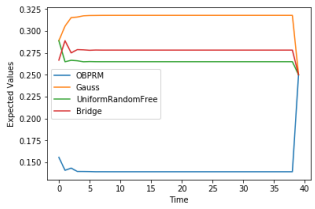


(b) **U-MDP State Equilibrium over time**

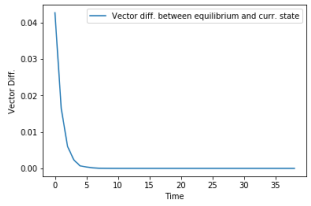


(c) **State occurrences after U-CSL**

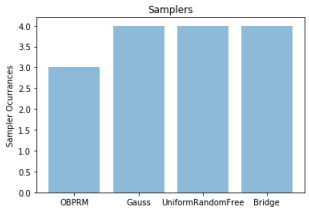
Fig. 4: 3D House environment



(a) **All state expected values over time**

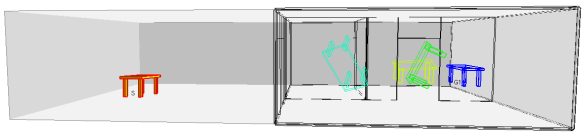


(b) **U-MDP State Equilibrium over time**

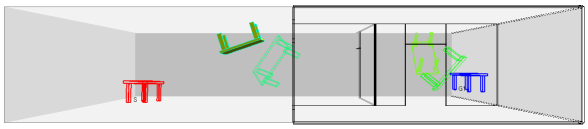


(c) **State occurrences after U-CSL**

Fig. 5: Cluttered Environment



(a) **Path assumed by the Invader**



(b) **Path assumed by the Pursuer**

Fig. 6: 3D House Environment

the number of strategies we can use to predict with our algorithm. We also plan on performing real world applications to test our algorithm in robot retrieval and monitoring scenarios. As an immediate plan, we wish 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] Jerome D Braverman. *Probability, logic, and management decisions*. McGraw-Hill, 1972.

[5] 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.

[6] 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.

[7] 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.

[8] Didier Dubois and Henri Prade. Fuzzy sets. *Neural Networks and Soft Computing, chapter Propagation and satisfaction of flexible constraints*, pages 166–187, 1980.

[9] Robert B Duncan. Mltiple decision-making structures in adapting to environmental uncertainty: The impact on organizational effectiveness. *Human Relations*, 26(3):273–291, 1973.

[10] Sourav Dutta and Chinwe Ekenna. Technical report. `http://www.cs.albany.edu/RACS/paper1.pdf`. Accessed: 2018-9-15.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] Audun Jøsang. Artificial reasoning with subjective logic. In *Proceedings of the second Australian workshop on commonsense reasoning*, volume 48, page 34. Citeseer, 1997.

[16] Morteza Lahijanian, Matthew R Maly, Dror Fried, Lydia E Kavraki, Hadas Kress-Gazit, and Moshe Y Vardi. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Transactions on Robotics*, 32(3):583–599, 2016.

[17] Raymond Reiter. A logic for default reasoning. *Artificial intelligence*, 13(1-2):81–132, 1980.

[18] René L Schilling. *Measures, integrals and martingales*. Cambridge University Press, 2017.

[19] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

[20] Siminelakis. Martingales and stopping times use of martingales in obtaining bounds and analyzing algorithms. *University of Athens*, 2010.

[21] 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.

[22] 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.

[23] 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.

[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] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[26] Xujiang Zhao, Feng Chen, and Jin-Hee Cho. Deep learning for predicting dynamic uncertain opinions in network data. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1150–1155. IEEE, 2018.