Final Project - Theoretical Report
AI and Robotics (236609)

Goldstein, Sharon
sharongold@campus.technion.ac.il
Computer Science & Physics
Technion - Israel Institute of Technology

Granov, Yotam
yotam.g@campus.technion.ac.il
Mechanical Engineering & Physics
Technion - Israel Institute of Technology

March 29, 2023

1 Introduction

Ensuring safety in motion planning critically depends on the quality of information about the environment of the robot. Unfortunately, the accuracy of inference is often poor because it depends on observations, probabilistic models, and learning methods. We want to propose a safe planning scheme that uses uncertainty to ensure safety.

We are looking for safe real-time planning in a generally mapped environment – the map limits are known but the map is not fully given a-priori and might include static and dynamic obstacles. These dynamic obstacles can include other robots in a distributed system. The map unfolds by sensor readings, and previously mapped areas might change over time. This setting fits systems that drastically change over time like search and rescue missions, civil missions, warehouse missions, driving on unpaved roads, and more. The given examples and more usually include people or valuable objects therefore, the relationship between safety and optimality is important.

2 Related Work

In general, computing an open-loop trajectory that reaches a desired goal cannot be solved in closed form since the dynamics and constraints can be non-linear and the environment, with dynamic and static obstacles is not fully known. Arbitrary obstacles can be non-smooth and require special differentiation [1] to guarantee convergence.

A known alternative is to discretize the state space and use sampling-based planning. When parts of the map are unknown or when the area includes rough terrain, some researchers used sensor readings to check traversability and consider it during the planning tree expansion. Some work utilized its sensors readings even more for Traversability-Based RRT* [2] and used point cloud data to ensure feasible nodes by directly sampling a point (node) from the point cloud.

A discretized grid of the space can function as a traversability map. The discrete traversability map is often expanded to continuous value in the range of [0.0, 1.0] where 0.0 is the most traversable, increasing values up to 1.0 indicate lower traversability and values exactly equal 1.0 indicate obstacles. Some researchers [3][4] used this map to develop traversability Hybrid A* that adds extensions to Hybrid A* algorithm.

Cost maps or risk maps evolved to hold traversability map and further risk analysis - now planning can be risk aware. It can also be solved with conditional Value at risk (CVaR) tools. Some researchers map kinds of dynamic obstacles movement and build a collision risk map according to the obstacles state and the robots. A group from Korea [5] presented a neural network approximation of the risk map proposed to reduce the computational cost of solving the planning problem. Their mapping assumes Gaussian process regression.

A group of researchers from NASA [6] presented full robot stack for a safe, feasible and fast trajectory in real-time. The stack includes risk map processing, geometric CVaR or basic graph based path search and Kinodynamic MPC planner. They begin their risk-aware kinodynamic planning using the best candidate trajectory from a trajectory library that includes the previous iteration trajectory, heuristically defined trajectories and more to make it an any-time algorithm.

3 Background

We define a risk map m as a grid matrix in the world map, that holds continuous values between zero and four when the value 4 symbolizes an untraversed obstacle. We use a learned policy, the set of rules or strategies used by a learning planner agent to determine its actions in an environment based on the information it has learned.

4 Problem Statement

In this work, we consider the world as a discrete-time and discrete-space system:

$$x_r(t+1) = f(x_r(t), u_r(t), w_r(t)), z_r(t) = h(x_r(t), v_r(t))$$

where $x_r(t) \in X$, $u_r(t) \in A$ and $y_r(t) \in R^{n_y}$ are the robot's state, input, and output. $z_r(t) \in Z$ is the observation of the robot and w_r, v_r model the transition and measurement uncertainty. We solve a 2-D motion planning, therefore the position part of the state is a grid placement. Given a known occupancy grid map $m : X \to R$, we define a stage reward function $R : X \times A \to R$.

Based on the problem formulation, the uncertain motion planning problem fits in a POMDP framework: $(X, A, Z, T, \gamma, O, R, b_0)$ where $T : X \times A \times X \to [0, 1]$ and $O : X \times Z \to [0, 1]$ are the transition probability and measurement likelihood with the rest of elements defined. If other robots in the system or dynamic obstacles are identifiable, they will be described similarly when their movement function is known or possibly unknown function but can be learned. Given the robot's initial state and a goal state it wants to reach, we want to plan a fast, safe trajectory that will in high probability, reach the goal place in the wanted state.

5 Solution Approach

First, we want to propose a dynamic robust risk map that holds continuous values. Second, we will show how one can use the map for cost/loss calculations and for energy calculations. We propose a scheme for planning in a mostly unmapped field that utilizes the risk map.

5.1 The Dynamic Risk Map

The risk map can act as a barrier for multi-agent or distributed systems by including a big part in the mapping communication and allowing more modular planning algorithms for different communication levels. We will explain the update and usage of one robot, but in a larger system, different robots can update the map simultaneously.

The initial values of the map will be initialized by the generally known map. An untraversed pixel in the general map will be assigned a value of 4 (maximal), and any other with a value of 2 (middle point). If an area is mapped by the robot's sensors, the map values will be calculated by the readings, and estimated movements according to their certainty. Old readings effect reduces over time according to a time parameter $0 \leq t$ and returns to the initial value. Higher time parameter values are set when the system includes a lot of uncertainty or a large number of possible dynamic obstacles. Lower time parameter can be set when the scanning process is slow.

Since most of the map is unknown, the risk map considers the neighbor cells value, when the neighbors' value effect according their distance from the relevant pixel. This smooths the readings, prevents sensor noise, and reduces the probability to enter an obstacle-filled, unmapped area, small corridors and more. The number of neighbors taken into account can be controlled by the radius parameter. When set to zero, no neighbors are included.

Algorithm 1 Risk Map Update

function RiskMapUpdate(sensor readings, time param. t, radius param. r, weights params.)
  for pixel in the sensor scanned area **do**
    if the pixel **is** an obstacle **then** $m[pixel] \leftarrow 4 - STD$
    if the pixel **is** traverse **then** $m[pixel] \leftarrow STD$
    if a risk calculation for the given problem **then**
      $m[pixel] \leftarrow m[pixel] + Problem\_Risk(pixel, m)$ ▷ assumed to be normalized
  for pixel outside of the sensor scanned area **do**
    if $m[pixel] < 4 - \varepsilon$ **then** $m[pixel] \leftarrow (2 - m[pixel]) \cdot 0.5^t$ ▷ not a sure obstacle
  if specific dynamic obstacle with estimated movement **then**
    reduce its pixel value and add to the next estimated state, according to the state prob. and the primal obstacle risk value ▷ can be replaced by adding a collision risk map
  $m[pixel] \leftarrow Weighted\_Mean(m, pixel, r)$

5.2 The Risk Map's Effect on Planning

We propose three steps planning scheme that divides the planning mission into smaller problems, utilizes the risk map for each step, and gives us safer, smoother movement. The steps include a geometric planner, a physical trajectories planner, and a learning-based chooser between trajectories. Providing only one path to optimize on can be good for mostly open fields, as in NASA's problem but might bring the robot to a dead-end in other areas, filled with obstacles, like in our case. Therefore, we develop a number of trajectories using a similar pipeline and choose the trajectory with lower risk cost and lower predicted change for a dead-end.

For a geometric planner, since the risk map is dynamic, an online-based algorithm for computing a path from the robot's starting point to the goal point is optimal. At first, some nodes will be generated near the goal and most of the nodes will be sampled near the robot according to the risk map. The sampling process will look at an area around the robot, without sure obstacles. The robot's movement abilities can also remove a part of the area. Now we can use the risk map in the sampled area as a

probability mapping for the probabilistic node sampling. This will provide us more nodes in a safer area and more feasible paths. An integral of the normalized risk map over an edge also acts as a heuristic, as done in previous research.

For a given map, we provide around 5 geometrically possible paths. If the general map holds helpful information in this area, one path can be added from the original map. For feasible trajectory planning according to the path, with physical constraints, we can look at the problem as a minimizing problem for CVaR cost based on the risk map while staying as close to the path as possible. Unfeasible paths will be removed and we will choose the preferable path with a learning-based chooser.

The chooser is a learning-based planner, that was trained based on the risk map. After a couple of test drives of the robot, we will have a couple of fully mapped scenarios. We will use their risk map to plan. Augmentations of the risk map values will allow us to have large number of different scenarios without simulating the total environment from scratch. The loss function is basic, and will include a negative reward as time passes, a negative reward for collision, a negative reward according to the integral over the risk map, end of simulation and a big negative reward for hard collision or getting stack and a high positive reward for reaching the goal. The chooser will learn to prefer paths that will probably prevent getting stack in a future unmapped area. Its preference will provide the final choosing between feasible paths. To prevent unsmooth movement, the final choice can apply a preference to a path that is near the previously planned path and trajectory.

It seems that this choosing planner can be used to plan the total trajectory and there will be no need for the other steps. Our solution allows the choosing planner to focus on a smaller problem, this allows us to use a simple loss function and less real-world data to train on. Adding feasibility considerations in the loss function might add the need for a greater amount of data. This planner can produce a path that can be added to the paths given by the geometric planner.

6 Formal Guarantees

The geometric planner is a sampling-based planner, algorithms like RRT* and Informed RRT* guarantee completeness and optimality in regard to the mapped areas. Dynamically sampling the nodes as the robot moves allows smarter sampling according to the relevant time and provides a better representation of the risk as time passes.

If sampling-based planning is too slow, we can use other paths given (if exist), such as a path based on the general map, the previously chosen path, the chooser path and more. Trajectory planning using the best candidate from a trajectory library that includes the previous iteration trajectory, heuristically defined trajectories and more to start from makes the trajectory planner, and from its point, our scheme as well, an any-time algorithm.

7 Empirical Evaluation

To empirically compare our approach to previous approaches, simulated scenarios of different fields can be made, like the making of scenarios for the chooser learning, but now including physical responses. In our scheme, only the chooser and other approaches will be operated. The first comparison criteria will be the percentage of simulated robots that reached the goal state. Other criteria will compare arrival time, distance from optimal solution, and the total risk of the movement according to a full risk map based on the total known map.

Energy optimization, some types of learning and other risk analysis can be done with risk contours maps or continuous representations. There are traversability risk contours mapping for risk-bounded motion planning under perception uncertainties and continuous dynamic obstacles collision risk maps. Further thinking towards a continuous risk map needs to be made and to check if the calculations added time benefits the scheme.

8 Conclusions

We see that one trajectory planning might cause dead-ends, or prevent a robot from entering a dead-end area when it can be prevented. The separation between geometric and feasible planning seems to shorten the calculation time with regards to other known solutions that optimize the problem at once. This separation allows the scheme to be partially any-time while providing a good start solution.

Further testing and improvements to the chooser loss is needed. Given the correct loss and scenarios to learn with, the chooser can become a fast but non-optimal geometric planner as well. A time dependent risk map might add overhead to planning but is very useful in a dynamic or mostly unknown map. The effects of number of robots updating the map simultaneously, especially when some dynamic obstacles behavior is known is interesting and can be further checked.

There is a costmap-based approach to deep reinforcement learning mobile robot navigation, the effect of our time dependent risk map in the costmap might be interesting. Given the scheme risk cost calculation and a communicative multi agent problem, dynamic task management and perhaps call for robot aid based on the missions estimated cost can be investigated.

References

[1] F. H. Clarke et al. *Nonsmooth Analysis and Control Theory*. Berlin, Heidelberg: Springer-Verlag, 1998. ISBN: 0387983368.

[2] Reiya Takemura and Genya Ishigami. "Traversability-Based RRT * for Planetary Rover Path Planning in Rough Terrain with LIDAR Point Cloud Data". In: *Journal of Robotics and Mechatronics* 29 (Oct. 2017), pp. 838–846. DOI: 10.20965/jrm.2017.p0838.

[3] Marius Thoresen et al. "Path Planning for UGVs Based on Traversability Hybrid A*". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 1216–1223. DOI: 10.1109/LRA.2021.3056028.

[4] Dmitri Dolgov et al. "Practical Search Techniques in Path Planning for Autonomous Driving". In: *AAAI Workshop - Technical Report* (Jan. 2008).

[5] Astghik Hakobyan and Insoon Yang. "Distributionally Robust Risk Map for Learning-Based Motion Planning and Control: A Semidefinite Programming Approach". In: *IEEE Transactions on Robotics* 39.1 (2023), pp. 718–737. DOI: 10.1109/TRO.2022.3200156.

[6] David Fan et al. "STEP: Stochastic Traversability Evaluation and Planning for Risk-Aware Off-road Navigation". In: July 2021. DOI: 10.15607/RSS.2021.XVII.021.