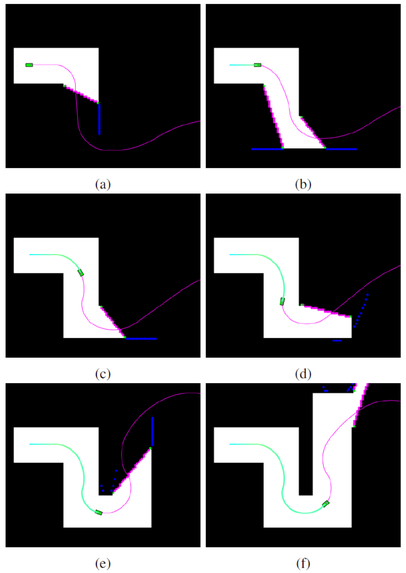
Using Bayesian Learning to Safely Plan through Partially Observable Environments

Toby John Buckley

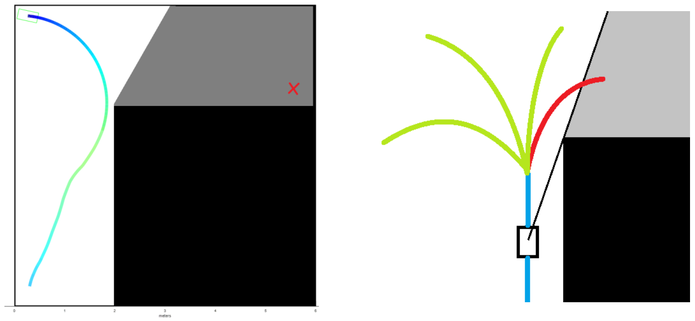
Intro to Path Planning with Uncertainty



With improvements in sensing technology and computational power, robots capable of moving through an environment in real-time are becoming more feasible. 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.

Shown left is the path of a car as it moves through time in an unknown environment. It's knowledge of the environment improves as it progresses.

The baseline uses hard safety constraints which require the vehicle to always have a feasible emergency stopping maneuver for candidate paths.

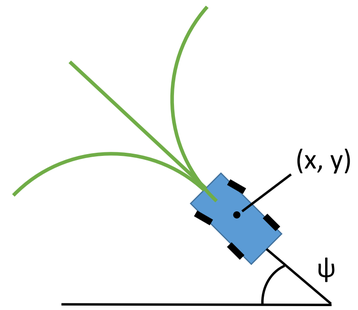


Motivating Example: Naïve baseline planner greedily moves towards goal. Such a planner will get stuck when it comes too close to the corner and be forced to execute its emergency braking maneuver.

A probabilistic planner avoids this through its learned probability of collision. It will either slow down/swing out when collision probability is high, or cut the corner sharply when collision probability is low.

Main Question: Given a car-like vehicle, how can we traverse an unknown environment quickly?

Solution: Implement Bayesian learning to calculate probability of collision for a given state.



Cost function:

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

Bayesian probability of collision:

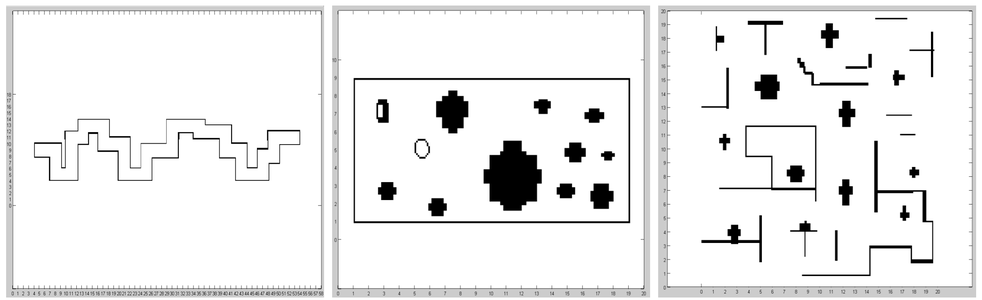
$$f_c(\phi) = P(y = \text{"collision"}|\phi, D) = \frac{\alpha(\phi)+\sum_{i=1}^{N} k(\phi,\phi_i)y_i}{\alpha(\phi)+\beta(\phi)+\sum_{i=1}^{N} k(\phi,\phi_i)} \quad (2)$$

Original Algorithm's Shortcomings: Maps are too high-dimensional to boil down to a handful of features, only trained on one type of environment.

New Changes: Double the number of features, train on more diverse maps (maze, forest, hybrid shown below).

Original Features:
A = min distance to obstacle
B = mean range to obstacle
C = min straight free path
D = total velocity

Additional New Features:
E = information gain
F = ratio of walls to free space
G = total turn angle
H = number of obstacle clusters



Algorithm 1: Modeling Probability

Used to generate training data

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

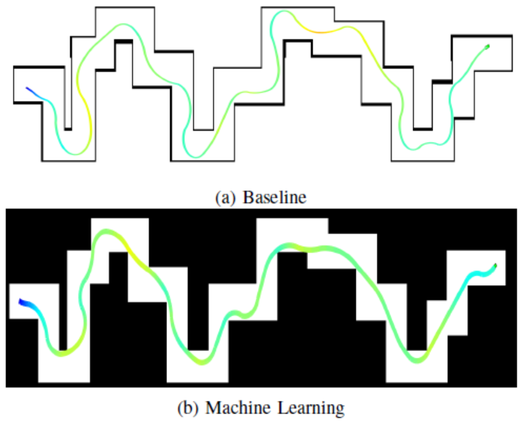
Algorithm 2: Bayesian Learning

Used to choose next action

```
y, φ_Train ← D
while not at goal do
    Cost.init(actions)
    for a ∈ actions do
        φ ← calcFeatures(a)
        K ← calcKernel(φ, φ_Train)
        α, β ← calcPsuedoPriors(a)
        f_c ← eq2 : f(y, α, β, K)
        Cost(a) ← eq1 : (a, f_c)
    end
    a* ← argmin(Cost)
    execute a*
end
```

References
1. C. Richter et al. Bayesian learning for safe high-speed navigation in unknown environments. In Proc. ISRR, 2015
2. C. Richter et al. High-speed autonomous navigation of unknown environments using learned probabilities of collision. In Proc. IEEE ICRA, 2014
3. Vega-Brown et al. Nonparametric bayesian inference on multivariate exponential families. NIPS, 2014

Simulation Results: Baseline vs. Machine Learning



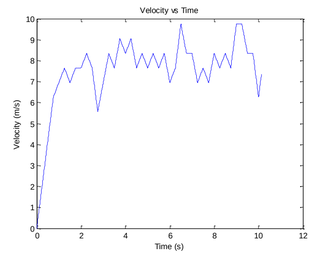
(a) Baseline

(b) Machine Learning

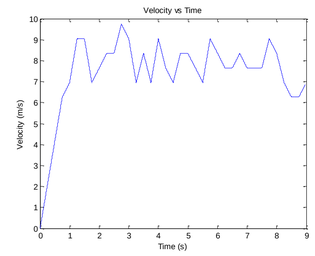
Simulation results for a maze-like map.

- Baseline is unnecessarily curvy from its cost function's dependence on maximizing velocity.
- Probabilistic planner cuts the majority of corners very closely.
- Near corners, machine learning data gives low risk of collision allowing maintenance of high speed

Velocity Profile:



Baseline



M.L.

Numerical Results:

	Time to goal (seconds)		
	Maze	Forest	Hybrid
Baseline	11.45	2.5	3.9
ML	9.25	2.9	3.4
ML new	9.65	3.15	4.25

	Percent Reduction		
	Maze	Forest	Hybrid
ML	19.21	-16.00	12.82
ML new	15.72	-26.00	-8.97

Conclusions

We have shown two variations on a greedy probabilistic path planning algorithm which reduce time to the goal significantly in some environments but with considerable risk involved.

The machine learning algorithms had 40% success rate. If collision cost was increased or more training data generated, the number of failed runs should decrease.

The results of the ML algorithm with changes show the limitations of selecting too many features without generating enough training data.

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