Towards high-speed autonomous navigation of unknown environments

Citation: Richter, Charles, and Nicholas Roy. "Towards High-Speed Autonomous Navigation of Unknown Environments." Ed. Thomas George, Achyut K. Dutta, and M. Saif Islam. N.p., 2015. 94671P. © 2015 Society of Photo-Optical Instrumentation Engineers (SPIE)

As Published: http://dx.doi.org/10.1117/12.2178668

Publisher: SPIE

Persistent URL: http://hdl.handle.net/1721.1/106286

Version: Final published version: final published article, as it appeared in a journal, conference proceedings, or other formally published context

Invited Paper

Towards High-Speed Autonomous Navigation of Unknown Environments

Charles Richter[a] and Nicholas Roy[a]

[a]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
77 Massachusetts Ave., Cambridge, MA 02139

ABSTRACT

In this paper, we summarize recent research enabling high-speed navigation in unknown environments for dynamic robots that perceive the world through onboard sensors. Many existing solutions to this problem guarantee safety by making the conservative assumption that any unknown portion of the map may contain an obstacle, and therefore constrain planned motions to lie entirely within known free space. In this work, we observe that safety constraints may significantly limit performance and that faster navigation is possible if the planner reasons about collision with unobserved obstacles probabilistically. Our overall approach is to use machine learning to approximate the expected costs of collision using the current state of the map and the planned trajectory. Our contribution is to demonstrate fast but safe planning using a learned function to predict future collision probabilities.

Keywords: Navigation, learning, unknown environments

1. INTRODUCTION

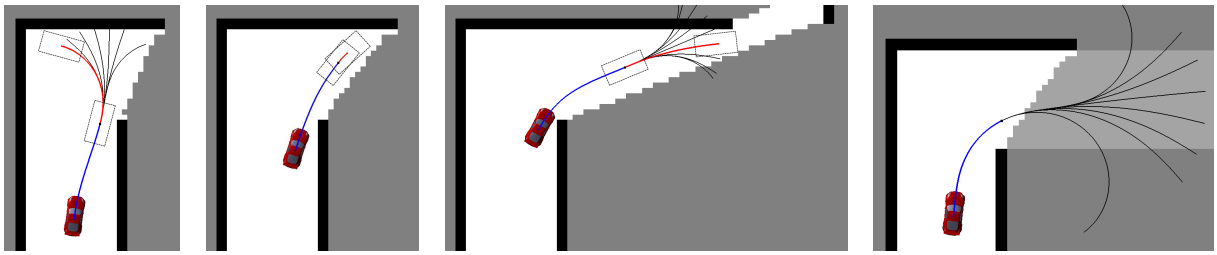
Mobile robots and autonomous vehicles offer solutions to a wide range of practical problems, including information gathering, transportation, industrial fabrication and assembly, home or workplace service, and many more. In each one of these cases, a robot must move around in a space that may be previously unmapped, or a prior map may be out of date due to rearrangement of objects. In either case, the level of prior map knowledge may not be sufficient to plan complete, high-speed, collision-free trajectories all the way to the goal. Still, we would like robots to complete tasks in a timely manner, which will require them to move much more quickly in unknown environments than they currently do. We would like to close the speed gap with human pilots and drivers and fast moving animals that can navigate at high speeds based on their learned or intuitive understanding of the way their actions and perceptual capabilities work in familiar environments.

A common method of planning in unknown environments is to use a receding-horizon approach: plan a partial trajectory given the current (partial) map knowledge, and begin to execute it while re-planning. By repeatedly re-planning, the robot can react to new map information as it is provided by the perception system. However, ensuring safety in a receding-horizon setting can be difficult, since the planner must not only perform basic collision checking of trajectories themselves, but it must also consider what actions will be available to the robot after completing the planned trajectories. Roughly speaking, the planner must not ask the robot to travel so fast that it will be unable to stop or turn to avoid an obstacle in a future planning cycle if necessary. Determining which speeds or actions are safe in a given planning scenario is a challenge for receding horizon planners that often requires conservative assumptions about the environment or a set of hand-coded rules and constraints.

For a receding-horizon planner to guarantee safety, it must plan trajectories that are known to be collision-free for all time.[1] In practice, this requirement can be satisfied by providing an infinite-horizon, collision-free escape route such as an emergency-stop trajectory associated with each planned trajectory. If the map is only partially known, the planner must also conservatively assume that every unknown cell may contain an obstacle, and therefore must plan its trajectories within the observed free space. We use a receding-horizon planner enforcing this absolute safety constraint as our "baseline" planner for comparison. This planner simply chooses the action that minimizes estimated time to reach the goal, subject to the existence of an emergency-stop trajectory in each re-planning cycle.

Further author information: (Send correspondence to Charles Richter)
Charles Richter: E-mail: car@mit.edu

Micro- and Nanotechnology Sensors, Systems, and Applications VII, edited by Thomas George, Achyut K. Dutta, M. Saif Islam, Proc. of SPIE Vol. 9467, 94671P · © 2015 SPIE
CCC code: 0277-786X/15/$18 · doi: 10.1117/12.2178668



(a) (b) (c) (d)

Figure 1: Actions chosen by the baseline planner for a robot rounding a blind corner while guaranteeing safety a–c. Observed free space is white, observed obstacles are black, and unknown regions of the map are gray. Chosen trajectories are drawn in blue, feasible emergency-stop maneuvers are drawn in red, and emergency-stop actions that collide with the environment or enter unknown space are black. The robot must slow from $4m/s$ in a to $1m/s$ in b to preserve the ability to stop before entering the unknown. Figure d shows the planner presented in section 6, equipped with a learned model of collision probability, rounding this corner much more quickly ($6m/s$) while remaining empirically safe. While this planner's emergency-stop actions enter unknown space, several lie within the free space in the hidden map (light gray).

Figures 1a–1c illustrate a sequence of plans for the baseline planner rounding a blind corner while guaranteeing safety. These figures show the planned trajectories at each time step as well as an emergency-stop action that could be executed if the corner turns out to be a dead-end and the planner is unable to re-plan. Guaranteeing the ability to stop if necessary forces the planner to slow down dramatically because the space available for an emergency-stop action becomes very limited near the map frontier (1b). Furthermore, the baseline planner does not consider information it might obtain along each action, and instead it chooses actions that make the most progress toward the goal. As the robot approaches the corner, its visibility is severely limited, preventing it from perceiving the free space around the turn.

In this paper, we summarize previous work[2] that showed how to make predictions about the effects of actions and observations, based on models learned from training data, can alleviate these limitations of the baseline planner and significantly improve navigation speed. For example, in figure 1d, our planner correctly inferred that it is safe to round a blind corner much faster than would be possible while guaranteeing safety. We also summarize more recent results that describe a novel formulation of the problem and a novel learner that provides robustness to data that is not identically distributed.

2. RELATED WORK

Motion Planning: A considerable amount of work has been devoted to planning collision-free actions for dynamic robots and establishing the conditions and constraints that need to hold in order for those actions to be safe.[3–6] These ideas have been extended to reason about potential collisions with respect to other moving agents in a dynamic environment.[7] Bekris and Kavraki developed a planner for kinodynamic navigation in unknown environments, using safety constraints, however this planner enforces the conventional rule that unobserved regions of the map are off limits and does not reason about taking actions into the unknown.[8] Schouwenaars et al., plan fixed-wing aircraft trajectories in unknown environments while guaranteeing safety through the use of circular holding patterns, however they assume a very simple sensor model, which reveals the complete environment structure within a certain radius of the vehicle.[9] Therefore, the nature of occlusions and visibility that are key components of our research are not captured. While avoiding collision with known obstacles is important, our proposed research differs from this body of work by relaxing absolute safety guarantees with respect to the unknown regions of the map, and replacing those constraints with predictions of collision probability which can be viewed as a form of data-driven constraints that enable faster navigation. Karaman and Frazzoli derive speed bounds for high-speed navigation in random environments, which is very relevant to our simulation experiments.[10]

Exploration: Much of the work on motion planning in unknown environments is focused on the exploration task, where the primary objective is to build a map, and actions may be taken to view unseen parts of the environment.[11,12] Some exploration work has focused on balancing the objectives of information gain, navigation cost and localization quality from a utility or decision-theory point of view.[13,14] Unlike the exploration or mapping tasks, our proposed objective is to reach a goal in minimum time, which gives the value of a measurement an explicit meaning in a cost-to-go sense.

POMDPs: A natural way to consider collision probabilities in a decision-theoretic way is through the formalism of the partially-observable Markov decision process (POMDP).[15] The notorious complexity of POMDPs has limited their

applicability to small planning problems, although a variety of algorithmic improvements have steadily grown the size of problems for which approximate solutions can be obtained.[16–18] The primary difference between our work and the POMDP literature is that we assume no knowledge of explicit distributions (over transitions, observations or environments), or black-box simulation capabilities to sample a model of the world at planning time. Therefore, we cannot simply apply existing POMDP techniques. Instead, we must learn approximations or functions of these missing distributions based on offline training.

POMDPs have been used for collision avoidance in the context of unmanned aircraft and air traffic control.[19,20] In this work, as is common in the MDP and POMDP literature, there is a large negative reward associated with collisions or near-collision events. This reward structure differs from our research in that we use relatively small penalties on collision to encourage the planner to aggressively drive as fast as possible by making a meaningful trade-off between risk and reward. Yet, even for small collision penalties, we are able to achieve 100% success rates.

Learning-Based Navigation: Learning has been applied to experimental autonomous vehicle navigation in a number of contexts. The DARPA Learning Applied to Ground Robots (LAGR) program was designed to stimulate research in perception-based navigation of unstructured outdoor environments using learning. The NYU entry, for instance, used a deep learning algorithm to classify traversable terrain at distances up to $100m$.[21] Neural networks and monocular depth estimation coupled with policy search have been used for high-speed collision avoidance.[22,23] Similarly, Ross et al., demonstrated learning of monocular reactive controllers from human expert demonstrations for quadrotor flight through a forest.[24] However these systems were essentially reactive in nature. They did not explicitly consider probabilistic action outcomes in an optimal or decision-theoretic way, preventing them from performing a trade-off between risk and reward.

Some recent work has attempted to apply reinforcement learning techniques to POMDP robot navigation problems. Koenig and Simmons extend Baum-Welch to tune the parameters of a robot navigation POMDP in response to interaction with the environment,[25] while Ross et al. apply Bayesian reinforcement learning to both improve the estimate of the model as well as maximize long-term reward.[26] These methods may both work for extremely small problems of a specific form, however it seems infeasible to apply them to the POMDP of interest, where the uncertainty lies in the thousands of variables describing probabilities of environment structures for which we have no meaningful prior, and which would not be useful parameters to transfer across environments.

3. PROBABILISTIC RECEDING HORIZON PLANNING

In previous work,[2] we showed how motion planning in partially observed environments could incorporate uncertainty in action execution that resulted from incomplete maps of the environment. Specifically, when planning a high-speed trajectory into an unknown region of the environment, the success or failure of that trajectory, and hence its true cost, depends on the occupancy of the unobserved region. Therefore, it is impossible to evaluate true trajectory costs at runtime.

Rather than minimizing the true cost of an action $J(a)$, we instead choose actions that minimize cost in expectation:

$$a^* = \underset{a \in A}{\operatorname{argmin}} \; E_{map}[J(a)] \tag{1}$$

where a is a trajectory from the current location of the vehicle all the way to the goal, and $E_{map}[J(a)]^*$ is the expected cost of the action with respect to the map. It is unlikely that we could learn to predict the probability of collision all the way to the goal, but there is some horizon over which we may be able to learn an accurate predictor. We use a receding horizon formulation, where we compute the expected cost up to a finite planning horizon, and then use a heuristic to estimate the path cost from that point to the goal:

$$a^* = \underset{a \in A}{\operatorname{argmin}} \; E_{map}[J(a)] + h(x_a) \tag{2}$$

This function consists of two components: the expected cost of the action to the planning horizon, and the heuristic estimate of cost-to-go, $h(x_a)$, from the state at the end of the action. The set A includes a subset of all possible trajectories from the current state to the planning horizon, and is intended to span the vehicle's maneuvering capabilities.

Although we are operating in an unknown environment, we assume a SLAM process is inferring a map from sensor data. Most SLAM processes do not provide a single map estimate but provide a probability distribution over the map, with

*To avoid confusion: The map distribution is over maps, not the *maximum a posteriori* distribution that is often referred to as p_{MAP} in machine learning.

greater uncertainty in the parts of the map with less sensor data. Because the map is not known exactly, we cannot compute the cost of a trajectory $J(a)$ exactly but need to compute the expected cost with respect to the map distribution $p(map)$. There are several ways to model $E_{map}[J(a)]$, depending on our assumptions. We could make the naïve assumption that unobserved regions of the map are free and traversable, or the conservative assumption that unobserved regions are occupied. Both of these assumptions are likely to be incorrect. Unfortunately, computing the expected cost exactly is equally difficult, because the map distributions are typically very high dimensional, and hence difficult to integrate, and also usually contain independence assumptions that allow efficient inference of the lower moments of the distribution but do not capture the entire distribution accurately enough to use for expected cost calculations.

An alternative approach is possible, since computing the expectation with respect to the full map distribution is actually unnecessary. In our planning problem, costs are due to either successfully completing the trajectory or experiencing a collision. The expectation can then be computed over the total probability of collision along the trajectory, $p_{total}(a)$:

$$E_{map}[J(a)] = p_{total}(a) \cdot J_{collision} + (1 - p_{total}(a)) \cdot J_{free}(a) \tag{3}$$

The expected value of a trajectory in equation (2) is a weighted sum of the free-space cost of a trajectory $J_{free}(a)$ and the penalty we assign to a collision $J_{collision}$. The weighting applied to these two terms is a function of the total probability that the vehicle will collide along the trajectory.

For the minimum-time problem, J_{free} is simply the time duration of the action. To compute p_{total} for any given trajectory, we use the quantity $p_{collision}(x_t, m_t)$, which represents the probability that the vehicle will enter an inevitable collision state (ICS) between the current time t and the subsequent planning iteration at $t + \Delta t$. This quantity is a function of the vehicle state, x_t, and the state of the vehicle's map of the environment, m_t, at that time. We consider the vehicle to be in an ICS if, on its next planning iteration, all of the possible trajectories the vehicle could execute collide with known walls or obstacles in the environment.

If we know $p_{collision}(x_t, m_t)$ at every discrete time increment t along a trajectory, then we can estimate the probability that the vehicle will enter an ICS along that trajectory:

$$p_{total} = 1 - \prod_{t=0}^{T} 1 - p_{collision}(x_t, m_t) \tag{4}$$

where T is the trajectory duration. Intuitively, the probability of collision approaches one as its duration increases for any given value of $p_{collision} > 0$. Therefore, it is important to compare trajectories of equal duration or otherwise correct for differing lengths. Equation (4) is an instance of the geometric distribution, used to estimate the cumulative effect of discrete risks represented by the discrete hazard function $p_{collision}$. This type of model representing risk over the lifetime of an agent is common in reliability engineering.

4. LEARNING PROBABILITIES OF COLLISION

Even with the expectation computed over the simpler binary probability distribution, it is still typically not possible to evaluate $p_{collision}$ from a map. The probabilities of environmental structure inside the unknown regions are typically very poor predictors of collision probability. Nevertheless, there are features of the environment and vehicle state that suggest how safe or how dangerous any given trajectory is likely to be. For instance, the vehicle is likely to be safe traveling down a straight, empty hallway with high visibility in its direction of travel. However, when it nears the turn at the end of the hallway, it will be unable to take a measurement of the environment ahead until it rounds the corner. In this scenario, the vehicle may risk a high probability of collision if it rounds the corner too quickly.

We estimate these probabilities of collision using a learned function, which maps features $\phi(x_t, m_t)$ of the state and environment map to a scalar probability of collision:

$$f_{collision}(\phi(x_t, m_t)) \approx p_{collision}(x_t, m_t) \tag{5}$$

We learn the function $f_{collision}$ from simulation data by repeatedly simulating the car starting from random points in space and allowing it to navigate toward the goal by greedily selecting the action that descends the heuristic function most rapidly. We record the features encountered by the vehicle at every time step, and associate an outcome (`collision`

Figure 2: 1:8 scale RC car used in experiments, with Hokuyo laser range-finder, Microstrain IMU and computer.

or `non-collision`) with each feature record. If the vehicle succeeds in reaching the goal, then the feature history associated with that trial is labeled `non-collision`. If a collision occurs during simulation, then the feature values preceding the collision are labeled as `collision`.

We use multivariate logistic regression to fit a sigmoid (logistic) function to approximate the binomial distribution of outcomes throughout the feature space. The probability of collision for a point in feature space is given by the logistic function of features ϕ_i and associated weights w_i:

$$f_{collision}(\phi) = \frac{1}{1 + \exp(-w_0 - \sum_{i=1}^{N} w_i \phi_i)} \tag{6}$$

Logistic regression is used to find weights w_i such that $f_{collision}$ most closely fits the observed ratio of `collision` to `non-collision` throughout feature space.

Five features were used to predict collision probabilities, denoted $\phi_1 \ldots \phi_5$. Let $d_{occ.}$ and $d_{unk.}$ denote the distances to the nearest obstacle and frontier, and let $\mathbf{r_{occ.}}$ and $\mathbf{r_{unk.}}$ denote the vectors between the vehicle position and the nearest obstacle and frontier, respectively. Let $\mathbf{v}$ denote the vehicle's velocity vector. The features are:

- Distance to obstacle: $\phi_1 = d_{occ.}$
- Distance to frontier: $\phi_2 = d_{unk.}$
- Velocity toward obstacle: $\phi_3 = \mathbf{v} \cdot \mathbf{r_{occ.}} / \|\mathbf{r_{occ.}}\|$
- Velocity toward frontier: $\phi_4 = \mathbf{v} \cdot \mathbf{r_{unk.}} / \|\mathbf{r_{unk.}}\|$
- Scalar vehicle speed: $\phi_5 = \|\mathbf{v}\|$

These features were selected to enable basic reasoning about the vehicle's motion with respect to the geometry of the known environment structure as well as the unknown regions. While many more features could be added to this list, feature selection lies outside the scope of this work.

5. EXPERIMENTAL RESULTS

We performed experiments on an autonomous 1:8 scale radio-controlled car equipped with an onboard computer with a dual-core Intel Core i7 processor, a Microstrain inertial measurement unit and Hokuyo planar laser range-finder. State estimation was performed onboard the vehicle using an IMU-driven extended Kalman filter with the filter formulation presented in,[27] using a scan-matching algorithm on the laser data to provide relative odometry measurements. The laser data were also used to populate a grid cell map of the environment in real time, and this map was used for planning. Figure 2 shows the car used in the experiments, with the laser, IMU and computer onboard.

Figure 3 shows the partial map produced by the car on an experimental run through an indoor space, along with the set of available actions from the states depicted. In this environment, the car reached a velocities up to $4m/s$. A video of experimental results is available at: `http://groups.csail.mit.edu/rrg/nav_learned_prob_collision`.

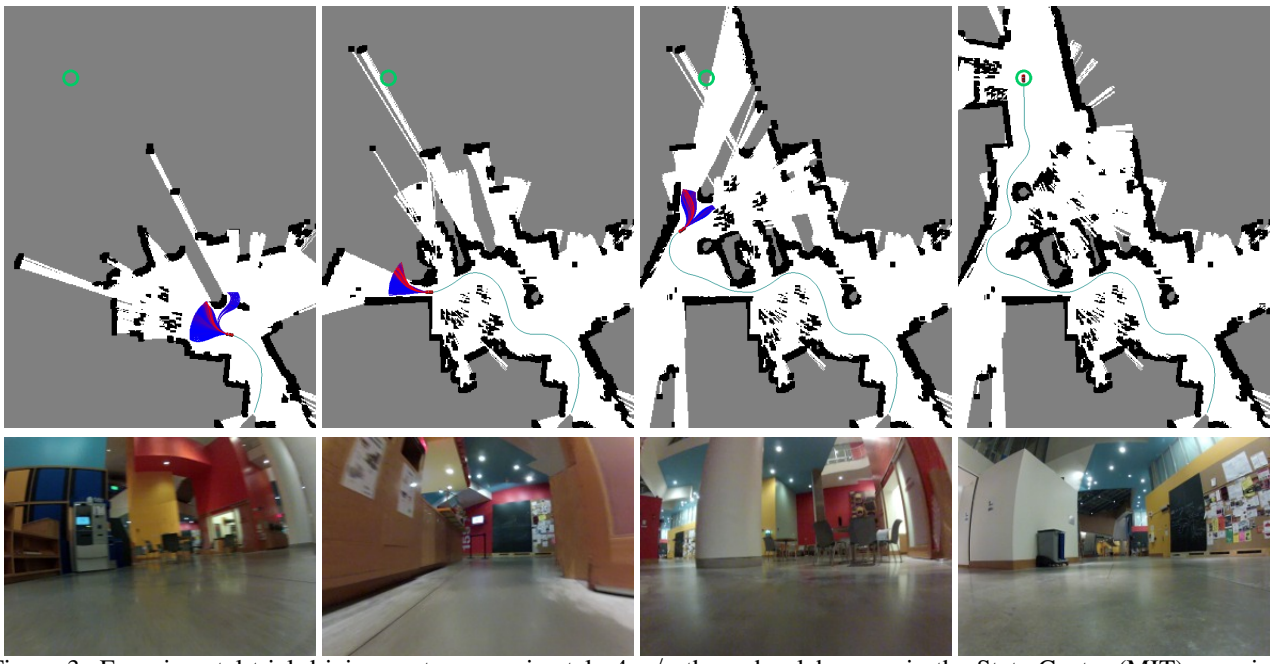


Figure 3: Experimental trial driving up to approximately $4m/s$ through a lab space in the Stata Center (MIT) covering $60m$ of unknown, unstructured space. The goal (green circle) was not visible until nearly the end of the experimental run.

6. BELIEF SPACE PLANNING

These previous results demonstrated a possible route to fast, aggressive motion in an unknown environment, but also had some fundamental limitations, in terms of the formalism and the learner. A more natural way to incorporate probabilistic models of collision probability in a decision-theoretic framework is through the partially-observable Markov decision process (POMDP) formalism. In the remainder of this section, we develop the POMDP formulation of our problem. We then discuss the central difficulty of this POMDP, which is that we do not have an accurate distribution over the real-world environments in which we hope to navigate. In subsequent sections, we will describe the approximations we make to the POMDP, and the functions that we can learn from data, which help us to overcome this fundamental difficulty.

A POMDP agent maintains a belief over its state $b(s) = P(q, m)$. We use $b_t(s_t)$ to denote the belief at time t over the state the state at time t. and m_t to denote the belief over maps at t, which is simply the map portion of the belief. Since q is fully observable in our problem, the belief really represents exact knowledge of the robot configuration combined with the agent's belief over which of the exponentially many possible maps it is in. In principle, if we knew the optimal cost-to-go $V^*(s)$ for every possible state s, we could compute the optimal action to take given belief b_t, using the following equation:

$$a_t^*(b_t) = \underset{a_t}{\text{argmin}} \left\{ \sum_{s_t} b_t(s_t) R(s_t, a_t) + \sum_{s_t} b_t(s_t) \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \sum_{o_{t+1}} P(o_{t+1}|s_{t+1}, a_t) V^*(s_{t+1}) \right\} \tag{7}$$

In the second summation over s_t, equation (7) is effectively taking an expectation over maps where the probability applied to each possible map is contained within $b_t(s_t)$. The summations over observations o_{t+1} and states s_{t+1} perform a state estimation update to compute the distribution over future states given the current belief b_t and an action a_t. Therefore, we can rewrite (7) as:

$$a_t^*(b_t) = \underset{a_t}{\text{argmin}} \left\{ \sum_{s_t} b_t(s_t) R(s_t, a_t) + \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) V^*(s_{t+1}) \right\} \tag{8}$$

If our belief, b_t, assigns uniform probability to all possible maps, which would be a very unrealistic distribution over maps, then the state estimation update serves to eliminate those maps that are not consistent with the current observation and raise the uniform probability of the remaining possible maps. On the other hand, if b_t assigns high probability to a small

set of realistic maps containing common structures such as hallways, rooms, doors, etc., and low probability to the many unrealistic (though technically possible) maps, then this state estimation update would help us infer useful information about regions of the map that have not been directly observed yet. For instance, it might infer that the straight parallel walls of a hallway are likely to continue to be straight parallel walls out into the unknown regions of the map. All of this prior knowledge about which environments are likely, or typical, enters the problem through the agent's initial belief b_0, which we must somehow provide.

7. MISSING MAP INFORMATION: THE CENTRAL CHALLENGE

Unfortunately, learning or modeling an accurate distribution over the space of all real-world environments the robot may encounter would be extremely difficult. This difficulty results from the high dimensionality ($\approx 10^6$) of building-sized occupancy grid maps, the strong assumption of independence between map cells, and the richness of man-made and natural spaces which resist a more compact parameterization. Without significant modeling effort, or restriction of the problem domain to specific environments, the best prior we can reasonably provide is to say that every map is equally likely. Of course, this uniform prior is completely unhelpful for planning, and prevents the agent from exploiting intuitive knowledge of "typical" environments to navigate faster. Our solution must compensate for this missing knowledge.

Our most recent proposed approach is to learn a specific function that helps the agent drive at high speed *as if* it had an accurate prior over environments enabling it to make reasonable inferences about the unobserved environment. This function is used to approximate a portion of the expected future cost that is computed on the right-hand end of equation (7), eliminating the need for expensive summations over states and observations, which would be intractable. More importantly, our approach eliminates the need for an explicit prior over environments. The relevant information that would be contained within a prior over environments is represented instead implicitly through training data for our learning algorithm. Our learned function, $f_c(\phi)$, approximates the probability that a collision will occur with some obstacle in the future, given some features ϕ of a planning scenario.

8. BAYESIAN NON-PARAMETRIC LEARNING

We use a non-parametric Bayesian inference model developed by Vega-Brown et al., which generalizes local kernel estimation to the context of Bayesian inference for exponential family distributions.[28] The Bayesian nature of this inference model will become important when we specify prior knowledge over collision probabilities to keep the robot safe when it encounters a novel environment for which it has no relevant training data. We consider collision to be a Bernoulli-distributed random event with beta-distributed parameter $\theta \sim Beta(\alpha, \beta)$, where α and β are prior pseudo-counts of collision and non-collision events, respectively. Using the inference model from Vega-Brown et al., we can efficiently compute the posterior probability of collision given a query point ϕ and the training data set $\mathcal{D}$:

$$f_c(\phi) = P(y = \text{“collision”}|\phi, \mathcal{D}) = \frac{\alpha(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i) y_i}{\alpha(\phi) + \beta(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i)}, \tag{9}$$

where $k(\phi, \phi_i)$ is a kernel function reflecting proximity in feature space between our query point ϕ and each other data point in $\mathcal{D}$. We write prior pseudo-counts as functions $\alpha(\phi)$ and $\beta(\phi)$, since they may vary across the feature space. We use the kernel sum, $N_{\text{eff.}} = \sum_{i=1}^{N} k(\phi, \phi_i)$, i.e., the effective number of training data points near ϕ, as a measure of data density. The prior also contributes $N_{\text{pr.}}$ pseudo data points to each prediction, where $N_{\text{pr.}} = \alpha(\phi) + \beta(\phi)$. The ratio $N_{\text{eff.}}/(N_{\text{eff.}} + N_{\text{pr.}})$ determines the relative influence of the training data and the prior in each prediction. We choose $N_{\text{pr.}}$ to be a constant around 1, and typically when planning in an environment type for which we have training data, $N_{\text{eff.}} > 100$.

We assume that for a given vehicle or dynamical system, there exists some true underlying probability of collision that is independent of the map, given features ϕ. In other words, $P(\text{“collision”}|\phi, m) = P(\text{“collision”}|\phi)$. Under this assumption, we can build a data set by training in any environments we wish, and the data will be equally valid in other environments that populate the same region of feature space. If two data sets gathered from two different environment types do not share the same output distribution where they overlap in feature space, we assume that our features are simply not rich enough to capture the difference. In order to apply our learned model to new environments, we need only for the new environments to lie within the region of feature space where we have training data. On the other hand, our learning algorithm should react appropriately if we have no relevant training data.

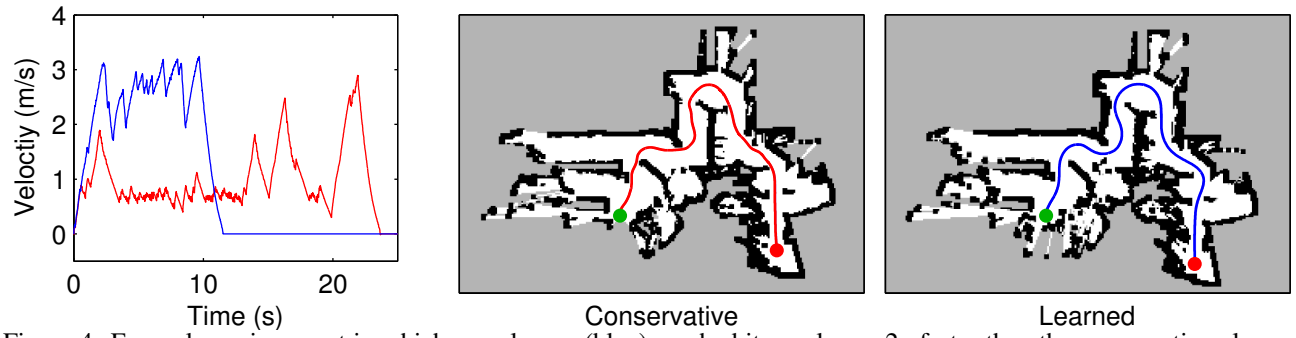


Figure 4: Example environment in which our planner (blue) reached its goal over 2x faster than the conservative planner (red). The left graph shows the velocity profiles of the conservative and learned planners, and the center and right images show the trajectory of each planner. The start and goal are marked with green and red dots.

Machine learning algorithms are designed to make good predictions for query points near their training data. For some algorithms, predictions that extrapolate beyond the region of training data may be arbitrarily bad. For navigation tasks, we want our planner to recognize when it has left the region of feature space for which it has training data, and automatically revert to safe behaviors. For instance, if the agent moves from a well-known environment type into one it has never seen before, we want it to stay safe. Fortunately, the inference model of equation (9) enables this capability.

The Bayesian nature of our inference model enables us to inject prior knowledge that will keep the planner safe when there is little relevant data. If we query a feature point in a region of high data density ($N_{\text{eff.}} \gg N_{\text{pr.}}$), $f_c(\phi)$ will tend to a local weighted average of neighbors and the prior will have little effect. However, if we query a point far from the training data ($N_{\text{eff.}} \ll N_{\text{pr.}}$), the prior will dominate the prediction. By specifying priors $\alpha(\phi)$ and $\beta(\phi)$ that are functions of our features, we can endow the planner with all of our own (perhaps conservative) domain knowledge about which regions of feature space are safe and which are dangerous. In this work, we have designed our prior functions $\alpha(\phi)$ and $\beta(\phi)$ such that $P(\text{“collision”}) = \alpha(\phi)/(\alpha(\phi) + \beta(\phi)) = 0$ if there exists enough free space for the robot to come safely to a stop from its current velocity, and $P(\text{“collision”}) = \alpha(\phi)/(\alpha(\phi) + \beta(\phi)) > 0$ otherwise. This information is computed using two of the features specified above, which give the velocity and the free distance ahead of the robot. Therefore, as $N_{\text{eff.}}$ drops to zero, this stopping-distance safety constraint becomes active, seamlessly and automatically turning our planner into one with essentially the same characteristics as the baseline planner.

9. EXPERIMENTAL RESULTS USING BAYESIAN LEARNING

We conducted experimental tests on our RC car, using our POMDP-derived formulation and Bayesian learning algorithm, to show that our planner can indeed navigate faster in certain environments than the baseline planner. For the experiment shown here, we chose a narrow path within a lab space with several 90° turns and many obstacles. Figure 4 shows the trajectories and velocity profiles of both planners. The baseline planner has difficulty navigating quickly in this environment because free space is occluded from the sensor view by obstacles. Since the baseline planner must enforce the existence of emergency-stopping trajectories lying within the observed free space, it is forced to move very slowly. In the velocity profile of the conservative planner, we can see that it frequently applies the brakes to slow to about $1m/s$, whereas our planner correctly uses its training data from the simulated hallway environment to predict that it is in fact safe to travel faster around the turns and therefore maintains a higher average speed. The conservative planner took $23.7s$ to reach the goal in this case, whereas our planner took $11.5s$, representing a factor of two improvement.

ACKNOWLEDGMENTS

This work was supported by the Army Research Labs under the Micro Autonomous Systems Technology CTA, and their support is gratefully acknowledged.

REFERENCES

[1] Fraichard, T., “A short paper about motion safety,” in [*Proc. ICRA*], (2007).

[2] Richter, C., Ware, J., and Roy, N., "High-speed autonomous navigation of unknown environments using learned probabilities of collision," in [*Proc. ICRA*], (2014).
[3] Asama, H. and Fraichard, T., "Inevitable collision states–a step towards safer robots," *Advanced Robotics* **18**, 1001–1024 (2004).
[4] Fox, D., Burgard, W., and Thrun, S., "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine* **4**(1), 23–33 (1997).
[5] Simmons, R., "The curvature-velocity method for local obstacle avoidance," in [*Proc. ICRA*], (1996).
[6] Frazzoli, E., Dahleh, M., and Feron, E., "Real-time motion planning for agile autonomous vehicles," *Journal of Guidance, Control and Dynamics* **25**(1) (2002).
[7] Fiorini, P. and Shiller, Z., "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research* **17**(7), 760–772 (1998).
[8] Bekris, K. E. and Kavraki, L. E., "Greedy but safe replanning under kinodynamic constraints," in [*Proc. ICRA*], (2007).
[9] Schouwenaars, T., How, J., and Feron, E., "Receding horizon path planning with implicit safety guarantees," in [*Proc. ACC*], (2004).
[10] Karaman, S. and Frazzoli, E., "High-speed flight in an ergodic forest," in [*Proc. ICRA*], (2012).
[11] Yamauchi, B., "A frontier-based approach for autonomous exploration," in [*Proc. CIRA*], (1997).
[12] Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M., and Schimdt, T., "Map learning and high-speed navigation in RHINO," *AI-based Mobile Robots: Case studies of successful robot systems. MIT Press, Cambridge, MA* (1998).
[13] Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F., "An experiment in integrated exploration," in [*Proc. IROS*], (2002).
[14] Stachniss, C., Grisetti, G., and Burgard, W., "Information gain-based exploration using rao-blackwellized particle filters.," in [*Robotics: Science and Systems*], (2005).
[15] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R., "Planning and acting in partially observable stochastic domains," *Artificial intelligence* **101**(1), 99–134 (1998).
[16] Pineau, J., Gordon, G., and Thrun, S., "Point-based value iteration: An anytime algorithm for pomdps," in [*Proc. IJCAI*], (2003).
[17] Kurniawati, H., Hsu, D., and Lee, W. S., "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in [*Robotics: Science & Systems*], (2008).
[18] Silver, D. and Veness, J., "Monte-carlo planning in large POMDPs," in [*Proc. NIPS*], (2010).
[19] Temizer, S., Kochenderfer, M. J., Kaelbling, L. P., Lozano-Pérez, T., and Kuchar, J. K., "Collision avoidance for unmanned aircraft using Markov decision processes," in [*Proc. GNC*], (2010).
[20] Bai, H., Hsu, D., Kochenderfer, M., and Lee, W. S., "Unmanned aircraft collision avoidance using continuous-state pomdps," in [*Robotics: Science and Systems*], (2011).
[21] Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Muller, U., and LeCun, Y., "Learning long-range vision for autonomous off-road driving," *Journal of Field Robotics* **26**(2), 120–144 (2009).
[22] Muller, U., Ben, J., Cosatto, E., Flepp, B., and LeCun, Y., "Off-road obstacle avoidance through end-to-end learning," in [*Proc. NIPS*], (2005).
[23] Michels, J., Saxena, A., and Ng, A., "High speed obstacle avoidance using monocular vision and reinforcement learning," in [*Proc. ICML*], (2005).
[24] Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., and Hebert, M., "Learning monocular reactive uav control in cluttered natural environments," in [*Proc. ICRA*], (2013).
[25] Koenig, S. and Simmons, R., "Unsupervised learning of probabilistic models for robot navigation," in [*Proc. ICRA*], (1996).
[26] Ross, S., Chaib-draa, B., and Pineau, J., "Bayesian reinforcement learning in continuous pomdps with application to robot navigation," in [*Proc. ICRA*], (2008).
[27] Bry, A., Bachrach, A., and Roy, N., "State estimation for aggressive flight in gps-denied environments using onboard sensing," in [*Robotics and Automation (ICRA), 2012 IEEE International Conference on*], 1–8, IEEE (2012).
[28] Vega-Brown, W. R., Doniec, M., and Roy, N., "Nonparametric bayesian inference on multivariate exponential families," in [*Proc. NIPS*], (2014).