Risk-Averse Planning for Marine Robots *

Mahya Mohammadi Kashani[1], Tobias John[2], Jeremy P. Coffelt[3], Einar Broch Johnsen[2] and Andrzej Wąsowski[1]

***Abstract*— Autonomous Underwater Vehicles (AUVs) often operate days without human intervention and must be able to do both: long-term task planning and short-term path planning. Such planning problems are typically solved by optimization with respect to a single variable, e.g. to maximize range or minimize energy consumption. However, optimizing for one variable in the general case might involve a higher risk than a slightly sub-optimal plan. To overcome this issue, we introduce a framework that first generates a selection of different plans and then selects the safest one. We introduce several risk metrics to assess the quality of the plans and we use a realistic underwater robot simulation to estimate the metrics for the plans.**

I. Introduction

Autonomous Underwater Vehicles (AUVs) are unmanned platforms for underwater operations such as gathering data for scientific and commercial purposes. Overtime, the applications of AUVs are expected to expand with manipulation and construction of underwater assets. The promise of autonomy makes them ideal for inspection tasks in harsh environments, as operating in these conditions poses a high risk to human workers. However the high cost of vehicles, the operation in vicinity of high-value assets, and the risk of causing natural disasters introduces a requirement of reliable and safe autonomy [1].

Traditionally, adaptive planning has been a key technology to achieve autonomy. However, mainstream works on planning optimizes for expected rewards. Ignoring the probability of achieving the expected reward (the *uncertainty*) seems overly optimistic for high risk operations. The expected reward might be high, while extremely different outcome may have similar or higher probabilities than the expectation.

Example 1: Consider a marine inspection scenario where the task is to safely inspect multiple sub-sea installations, e.g. the vertical small tanks in Figure 1. During the mission, the underwater robot needs to choose between five paths $P_1, \dots, P_5$. A typical planner here finds that plan (1) is the shortest and selects that. However, if we simulate the different plans, we see that it not only leads in general to a higher execution time than most of the other plans but the variance is also quite high. I.e., there is always the risk that the execution time is way larger than expected beforehand. The

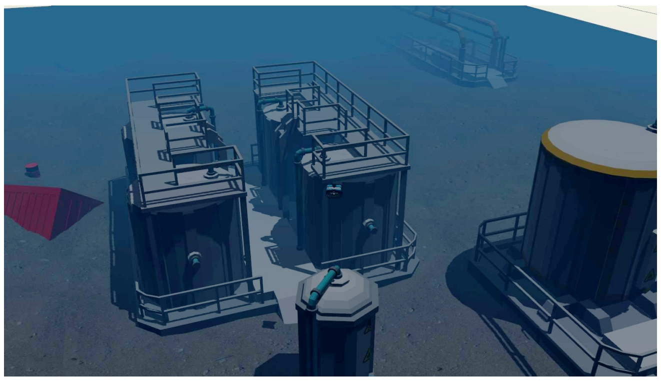


Fig. 1: An underwater scene visualized in Gazebo

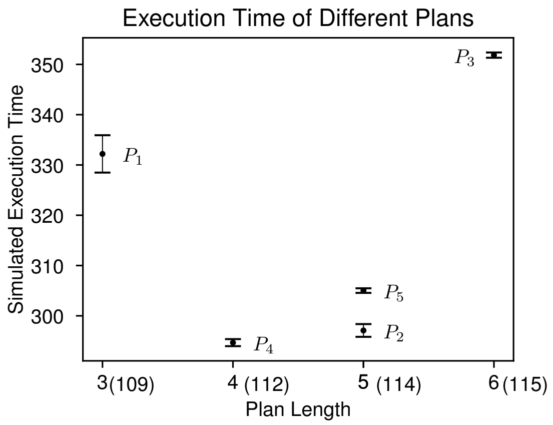


Fig. 2: Comparing simulated execution times of different plans. IDs refer to the ones in Table I. Numbers in parenthesis represents number of way-points

safest plan, with a small expected execution time and a small variance for it is plan 4. This example leads us to investigate and compare our framework with two sides: simulation and planning framework. The current example demonstrates why it is important to have simulations, but not why variance is important.

Unfortunately, although explicit reasoning about uncertainty in planning systems seems necessary in dynamic environments, most existing research work on underwater mission planning, e.g. on re-planning [2], [3] or temporal planning [4], still defines planning problems as optimization against a single objective such as time or energy [5].

Due to the risks and costs, it is necessary that AUV missions are not aborted unnecessarily and consider alternative plans. To this end, we propose a solution that enables marine robots

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956200 REMARO.

[1]Mahya Mohammadi Kashani and Andrzej Wąsowski are with Computer Science Department, IT University of Copenhagen, 2300 Copenhagen, Danmark mahmo@itu.dk, wasowski@itu.dk

[2]Tobias John and Einar Broch Johnsen are with the Department of Informatics, University of Oslo, Oslo, Norway tobiasj@ifi.uio.no, einarj@ifi.uio.no

[3]Jeremy Coffelt is with the research and technology group, ROSEN, Bremen, Germany jcoffelt@rosen-group.com

to autonomously assess the risk in operational conditions by ranking potential plans and choosing the safest plan. The goal of this work is to increase the robustness of planning for underwater robots in dilemma situation. We contribute a framework with the following distinct features:

(i) Generating probabilistic planning problems from sonar sensor data,
(ii) Transforming the planning problem to generate a selection of candidate plans that involve a different degree of risk-averseness,
(iii) Risk evaluation for the candidate plans using a realistic underwater simulation,
(iv) Using several risk metrics for selecting one of the plans.

II. Related Work

An example how to compensate for environmental challenges such as location uncertainty and communication limitations can be found in [6], where the Problem Domain Definition Language (PDDL) is used to model an abstract-level planning problem for Lightweight Autonomous Unmanned Vehicles (see also [7]). Additionally, temporal deterministic planning, namely the OPTIC planner [4] is used to determine plan cost by preferences. OPTIC has consistently produced high-quality plans with the lowest cost. However, it is clear that this planner takes significant time to solve the problem, and planning is repeated several times. To overcome this problem, translating the planning domain into Event-B [8] is suggested, eliminating several deadlock situations, which halved the computation time. The solver computation time can be significantly affected by the manner in which the planning domain is formulated.

There exists work on formalizing a mission underwater planning for allocating tasks to multiple ROVs to distribute tasks among them using existing temporal planners [9]. According to Cashmore et. al., the efficiency of long-horizon mission planning is better and more reliable when the uncertainty is neglected, and re-planning is used instead [10], [11]. However, re-planning is time-consuming and unnecessary if one can check if all preconditioned are achieved [12].

A related topic in planning is explaining runtime planning execution. Cashmore et. al. present a prototype framework for explainable planning as a service for safety-critical domains. Users in such scenarios will not accept explanations generated by planners or models they don't trust because they differ from their own models. To compare the planner's plan and what the user expected, they provided contrastive explanations [13]. Carreno et. al. introduced a class of planning problems that define a set of interesting questions in the context of well-known planning domains and real-world underwater tasks. In contrast, the planning domain has incomplete knowledge and sensing information notions. The authors claim that by analyzing the planning problem with this approach, one can have multiple planning choices at the planning and execution time, although finding appropriate explainable metric(s) still needs to be investigated [12].

There were some attempts to provide variance of cumulative reward and quantify generated policy using mean and var [14]. However, the concept of risk-sensitivity is ill-defined for risk-averse condition. Our approach of risk-averse planning is inspired by [15]. The author provides a risk-sensitivity spectrum from risk-seeking, and risk-aversing. We also bound the risk factor to a smaller interval to make the problem well-defined. We integrate risk-averse planning problems into a hierarchical framework to make it suitable for planning in the marine simulator.

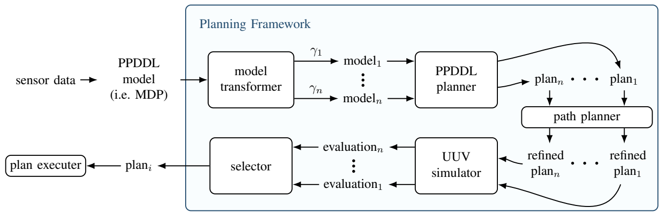


Fig. 3: Flow diagram of proposed method.

III. A Method for Risk-Averse Planning

A. Framework

In our work, we distinguish between high-level task planning and low-level path planner. Both are crucial to have a successful marine mission. Each mission consist of a high-level plan, which uses sub-plans generated by some low-level planners, e.g. path planners. Such low-level plans can be used for docking (from or to the docking station), for generating inspection trajectories around objects of interest such as the fuel tanks shown in Fig. 1, and for following way-points selected by a robot operator. The problem addressed by (high-level) AI planning is to synthesize a plan to reach a state where the desired goals are achieved, given descriptions of the initial state of the world and a set of possible actions. Symbolic AI planning is advantageous if the problem can be described declaratively and non-trivial domain knowledge exists and is relevant. Explaining a particular course of action that the system takes, is also important and supported by planning methods [16].

Our proposed approach is depicted in Figure 3. We assume to have a probabilistic model of the current situation of the robot, which was extracted from sensor data, e.g. sonar data. This also involves the possible actions that the robot can execute and the goal of the mission, e.g. to inspect all the underwater infrastructure of interest. We generate different plans (with different sensitivity to risk) and use a (low-level) path planner to refine the high-level actions of the generated plans. Afterwards, we evaluate the performance of the refined plans using simulation runs. The generated evaluation for the different plans allows us to choose the plan that blanches best the involved risk and expected performance.

B. Plan Generation

We consider as models problems defined in *Probabilistic Programming Domain Definition Language* (PPDDL), which are a representation of *Markov decision processes* (MDPs) [17]. We consider an MDP as a 5-tuples $\mathcal{M} = (S, A, \mathbf{P}, s_0, \mathrm{R})$ where S is a finite set of states, A a finite set of actions,

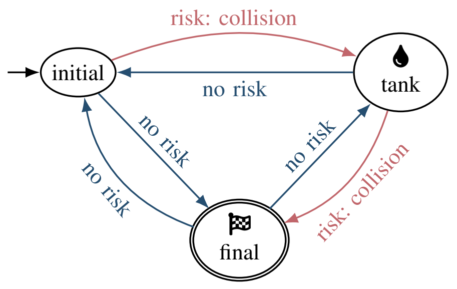


Fig. 4: Example scenario that can be described with an MDP. It includes three waypoints, which are the critical ones for the mission.

$\mathbf{P} : S \times A \times S \rightarrow [0,1]$ the transition function, $s_0 \in S$ the initial state and $\mathrm{R} : S \mapsto \mathbb{R}_{\geq 0}$ the reward. A PPDDL problem also defines a set of goal states $G \subseteq S$ that the plan should reach.

Example 2: An example of how we use MDPs as a model is shown in Figure 4. It depicts a scenario with where the robot can navigate between three waypoints and its mission is to inspect all tanks to make sure that there are no leakages. There is a non-deterministic choice, to which waypoint the robot moves next. Some transitions between waypoints involve risks, i.e. the AUV collides with some infrastructure with some probability. The goal state is to reach the waypoint "final" and to have performed an inspection at the waypoint "tank". The cost is the length of the generated plan.

Given a plan $\pi : S \mapsto A$, nondeterminism in the MDP can be resolved to obtain the induced *Markov chain (MC)* $\mathcal{M}_\pi = (S, \mathbf{P}_\pi, s_0, \mathrm{R})$ where for each pair of states s, s' the transition probability is defined by $\mathbf{P}_\pi(s, s') = \mathbf{P}(s, \pi(s), s')$. A *history* h is a finite sequence of states $h = (s_0, s_1, \ldots, s_n)$. Define a history's *probability* by $\mathbf{P}(h) = \prod_{i=0}^{n-1} \mathbf{P}_\pi(s_i, s_{i+1})$ and its *reward* by $\mathrm{R}(h) = \sum_{i=0}^{n-1} \mathrm{R}(s_i)$. For goal states $G \subseteq S$, the corresponding reward of the Markov Chain, $\mathrm{R}_\mathcal{M}(G)$, is a discrete random variable with:

$$\Pr\Big(\mathrm{R}_\mathcal{M}(G) = x\Big) = \sum_{\substack{h=(s_0,\ldots,s_n) \\ s_n \in G; s_0,\ldots,s_{n-1} \notin G \\ \mathrm{R}(h)=x}} \mathbf{P}(h)$$

For a given MDP model, we first generate a set of different *candidate plans*. These candidates trade some of the expected reward for a lower risk. Our framework is agnostic to the algorithm to compute the different candidates.

In this paper, we generate candidate plans using Koenig and Simmons' approach [15] to generate risk-averse plans with a non-linear utility function for cumulative rewards. This function values a difference between high rewards less than the same difference between smaller rewards; i.e., the cumulative reward of a few "best-case" paths has smaller impact on plan selection than many paths with a decent cumulative reward. The utility function depends on a parameter γ, which balances expected reward and involved risk. The main advantage of this construction is that we obtain a utility function for the cumulative reward for risk-sensitive planning by only making local changes; i.e., the probabilities of the MDP's transitions are replaced by values that take the probability, immediate reward and the risk sensitivity into account.

Definition 1 (Transformed Transition System): Given an MDP $\mathcal{M} = (S, A, \mathbf{P}, s_0, R)$ and a parameter γ with $0 < \gamma < 1$, we define the transformed transition system $\mathcal{M}^\gamma = (S, A, \mathbf{P}^\gamma, s_0)$ where $\mathbf{P}^\gamma : S \times A \times S \rightarrow [-1, 0]$ with

$$\mathbf{P}^\gamma(s, a, s') = \mathbf{P}(s, a, s') \cdot \Big(-\gamma^{R(s)}\Big)$$

Although this transition system is not an MDP, standard algorithms can be used to find the plan that maximizes the "pseudo-probability" to reach a goal state [15]. This construction is well-suited for our setting, as we consider PPDDL domains to describe the MDP and the transformation described in Def. 1 can be done at the level of the actions in these domains.

We can now iterate over different values of γ and call a generic PPDDL planner to generate the optimal plan for each value. This "optimal plan" is simply the plan that maximizes the probability of reaching one of the goal states. For different values of γ, we get different plans, which form our set of candidates.

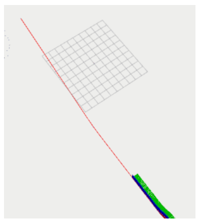
A crucial point of our approach is that plans are always generated w.r.t. a limited number of risk metrics, as planners can only handle a limited number of objectives at a time. In our specific case, only the parameter γ is considered. This risk measurement might—but need not—correlate to other risk metrics, e.g., variance or entropy of the cumulative reward. Therefore, we propose different methods to asses candidate plans across *all* risk measurements of interest.

C. Plan Evaluation

To select the best plan, we use metrics based on the probability distribution $D = \mathrm{R}_{\mathcal{M}_\pi}(G)$ of the reward to evaluate each of the candidate plans. One approach for this is to utilize Monte Carlo simulation in the MDP described by the planning problem. However, this is insufficient for autonomous underwater robots since formalizing an MDP model for robotic behaviour always involves simplifying assumptions of their physical environment. For AUVs, this would require neglecting critical considerations such as unpredictable water currents, GPS-denied vehicle localization or noisy acoustic sensors. In order to overcome such limitations, we use an *underwater physics simulator* that also provides increased realism. In order to get a good estimation of the risk associate with the plans, each generated plan needs to be simulated multiple times. The simulator uses random variables for environment conditions, e.g. underwater currents or water visibility, to test the robustness of the plans.

We use different metrics to assess the current and anticipated risks for the above mathematical model.

1) The *expected reward* $\mathbb{E}[D]$ is typically a primary concern when selecting a plan. Even in risk-averse settings, a decent expected reward is required.


(a) Dangerous but short path.

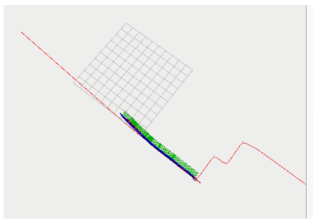

(b) Safer but longer path.

Fig. 5: RViz Demonstration of (a) a dangerous path that optimizes plan length and (b) a safer path.

2) The *variance* of the reward $\mathbb{E}[D - \mathbb{E}[D]^2]$ is an often used measurement for risk [18]. The higher the variance, the more risk is involved.
3) The *entropy* of the reward $-\sum_{x \in R} \Pr(D = x) \log_2(\Pr(D = x))$ is another common metric used to measure the uncertainty or "surprise" or a reward. In general, a generalization of entropy to continues variables might be needed.
4) The *reward-bounded probability* $\Pr(D \leq b)$ is the probability that the reward falls below a given bound b. This metric allows us to estimate how often bad runs occur.
5) The *Value at Risk (VaR)* is designed to capture tail risk. It provides the maximum reward that can be expected given a time horizon and a confidence level.
6) The *Expected Shortfall (ES)* is the average of the worst (1 - b%)ile of losses. It overcomes some shortfalls of VaR. VaR does not tell us what we may lose if a bad day occurs, unlike ES.

After simulating the candidate plans and estimating the risk metrics of interest, we select the plan that balances the different metrics best. Afterwards, we hand over the selected plan to the plan executor, which runs the plan on the actual robot. This involves using low-level planners to refine the actions in our generated high-level plan.

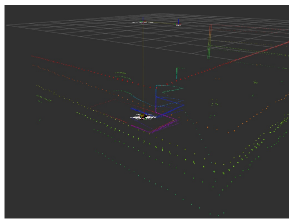
IV. Experimental Results

A. Design

The goal of our experiments is to demonstrate that our framework is able to generate the safest plan for an AUV given an inspection mission, even if this plan is not optimal w.r.t. to the expected cost.

We consider plan length as our cost and the risk of colliding with the infrastructure. Therefore, navigating close to objects, such as navigating through a gab between tanks as depicted in Figure 1, can be part of a plan with low cost but high risk.

Example 3: We consider the scenario from Figure 1. The robot starts somewhere in the distance on the top the illustration and the goal is to navigate and inspect the tank on the bottom of the illustration. Figure 5 shows two representative plans that demonstrate the trade-off between expected cost and risk. The plan on the left, where the AUV navigates through the tanks, is shorter. The slightly longer plan on the right, where the AUV navigates around the large vertical tank clone, is safer and thus preferred. We demonstrate with our experiments that our framework is able to find plans like the one on the right.


(a) High resolution mapping.

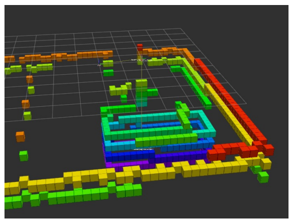

(b) Low resolution mapping.

Fig. 6: Some examples of using OctoMap [19] for seabed reconstruction. Each Voxel represent a probabilistic occupancy grid. In (a) we demonstrate high resolution settings (10000 samples with 0.05 resolution), which is good for path planning, (b) we demonstrate low resolution, which is sufficient for high-level planning.

The first step for high-level planning is obtaining a PPDDL model of the scenario (see Section III). For our experiments, we fix the goal of the mission and the actions that the AUV can perform. The actions involve moving not only from one waypoint to another but also inspecting objects at 50 waypoints using a helical trajectory around the object. The details of the actions are created using low-level planners after the high-level plan is obtained. We generate the waypoints from simulated data. One crucial step to make the planning more efficient, is to limit the amount of states. To keep the state space as small as possible, we do not use all waypoints for the high-level planner, as the low-level path planner does, but we only consider critical waypoints. Such points are the ones close to one of the fuel tanks or between two narrow infrastructure objects.

After obtaining a model, we generate transformed versions of it. To do that, we sample risk factors γ with a random uniform distribution over $[0.4, 1)$. A value of 1 means that the planner only tries to optimize expected cost, while a value of 0 would mean that the planner only tries to minimize the cost that can be guaranteed. Our interval provides a broad spectrum of plans, while still ensuring that the plans optimize the expected reward sufficiently.

We use an of-the-shelf PPDDL planner to generate plans for the generated transformed models. Afterwards, we refine the plans by instantiating the actions using low-level planners for path planning. After running the refined plans in Unmanned Underwater Vehicle (UUV) simulations, we estimate the metrics expected value variance and entropy. We select the plan based by defined metrics.

B. Setup

The source code to reproduce our experiments can be found online,[1] We run the experiments on a computer with *CPU Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz* and *32 GB RAM*, running *OS Ubuntu 20.04*. We use *ROS1 Noetic*,[2]

[1]https://github.com/remaro-network/risk-averse_planning
[2]https://wiki.ros.org/noetic/Installation/Ubuntu

Gazebo 11,[3] a modified version of UUV simulator[4] to create our simulation environment.

We created different scenarios for gas and oil infrastructure inspection by using public assets created in the DAVE project.[5] The 3D models can be obtained on the TurboSquid website.[6] To get the waypoints for the model, we use OctoMap [19] to collect the probability that a section of the environment is empty or occupied using simulated forward multibeam p900 sonar data.[7] We do this with different resolutions. Figure 6 demonstrates this process. Each color represent a probability between 0 and 1. The cubes are called voxels and they are used to form the initial state of our planning problem.

We use a modified version of safe-planner to generate different risk-averse plans [20]. The modifications where necessary to incorporate the usage of the risk parameter γ. The output for each generated plan consists of a JSON file, which includes the plan and a stat file that documents the performance of the produced plan.

We use LIPBInterpolator (Linear Parabolic Interpolator) for generating trajectories[8] as a (low-level) path planner to refine the actions of the generated plans by generating intermediate waypoints.

We also run each resulting plan 10 times to evaluate its performance. We change positions of vertical small tank; e.g. *X, Y, and Z* of the simulation environment slightly randomly to estimate the distribution of the risk metrics w.r.t. Expected Value, Variance, and Entropy. In the end, we select the best plan as described in Subsection IV-A.

C. Simulation

We provide a scenarios as shown in Figure 1 in Gazebo simulation. In the simulation, we spawn 6 different inspection points including in front of *large vertical tank*, inside of *platform*, in front of *tank pairs*, in front of and behind *quad tank*, next to the *large vertical tank clone* and in front of *small vertical tank*.

In this scenario, we have 5 different strategies or plans that underwater robot can inspect suspect area safely while aversing risky places. There exist 4 critical waypoints: (1) in right of the small vertical tank, (2) in front of the narrow path in the quad tank, (3) between the quad tank and the largest tank and (4) left of the small vertical tank. We also model extra energy consumption when the robot speeds up in the safer areas. Therefore, the reward can be different depending on whether the plan is risky or not. Therefore, as you see in Figure 2, although we have shortest plan with the length of 3, the risk of this plan is high because of high variance. According to Table II, scenario with ID 4 present itself as a safest plan. This ID 4 corresponds with P_3 presented in Figure 2. We use two different type of actions including *Following Waypoint* and *Collision Detector*. Our *Following Waypoint* follow some waypoint that we generate for each type of plans, e.g. shortest plan, run out of battery plan, safest plan, longer and dangerous plans. you can see these types of plans in Figure II. *Collision Detector* is measuring distance between robot and close infrastructures and inform us by logging in WARN or COLLISION area developed in ROS1.

D. Results

In table I, we present a schema of generated plans and sub-plans w.r.t the depth for scenario 4 with 6 critical states (as mentioned in Table II). A few of the plans are part of other schemes, which is interesting as we can generate new plans by combining them. However, one needs to be careful to ensure that the new plan is still solvable. The plan with ID 4 from the Table I contains the safest plan. Each schema has ID number which is as the same as id of plans from Figure 3.

Table II shows properties of the optimal generated plan in different scenarios. All scenarios contain between 109 and 115 waypoints but differ in the number of critical states, which are the ones represented in the high-level planning problem. The scenarios have an increasing complexity, as one can see on the rising number of critical states. All our scenarios are solvable, i.e. we can generate a plan to reach the goal of the mission. One of the most common problems with *probabilistic planners* is the possibility that planning cannot exit from cycling. Our approach guarantees that we can produce different plans or even sub-plans if this cycle takes less than 60 seconds to complete. We expect all of our scenarios found at least a solution with different Cost or plan length. We also have experiment to increase branches for the structured tree of proposed planner. Since our proposed framework is based on BFS, the planning time would be increased exponentially. Our domain-specific risk achieving reward is to find safest inspection plan, we haven't done much experiments; these expanding branches of tree are useful for ordered task like manipulation task.

We provide some properties of problem and also properties of solution for each scenario. we called $variable_i$. In all scenarios, Path planner generate between 109 and 115 waypoints depends on plan length. One of challenging aspect of high-level task planner is whether this plan is solvable or not. Since we use safe-planner as a backbone of our strategy, all our scenarios satisfied being solvable. Values of risk below 1 indicate that safest plan is not the one that minimizes cost. Risk is also generate along with *plan generation* based on random uniform distribution. Since risk-averse planner is based on BFS, we almost have a tree with depth equal to number of *States*. Another interesting aspect of our solution is that increasing number of states and critical states could not affect on our generating plan execution time. But whether we have a lot branches or not can increase the complexity of problem like as scenario 9 to 15 in Table **??** that we increase the branch from 1 state to 3 states. *Planning Time* can be affected directly with *Cost* or plan length as we see at the table in those columns. The number of plans generated by our

[3]https://classic.gazebosim.org/tutorials?tut=install_ubuntu
[4]https://github.com/mahyamkashani/uuv_simulator
[5]https://field-robotics-lab.github.io/dave.doc/
[6]https://www.turbosquid.com/3d-models/3d-fuel-tank-1443266
[7]https://github.com/Bluerov2/MASTER
[8]https://uuvsimulator.github.io/packages/uuv_simulator/docs/python_api/uuv_trajectory_generator/lipbinterpolator

TABLE I: Generated plans using risk-averse planning. We have 6 types of infrastructures in the simulation. The *lg tank* and *sm tank* are abbreviation of large vertical tank and small vertical tank.

ID	schema	computation time (s)		plan length	execution time		
		planning	simulation		mean	variance	entropy
1	$lgtank \rightarrow tankquad \rightarrow smtank$	0.09	0.1433	3	332	13.8	2.3
2	$lgtank \rightarrow tankpairs \rightarrow platform \rightarrow quadtank \rightarrow smtank$	0.11	0.1391	5	297	1.6	2.3
3	$lgtank \rightarrow tankpairs \rightarrow lgtank \rightarrow platform \rightarrow quadtank \rightarrow smtank$	0.147	0.1390	6	351	0.3	2.3
4	$lgtank \rightarrow quadtank \rightarrow lgtankclone \rightarrow smtank$	0.067	0.1373	4	294	0.2	2.3
5	$lgtank \rightarrow tankpairs \rightarrow lgtank \rightarrow quadtank \rightarrow smtank$	0.108	0.1361	5	305	0.2	2.3

TABLE II: Different scenarios for risk-averse planning using between 109 and 115 way points. All scenarios are solvable. The output is the safest plan.

ID	Property of Problem		Property of safest plan		
	Depth (State)	#Critical States	Cost (Plan Length)	γ (risk parameter)	Planning Time (s)
1	3	3	5	0.95	0.05
2	4	4	5	0.74	0.06
3	5	5	6	0.40	0.11
4	6	6	17	0.57	0.10
5	10	5	13	0.67	0.11
6	15	5	13	0.49	0.14
7	20	6	17	0.84	0.08
8	30	10	52	0.58	0.13
9	40	15	66	0.84	0.16
10	50	15	86	0.96	0.21
11	60	15	106	0.97	0.29
12	70	15	126	0.95	0.40
13	80	15	146	0.59	0.54
14	90	15	166	0.57	0.93
15	90	25	166	0.42	0.98
16	90	35	166	0.75	1.04

proposed framework depends on two factors: one, number of states and two, number of branches in each row can affect our final result. As in pipeline inspection, we don't usually have more branches this part, cannot affect the desired result for us.

V. Conclusions

In this research work, we present a risk-averse planner which not only produces different plans but also measures risks from them. We design various scenarios and generated the safest plans for them. Having generated and measured risks, we demonstrate, using a physic-based simulator, that there exist different plans with different risk values.

As a future work, we integrate all components of proposed framework to have an end-to-end framework. Besides, we will do more experiments with our introduced metrics such as *VaR* and *ES*.

References

[1] X. Chen, N. Bose, M. Brito, F. Khan, B. Thanyamanta, and T. Zou, "A review of risk analysis research for the operations of autonomous underwater vehicles," *Reliability Engineering & System Safety*, vol. 216, p. 108011, 2021.

[2] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[3] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[4] J. Benton, A. Coles, and A. Coles, "Temporal planning with preferences and time-dependent continuous costs," in *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[5] M. Cashmore, M. Fox, T. Larkworthy, D. Long, and D. Magazzeni, "AUV mission control via temporal planning," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 6535–6541.

[6] L. Steenstra, "PDDL-based task planning of survey missions for autonomous underwater vehicles: A generic planning system, taking into account location uncertainty and environmental properties," Master's thesis, TU Delft, 2019. [Online]. Available: http://resolver.tudelft.nl/uuid:5bc6b1f4-1acf-4b40-8154-828427f9a4e9

[7] A. Sousa, L. Madureira, J. Coelho, J. Pinto, J. Pereira, J. B. Sousa, and P. Dias, "LAUV: The man-portable autonomous underwater vehicle," *IFAC Proceedings Volumes*, vol. 45, no. 5, pp. 268–274, 2012.

[8] F. Fourati, M. T. Bhiri, and R. Robbana, "Verification and validation of PDDL descriptions using event-b formal method," in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE, 2016, pp. 770–776.

[9] Y. Carreno, È. Pairet, Y. Petillot, and R. P. Petrick, "A decentralised strategy for heterogeneous AUV missions via goal distribution and temporal planning," in *Thirteenth International Conference on Automated Planning and Scheduling*, 2020.

[10] M. Cashmore, M. Fox, D. Long, B. C. Ridder, and D. Magazzeni, "Opportunistic planning for increased plan utility," in *Proceedings of the 4th ICAPS Workshop on Planning and Robotics (PlanRob 2016)*, 2016.

[11] M. Cashmore, M. Fox, D. Long, D. Magazzeni, and B. Ridder, "Opportunistic planning in autonomous underwater missions," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 519–530, 2017.

[12] Y. Carreno, A. Lindsay, and R. Petrick, "Explaining temporal plans with incomplete knowledge and sensing information," in *ICAPS 2021 Workshop on Explainable AI Planning*, 2021.

[13] M. Cashmore, A. Collins, B. Krarup, S. Krivic, D. Magazzeni, and D. Smith, "Towards explainable AI planning as a service," *arXiv preprint arXiv:1908.05059*, 2019.

[14] M. Sato, H. Kimura, and S. Kobayashi, "TD algorithm for the variance of return and mean-variance reinforcement learning," *Trans. Jap. Soc. for Artificial Intelligence*, vol. 16, no. 3, pp. 353–362, 2001.

[15] S. Koenig and R. G. Simmons, "How to make probabilistic planners risk-sensitive (without altering anything)," in *Proc. 2nd Intl. Conf. on Artificial Intelligence Planning Systems*. AAAI, 1994.

[16] J. Pitrat, *Artificial beings: the conscience of a conscious machine*. John Wiley & Sons, 2013.

[17] H. L. S. Younes and M. L. Littman, "PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects," Carnegie Mellon University, Tech. Rep. CMU-CS-04-167, 2004. [Online]. Available: http://reports-archive.adm.cs.cmu.edu/anon/anon/home/ftp/usr0/ftp/2004/CMU-CS-04-167.pdf

[18] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.

[19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[20] V. Mokhtari, A. S. Sathya, N. Tsiogkas, and W. Decré, "Safe-planner: A single-outcome replanner for computing strong cyclic policies in fully observable non-deterministic domains," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 974–981.