CS229 Project: Building on existing Bayesian learning for Safe High Speed Planning in Partially Observable Environments

Toby John Buckley

I. Introduction

With improvements in sensing technology and computational power, robots capable of moving through an environment in real-time are becoming more and more feasible. One such example is an autonomous four-wheeled vehicle which obeys differential constraints (Fig. 1). The general methodology for a robot, given a movement task, is to sense its surroundings, compute a trajectory which will bring it closer to the goal location, and begin to move. While moving, the robot may re-sense the environment and update its algorithm to utilize this new information. An example of this is shown in fig. (2) where a car is traversing a maze-like environment. This is useful in multiple scenarios; if there are moving objects, uncertainties in the sensors, or an unknown map to traverse, the task would be near impossible to complete without feedback. Each of these cases are exacerbated by high-speeds and energetic dynamics, as the planner may not have time to react properly to dangerous behavior. As a result, safety constraints are maintained in calculations for the planner to ensure collisions do not occur.

A typical constraint is to simulate actions into the future, and confirm that at least one possible action does not result in collision. If no such actions exist, the current state is called an Inevitable Collision State (ICS) [1]. Further constraints can be derived by considering the dynamics of objects in the environment, as well as forcing the horizon to be arbitrarily large [2].

In any case, in order to perform the simulation, assumptions must be made about the unexplored state space. In the most conservative approach, the robot treats any part of the environment not already explored as an obstacle. This approach maintains zero percent probability of collision but has multiple drawbacks as well. For instance, a robot rounding a hallway corner will slow down so that it can sense the unexplored path as it's turning, resulting in more time taken. In the same scenario, a human knows from experience that it is very unlikely a hallway dead-ends and subsequently will turn the corner sharply. If a path contains many sharp

tobyb@stanford.edu

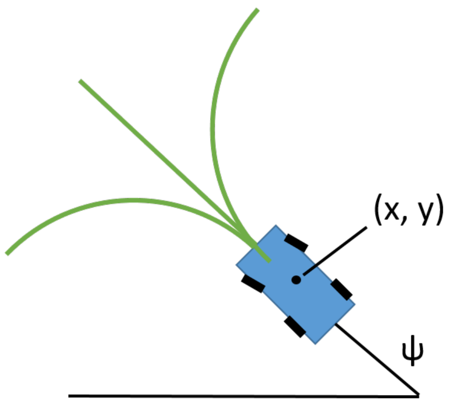


Fig. 1: A car-like vehicle. No slip is assumed, so differential constraints must be obeyed which limit the vehicle's movement abilities in the lateral direction.

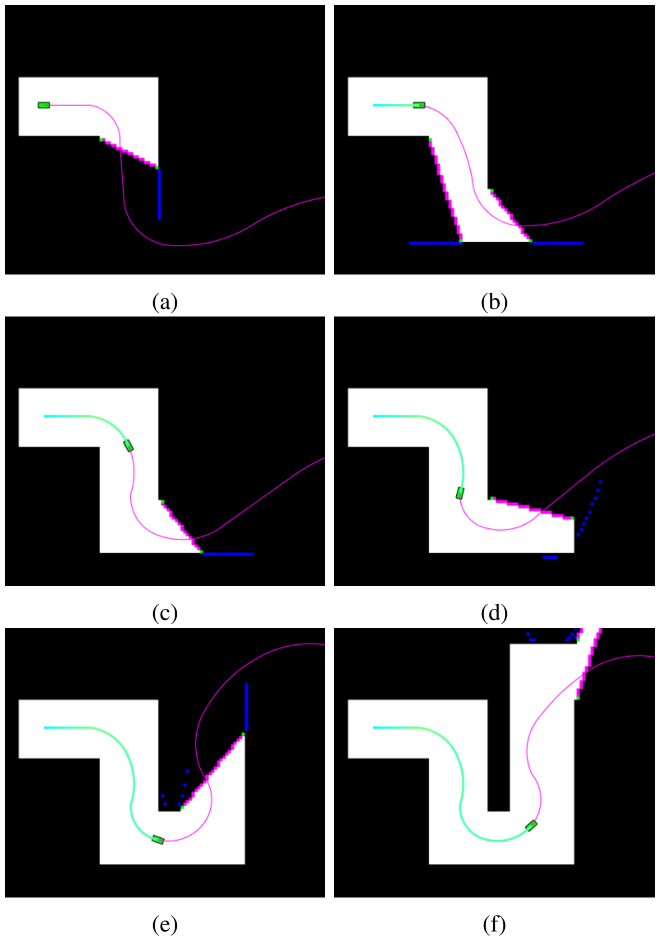


Fig. 2: The path of a car as it moves through time in an unknown environment. It's knowledge of the environment improves as it progresses. Green is the executed path, magenta is the planned path given its current knowledge, pink is the boundary into the unknown, and blue are the inferred walls.

turns the lost time can add up, resulting in much slower completion time.

There are multiple methods to speed up the robot; the first is to relax the safety constraints and approximate the probability of collision in some manner. This, combined with a penalty for collisions allows the robot to naturally gauge whether an action's risk vs reward is worthwhile. This method often requires hard-coded behavior which can be detrimental in scenarios differing from the algorithm's original intent.

Another method is to maintain safety constraints but encourage behavior that reduces the likelihood a robot finds itself in an unfavorable scenario. With the same example as previously, when taking a sharp corner at high speed a robot can swing out wide in order to increase its vision around the corner and give itself room to take evasive action if necessary. If the hallway is clear, it will continue its turn without having to accelerate back to full speed.

This paper uses the latter method as a baseline planner, and compares it to the probability based planner which utilizes Bayesian inference.

II. Literature

There are many methods to tackling the problem of high speed planning in partially observable environments, each with their own strengths and weaknesses.

First off, it can be advantageous to recast this problem as a POMDP, as many people have, and use this framework to work in the belief space instead of the state space [3], [4], [5]. With such a formulation, it's possible to manipulate the beliefs directly and take them into account when choosing a next action. A large branch of the POMDP formulation research is focused on relaxing the safety constraints outlined previously and using methods such as machine learning or sampling of the belief space to estimate the probability of collision [6].

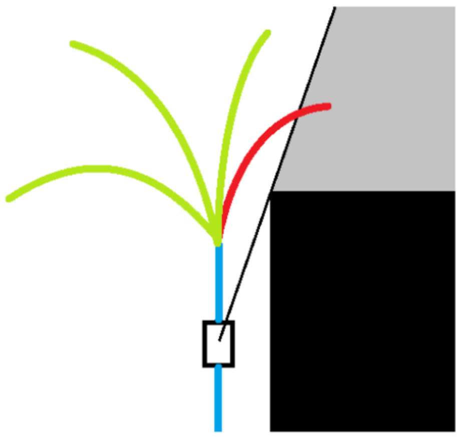


Fig. 3: Computing ICS. The blue line depicts a candidate action, with green and red showing valid and invalid emergency stopping maneuvers respectively. Even though the grey un-seen space is obstacle free, the vehicle doesn't know that from its initial position. So to be conservative, it assumes such space is an obstacle.

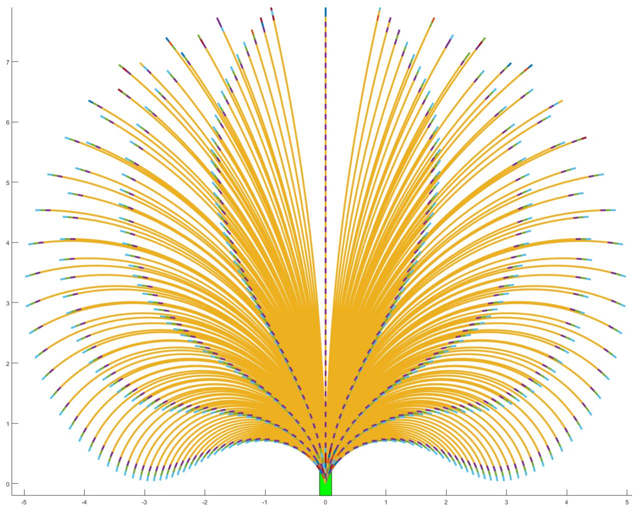


Fig. 4: All possible quarter second actions for a car with a max speed of 32 m/s.

One such method was recently proposed by Charles Richter et. al. Their planning algorithm utilizes machine learning to estimate the probability of collision for a given state [7]. The collision probability depends on calculating pre-defined features of an observation, such as minimum distance to nearest known obstacle or the final speed of an action, and comparing it against the machine-learned data. While such algorithms obtain impressive results, it ultimately only works in environments similar to those trained on. Furthermore, there is an argument to be made that environments are too high in dimensionality to be boiled down to a handful of features.

While Richter's algorithm does reduce completion time, there are areas to improve on. First, Richter et. al. assume a data set can be built by training in any environment and that any differences in output are due to their features lacking the subtlety to capture the difference. This may lead to irrational behavior in the robot if two training environments happen to share features but differ greatly in collision probability. This method is also open to incorrectly identifying a novel environment as one that has been trained in. For example, results from Richter's simulation in a hallway-forest hybrid map using a prior and training data shows that even in environments not trained in (forest), there are peaks in the data-density, implying in those time steps the planner mistook the forest for a hallway environment. This problem may be reduced by introducing more features such as largest arc-length of an obstacle projected on the horizon, or longest straight-edge of an obstacle

Second, Richter's algorithm does not take into account potential information gained from executing actions that brings the robot close to the edge of unexplored territory. In situations where data-density is low, it would be beneficial to return to areas of high data-density. If the planner was choosing between two similar actions, one of which continued in

```
D.init(K)
for $k \leftarrow 1$ to $K$ do
    randomly sample feasible configuration,
      map, and action
    calculate stopping maneuvers
    if collision free then
        $y^{(i)} \leftarrow 0$
    else
        $y^{(i)} \leftarrow 1$
    end
    $\phi \leftarrow calcFeatures(action)$
    $D(k) \leftarrow \{y^{(i)}, \phi\}$
end
```

Algorithm 1: Probability Modeling

```
$y, \phi_{Train} \leftarrow D$
while not at goal do
    Cost.init(actions)
    for $a \in actions$ do
        $\phi \leftarrow calcFeatures(a)$
        $K \leftarrow calcKernel(\phi, \phi_{Train})$
        $\alpha, \beta \leftarrow calcPsuedoPriors(a)$
        $f_c \leftarrow eq2 : f(y, \alpha, \beta, K)$
        $Cost(a) \leftarrow eq1 : (a, f_c)$
    end
    $a^* \leftarrow argmin(Cost)$
    $execute\ a^*$
end
```

Algorithm 2: Bayesian Learning

low data-density region and the other with the potential to observe a region of high data-density, the latter action should be chosen.

III. Proposed Solution

Charles Richter uses four features to predict the probability of crashing. They are as follows: 1) minimum distance to the nearest obstacle along the path, 2) average distance to an obstacle or horizon in a 60 angle in front of the robot along the action, 3) average free straight path directly in front of the robot along the action, and 4) total speed at the end of the action.

In order to test additional features and see how the results change, I chose four additional features to test; 5) ratio of sensed new cells to total cells, 6) ratio of walls to free space, 7) total turn angle, 8) number of obstacle cluster (calculated using k-means clustering).

Training data, D, is generated by randomly sampling a feasible configuration and action within a training map. An observation is made from the configuration and the features are calculated with respect to the chosen action. Next, emergency braking maneuvers are executed from the end of the action for a variety of steering angles to see if the vehicle is in an inevitable collision state (Fig. 3). If any maneuvers successfully bring the vehicle to a stop without collision, then y_i is set to 0. Otherwise $y_i = 1$. The features and ICS check results are placed in D and the process repeats. Algorithm 1 outlines this process.

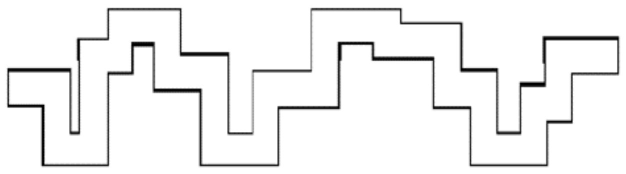
In order to choose the next action, the cost of each feasible action is calculated (1) and the minimum is chosen. Fig. (4) shows an example of each action possible for a car (sans obstacles). $J_a(a_t)$ denotes the time to execute action a_t, $h(b_t, a_t)$ denotes the heuristic cost to the goal for a_t given the current belief b_t, J_c denotes the cost of collision, and f_c denotes the posterior probability of collision.

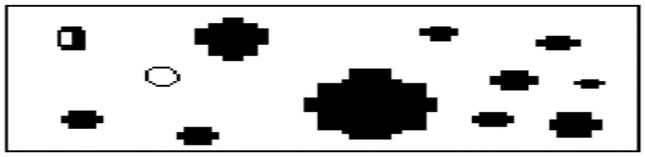
$$a_t^*(b_t) = argmin_{a_t}\big\{J_a(a_t) + h(b_t, a_t) + J_c * f_c(\phi(b_t, a_t))\big\} \tag{1}$$

Posterior probability of collision is calculated according to a non-parametric Bayesian inference model (2). This model was developed by Vega-Brown et al [8].

$$f_c(\phi) = P(y = \text{"}collision\text{"} | \phi, D) = \frac{\alpha(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i) y_i}{\alpha(\phi) + \beta(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i)} \tag{2}$$

Where $k(\phi, \phi_i)$ is the radial basis function (Gaussian kernel). The prior pseudo-counts α and β act as a form a Laplace smoothing, where the counts are a function of the features present for the given action. If action a_t results in an inevitable collision state when assuming that all unknown space is an obstacle, then α is set to a positive value. Otherwise it is zero. This ensures that when the vehicle enters


(a) maze


(b) forest

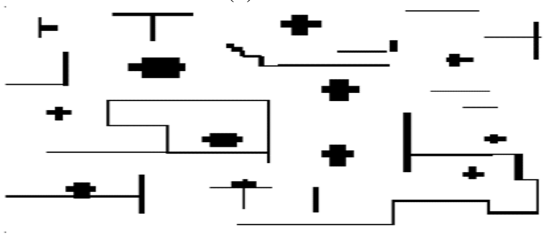
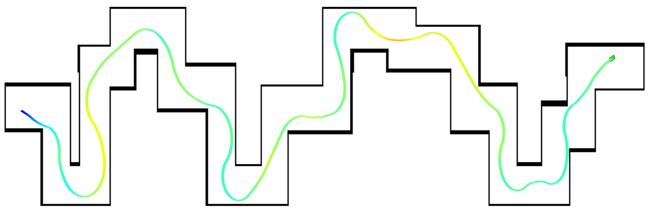
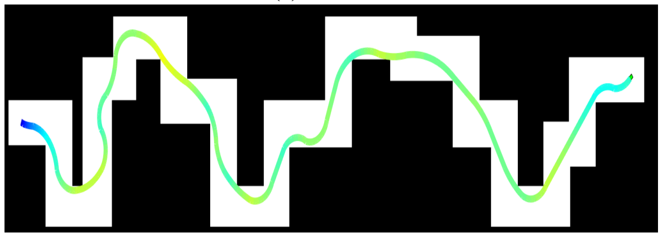

(c) hybrid

Fig. 5: Maps used to train the machine learning planner.



(a) Baseline



(b) Machine Learning

Fig. 6: Simulation results for the maze map from the two algorithms. Color denotes speed of the vehicle where dark blue is zero and red is high-speed.

a region with little to no training data, the prior distribution dominates and a similar safety constraint compared to the baseline is used to guide the vehicle through the region. Algorithm 2 outlines this process.

The result of the action cost function is that when the algorithm is very confident a collision will not occur due to high training data-density and a small number of recorded collisions, then the robot will maintain high speed while steering towards unknown regions of the map. The planner assumes the space past the boundary will be open and that no collision will occur. Conversely, if the data shows that actions with similar features crashed a majority of the time, then the cost will drastically increase, making it unlikely the planner will choose such an action, and will choose a safer alternative.

IV. Simulation/Experiments

In our simulation, the full dynamics of a car are used, complete with applied force at the front axle to act as the electric motor or braking system. The available controls are steering angle and applied force. The original planner was trained on a maze-like environment, while the extended planner was trained on all three types of maps (Fig. 5). A total of 8000 training points were generated, and a value of $J_c = 0.8$ was used for the cost of collision. If an action is an ICS according to a conservative map estimate, then α is set to 5, and 0 otherwise. And the same as Richter's paper, $\alpha + \beta$ is set to 5.

Now that the specifics of how the simulation was run are taken care of, we can examine the results. Fig. (6) shows an example of paths taken by the two algorithms for a maze-like map. Note that the baseline is unnecessarily curvy, this is from the cost function depending heavily on speed at the end of the action. As a result, when traversing an open hallway the planner goes close to the wall as this reduces distance to the goal. But once a corner is approached the baseline planner must swing overly wide in order to maintain high speed and meet the ICS constraint. Fig. 7 shows how this behavior truly reduces total time. As seen, the difference in velocity between the ML planner and the baseline is almost always positive, and in upwards of 3 m/s in magnitude. By maintaining higher speed for longer, the time to the goal is significantly reduced.

In contrast, observe how the probabilistic planner cuts the majority of the corners very closely. This is because the machine learning data tells it that each corner has high likelihood of continuing past the unknown boundary, so it associates low risk with maintaining high speed while moving through the region. Presented in table I are the time

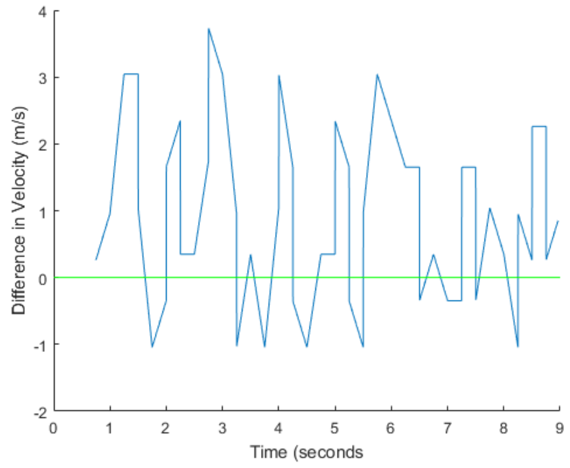


Fig. 7: Difference in velocity between the ML planner and baseline.

taken to complete three different maps. Simulations were ran with randomized starting locations within a bounding start box. The original machine learning planner experiences speedups in maps where it has some familiarity (maze & hybrid) and is slower in the unknown environment (forest). Table II show the percent reduction of the ML planners compared to the baseline. The new machine learning planner is also faster in the maze, but far slower on the other two maps. Overall the original ML planner beats the new ML planner in each category. This is interesting since the new planner has been trained on the new environments whereas the original has not. I attribute the lack of speed-up due to the fact that the feature space has increased from four to eight, but the number of training points stayed the same.

TABLE I: Time for completion of multiple maps. All results in seconds.

	Maze	Forest	Hybrid
Baseline	11.45	2.5	3.9
M.L.	9.25	2.9	3.4
M.L. new	9.65	3.15	4.25

TABLE II: Percent Reduction compared to baseline

	Maze	Forest	Hybrid
M.L.	19.21	-16.00	12.82
M.L. new	15.72	-26.00	-8.97

This shows the limitations of selecting too many features without generating enough training data.

It is also clear that machine learning is not right for all environment types; in some cases, the baseline planner is satisfactory.

V. Conclusions

I have shown two variations on a greedy path planning algorithm which reduce time to the goal by up to 19.2% compared to a baseline planner. This is because the baseline is very conservative, maintaing hard-coded safety constraints, and naive, it simply tries to maximize velocity throughout the course.

This speedup comes with considerable risk involved. Of the 50 trials ran with a collision cost of 0.8, only 21 completed. If collision cost was increased or more training data generated, the number of failed runs should decrease.

Finally, I have also shown that selecting too many features without properly generating enough training points can lead to worst behavior than a naive baseline.

References

[1] Inevitable collision states. a step towards safer robots?.

[2] A short paper about motion safety.

[3] High-speed autonomous navigation of unknown environments using learned probabilities of collision.

[4] Intention-aware online pomdp planning for autonomous driving in a crowd.

[5] Motion planning under uncertainty using iterative local optimization in belief space.

[6] Motion planning under uncertainty for robotic tasks with long time horizons.

[7] C. Richter et al. Bayesian learning for safe high-speed navigation in unknown environments. In *Proc. ISRR*, 2015.

[8] Vega-Brown et al. Nonparametric bayesian inference on multivariate exponential families.