In Proceedings of the International Symposium on Robotics Research (ISRR 2015).

Bayesian Learning for Safe High-Speed Navigation in Unknown Environments

Charles Richter, William Vega-Brown, and Nicholas Roy

Abstract In this work, we develop a planner for high-speed navigation in unknown environments, for example reaching a goal in an unknown building in minimum time, or flying as fast as possible through a forest. This planning task is challenging because the distribution over possible maps, which is needed to estimate the feasibility and cost of trajectories, is unknown and extremely hard to model for real-world environments. At the same time, the worst-case assumptions that a receding-horizon planner might make about the unknown regions of the map may be overly conservative, and may limit performance. Therefore, robots must make accurate predictions about what will happen beyond the map frontiers to navigate as fast as possible. To reason about uncertainty in the map, we model this problem as a POMDP and discuss why it is so difficult given that we have no accurate probability distribution over real-world environments. We then present a novel method of predicting collision probabilities based on training data, which compensates for the missing environment distribution and provides an approximate solution to the POMDP. Extending our previous work, the principal result of this paper is that by using a Bayesian non-parametric learning algorithm that encodes formal safety constraints as a prior over collision probabilities, our planner seamlessly reverts to safe behavior when it encounters a novel environment for which it has no relevant training data. This strategy generalizes our method across *all* environment types, including those for which we have training data as well as those for which we do not. In familiar environment types with dense training data, we show an 80% speed improvement compared to a planner that is constrained to guarantee safety. In experiments, our planner has reached over 8 m/s in unknown cluttered indoor spaces. Video of our experimental demonstration is available at `http://groups.csail.mit.edu/rrg/bayesian_learning_high_speed_nav`.

1 Introduction

A common planning strategy in unknown environments is the 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 perceived. However, avoiding collision in a receding-

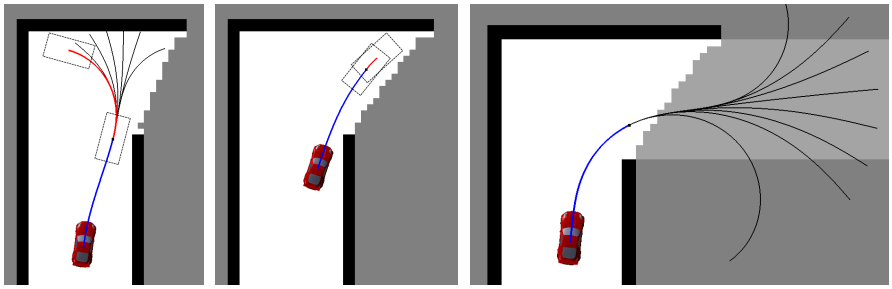
Charles Richter, William Vega-Brown, and Nicholas Roy
Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, e-mail: `car,wrvb,nickroy@mit.edu`

horizon setting can be difficult, since the planner must not only ensure that its chosen actions are collision-free within the planning horizon, but it must also consider what actions will be feasible for the robot after the planned trajectory has been completed. To guarantee safety, a receding-horizon planner must plan trajectories that are known to be collision-free for all time [10]. This constraint is often satisfied through hand-coded rules or contingency plans, such as ensuring the existence of a collision-free emergency-stop maneuver or cyclic holding pattern [2, 23]. If the map is partially known, a safe planner must conservatively assume that every unknown cell may contain an obstacle, and therefore must confine its trajectories to lie within the observed free space. Figures 1a–1b illustrate this scenario for a robot approaching a blind corner while guaranteeing safety. As the robot approaches the corner, it slows dramatically to preserve its ability to complete a feasible emergency-stop maneuver before entering unknown space. We use this "baseline" safe planner for comparison in this paper.

Safety constraints imply an assumption that driving into unknown regions of the map is always dangerous. However, our central claim in this line of research is that this assumption may be overly conservative. In many cases, the agent can safely drive at high speeds into the unknown if it is equipped with an accurate model of collision probability based on previous experience in similar situations [20]. For instance, many buildings have straight hallways with 90° turns and intersections. If a robot observes a hallway with a turn, and has trained in similar environments, it should be able to reason with high certainty that the hallway will continue, and that there will be free space around the turn. Figure 1c shows our proposed solution planning a high-speed turn around a blind corner, violating the safety constraint. Emergency-stop trajectories from the chosen action (illustrated in black) all enter unknown space. Yet, this type of maneuver is often empirically safe for hallway environments and our solution correctly infers that it is not excessively risky.

One way to estimate the likelihood of collision in unknown space might be to infer the probability that certain unknown map regions are occupied, using sensor measurements and an accurate prior distribution over maps. Unfortunately, modeling an accurate distribution over real-world environments would be extremely difficult due to the very high dimensionality 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 compact parameterization. Without significant modeling effort, or restriction of the problem domain to specific environments, the best prior knowledge we can reasonably provide is to say that every map is equally likely. Of course, this prior is very inaccurate and completely unhelpful for planning, and prevents the agent from exploiting any intuitive knowledge of "typical" environments. To compensate for this missing knowledge, we adopt a machine learning approach to predict collision probability from training data, which implicitly captures the relevant traits of our training environments rather than modeling the map distribution explicitly. This approach leads to a much lower-dimensional model that is significantly easier to learn and efficient enough to run online.

In the next section, we develop the POMDP formulation of our problem, which lays the foundation for decision theoretic planning. We then derive our approxima-



(a) (b) (c)

Fig. 1: Actions chosen for a robot approaching a blind corner while guaranteeing safety (a)–(b). Chosen trajectories are drawn in blue, and feasible emergency-stop maneuvers are drawn in red. Emergency-stop actions that would cause the robot to collide with obstacles or unknown cells are drawn in black. The safety constraint requires the robot to slow from 4 m/s in (a) to 1 m/s in (b) to preserve the ability to stop within known free space. In (c), our solution plans a 6 m/s turn, which commits the robot to entering the unknown, but the planner's experience predicts a low risk of collision. The light gray region shows the hallway of the true hidden map.

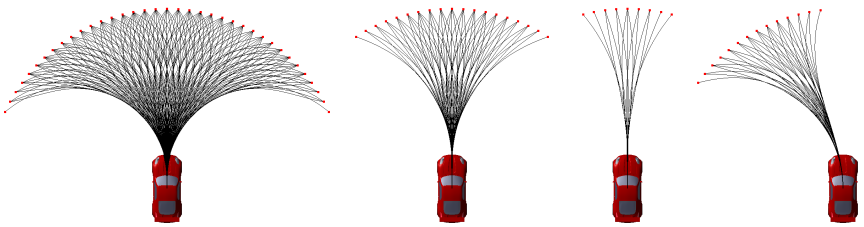
tions to the POMDP to make the problem tractable, which give rise to our learned model of collision probability. Then, we discuss how we learn that model and how we encode safety constraints as a prior over collision probability to help our planner remain safe in novel environments for which it has no relevant training data. Finally, we will present simulation and experimental results demonstrating 100% *empirical* safety while navigating significantly faster than the baseline planner that relies on formal constraints to remain safe.

2 POMDP Planning

We wish to control a dynamic vehicle through an unknown environment to a goal position in minimum expected time, where the expectation is taken with respect to the (unknown) distribution of maps. While solving this problem exactly is completely intractable, the POMDP formalism is useful for defining the planning task and motivating our approximations. The following POMDP tuple, (S, A, T, R, O, Ω), applies to an RC car equipped with a planar LIDAR being used for this research:

States S: $\{Q \times M\}$. Q is the set of vehicle configurations: $q = [x, y, \psi, k, v]$, representing position, heading, curvature (steering angle) and forward speed, respectively. M is the set of n-cell occupancy maps: $m = [m_0, m_1, \ldots, m_n] \in \{0,1\}^n$, where m_i represents the i^{th} cell in the map. For a given problem instance, the true underlying map, m, is fixed, while the configuration, q, changes at each time step as the robot moves. We assume that q is fully observable, while m is partially observable since only a subset of the map can be observed from a given location. We also assume that n is fixed and known.

Actions A: A is a pre-computed discrete action library spanning the vehicle's maneuvering capabilities. Several examples are illustrated in Figure 2. All actions reach a planning horizon of 2 meters, but differ in their time duration as a function of their speeds. Due to space constraints, we refer the reader to Howard et al. for a discussion of discrete action sets for dynamic vehicles [13].



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

Fig. 2: Examples of pre-computed action sets from 1 m/s (a), 4 m/s (b), and 8 m/s (c) with zero initial curvature, and from 4 m/s with non-zero initial curvature (d).

We define several deterministic functions related to transition dynamics. The collision function $C(s,a) : S \times A \mapsto \{0,1\}$ indicates whether taking action a from state s would result in a collision. The state-transition function $F(s,a) : S \times A \mapsto S$ returns the state reached by taking action a from state s according to the dynamics. In F, the map portion of the state remains fixed at its true value. If a collision would occur along trajectory a (i.e., if $C(s,a) = 1$), then $F(s,a)$ clamps the configuration to its last feasible value along a and sets the velocity to zero. The function $ICS(s) : S \mapsto \{0,1\}$ indicates whether state s is an inevitable collision state [3], i.e., if there exists no infinite-horizon sequence of actions $\langle a_0, a_1, \dots \rangle$ from s that will avoid collision with the environment.

Conditional Transition Probabilities T: We assume deterministic vehicle dynamics and a fixed map, $p(s_{t+1}|s_t,a_t) = 1$ for $s_{t+1} = F(s_t,a_t)$ and 0 otherwise. While actions have different time durations, we use the subscript $t+1$ to indicate the discrete "time" after completing action a_t.

Observations Ω: Each observation consists of a perfect measurement of q_t and the partial map of cells visible to the robot from s_t. This partial map consists of occupied cells at locations $r_{i,xy}$ corresponding to each LIDAR range measurement r_i, and unoccupied cells along the ray from q_t to each $r_{i,xy}$.

Conditional Observation Probabilities O: We assume a noiseless sensor, so $p(o_{t+1}|s_{t+1},a_t) = 1$ for the partial map corresponding to the map geometry visible from state s_{t+1}, along with a perfect measurement of q_{t+1}, and 0 otherwise.

Cost Function R: We use a minimum-time cost function and denote the time duration of action a as $J_a(a)$. Let S_G denote the set of goal states. $R(s,a) = 0$ for $s \in S_G$. For $s \notin S_G$, $R(s,a) = J_a(a)$ if $C(s,a) = 0$ and adds an additional collision penalty, J_c, if the action results in a collision: $R(s,a) = J_a(a) + J_c$ if $C(s,a) = 1$.

2.1 Missing prior distribution over environments

A POMDP agent maintains a belief over its state $b(s_t) = P(q_t, m)$, and has a state estimator, which computes a posterior belief that results from taking an action and then receiving an observation, given a current belief: $b(s_{t+1}) = P(s_{t+1}|b_t, a_t, o_{t+1})$. If the agent's current belief, b_t, assigns uniform probability to all possible maps with independence between map cells (a very unrealistic distribution), then an observation and subsequent inference over the map distribution simply eliminates those maps that are not consistent with the observation and raises the uniform probability of the remaining possible maps. If, on the other hand, the prior belief correctly assigns high probability to a small set of realistic maps with common structures such as hallways, rooms, doors, etc., and low probability to the unrealistic maps, then this belief update may help to infer useful information about unobserved regions of the map. For instance, it might infer that the straight parallel walls of a hallway are likely to continue out into the unobserved regions of the map. All of this prior knowledge about the distribution of environments enters the problem through the agent's initial belief b_0, which we must somehow provide.

Unfortunately, as we noted in Section 1, modeling the distribution over real-world environments would be extremely difficult. We have little choice but to initialize the robot believing that all maps are equally likely and that map cells are independent from each other. To compensate for this missing environment distribution, our approach is to learn a much more specific distribution representing the probability of collision associated with a given planning scenario. This learned function enables the agent to drive at high speed *as if* it had an accurate prior over environments enabling reasonable inferences about the unknown. In the following sections, we will derive our learned model from the POMDP and describe how we can efficiently train and use this model in place of an accurate prior over environments.

2.2 Approximations to the POMDP

Let $V_\pi(s) = \sum_{t=0}^{\infty} R(s_t, \pi(s_t))$ be the infinite-horizon cost associated with some policy π mapping states s_t to actions a_t. The optimal cost-to-go, $V^*(s)$ could then, in principle, be computed recursively using the Bellman equation. Having computed $V^*(s)$ for all states, we could then recover the optimal action for a given belief[1]:

$$a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}} \Bigg\{ \sum_{s_t} b(s_t) R(s_t, a_t) + \sum_{s_t} b(s_t) \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t) \sum_{o_{t+1}} P(o_{t+1}|s_{t+1}, a_t) V^*(s_{t+1}) \Bigg\} \tag{1}$$

The summations over s_t and o_{t+1} perform a state-estimation update, resulting in a posterior distribution over future states s_{t+1} from all possible current states in our current belief b_t, given our choice of action a_t. We can rewrite (1) as:

[1] Since we use a discrete set of actions, there is a finite number of configurations that can be reached from an initial configuration. Therefore, we can sum over future states rather than integrating.

$$a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}} \left\{ \sum_{s_t} b(s_t) R(s_t, a_t) + \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) V^*(s_{t+1}) \right\} \tag{2}$$

Since our cost function $R(s,a)$ applies separate penalties for time and collision, we can split $V^*(s)$ into two terms, $V^*(s) = V_T^*(s) + V_C^*(s)$, where T and C refer to "time" and "collision". $V_T^*(s)$, gives the optimal time-to-go from s, regardless of whether a collision occurs between s and the goal. We assume that if a collision occurs, the robot can recover and proceed to the goal. Furthermore, we assume that the optimal trajectory from s to the goal will avoid collision if possible. But there will be some states s for which collision is inevitable ($ICS(s) = 1$) since the robot's abilities to brake or turn are limited to finite, realistic values. Since we assume that collisions only occur from inevitable collision states, we can rewrite the total cost-to-go as: $V^*(s) = V_T^*(s) + J_c \cdot ICS(s)$. Substituting this expression, we can rewrite (2) as:

$$\begin{aligned} a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}} \Bigg\{ & \sum_{s_t} b(s_t) R(s_t, a_t) + \\ & \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) V_T^*(s_{t+1}) + J_c \cdot \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) ICS(s_{t+1}) \Bigg\} \end{aligned} \tag{3}$$

The first term in equation (3) is the expected immediate cost of the current action, a_t. We assume no collisions occur along a_t since the robot performs collision checking with respect to observed obstacles, and it nearly always perceives obstacles in the immediate vicinity. This term simply reduces to $J_a(a_t)$, the time duration of a_t.

The second and third terms of (3) are expected values with respect to the possible future states, given a_t and b_t. However, as we have observed in Section 2.1, our initial uniform belief over maps means that b_t will be very unhelpful (and indeed misleading) if we use it to take an expectation over future states, since the true distribution over maps is surely far from uniform. The lack of meaningful prior knowledge over the distribution of environments, combined with the extremely large number of possible future states s_{t+1}, means that we must approximate these terms.

For the expected time-to-go, we use a simple heuristic function:

$$h(b_t, a_t) \approx \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) V_T^*(s_{t+1}), \tag{4}$$

which performs a 2D grid search using Dijkstra's algorithm (ignoring dynamics) from the *end* of action a_t to the goal, respecting the observed obstacles in b_t, assuming unknown space is traversable and that the current speed is maintained. The use of a lower-dimensional search to provide global guidance to a local planner is closely related to graduated-density methods [12]. While other heuristics could be devised, we instead focus our efforts in this paper on modeling the collision probability.

The third term in equation (3), $J_c \cdot \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) ICS(s_{t+1})$, is the expected future collision penalty given our current belief b_t and action a_t. As we noted in Section 2.1, the fundamental problem is that our belief b_t does not accurately capture which map structures are likely to be encountered in the unknown portions of the

environment. Correctly predicting whether a hallway will continue around a blind corner, for instance, is impossible based on the belief alone. We instead turn to machine learning to approximate this important quantity from training data:

$$f_c(\phi(b_t, a_t)) \approx \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t) ICS(s_{t+1}) \tag{5}$$

The model, $f_c(\phi(b_t, a_t))$, approximates the probability that a collision will occur in the future if we execute action a_t given belief b_t. It is computed from some features $\phi(b_t, a_t)$ that summarize the relevant information contained in b_t and a_t, for example vehicle speed, distance to observed obstacles along a_t, etc. With the approximations of all three terms in equation (3), we arrive at our receding-horizon control law:

$$a_t^*(b_t) = \underset{a_t}{\operatorname{argmin}} \left\{ J_a(a_t) + h(b_t, a_t) + J_c \cdot f_c(\phi(b_t, a_t)) \right\} \tag{6}$$

At each time step, we select a_t^* minimizing (6) given the current belief b_t. We begin to execute a_t^* while incorporating new sensor data and re-planning. Next, we describe how we collect data and build a model to predict collision probabilities.

3 Predicting Future Collisions

In this section, we focus on learning to predict the probability of collision associated with a given planning scenario, represented as a point in feature space, Φ. We assume that for a given vehicle or dynamical system, there exists some true underlying probability of collision that is independent of the map and robot configuration, given features, ϕ. In other words, $P(\text{“collision”}|\phi, q, m) = P(\text{“collision”}|\phi)$. 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 regions of feature space. If two data sets gathered from two different environments 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.

This assumption also implies that if an environment is fundamentally different from our training environments with respect to collision probabilities, it will populate a different region of feature space. If the robot encounters an environment for which it has no relevant training data nearby in feature space, it should infer that this environment is unfamiliar and react appropriately. In these cases, we require that our learning algorithm somehow provide a safe prediction of collision probability rather than naïvely extrapolating the data from other environments. Our features must therefore be sufficiently descriptive to predict collisions as well as differentiate between qualitatively different environment types.

Quantifying a planning scenario using a set of descriptive functions of belief-action pairs currently relies on the domain knowledge of the feature designer. For this paper, our features are four hand-coded functions: (a) minimum distance to the nearest known obstacle along the action; (b) mean range to obstacle or frontier in the 60° cone ahead of the robot, averaged over several points along the action;

(c) length of the straight free path directly ahead of the robot, averaged over several points along the action; and (d) speed at the end of the action. While these features work adequately, our method is extensible to arbitrary numbers of continuous- and discrete-valued features from a variety of different sources, including features that operate on a history of past measurements. We expect that additional features will enable more intelligent and subtle navigation behaviors.

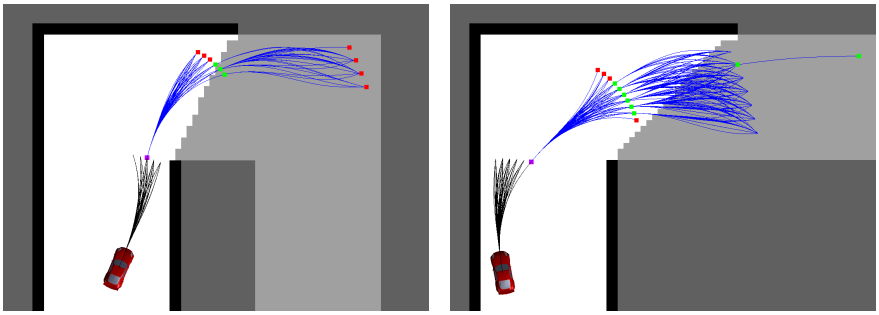
3.1 Training procedure

To predict collision probabilities, we collect training data $D = \{(\phi_1, y_1), \ldots, (\phi_N, y_N)\}$. Labels y_i are binary indicators ("collision", "non-collision") associated with belief-action pairs, represented as points ϕ_i in feature space. To efficiently collect a large data set, we use a simulator capable of generating realistic LIDAR scans and vehicle motions within arbitrary maps. The training procedure aims to generate scenarios that accurately emulate beliefs the planner may have at runtime, and accurately represent the risk of actions given those beliefs. While at runtime, the planner will use a map built from a history of scans, we make the simplifying assumption that a single measurement taken from configuration q_t is enough to build a realistic map of the area around q_t, similar to one the planner might actually observe. With this assumption, we can generate training data based on individual sampled robot configurations, rather than sampling extended state-measurement histories, which not only results in more efficient training, but also eliminates the need for any sort of planner or random process to sample state histories.

We generate each data point by randomly sampling a feasible configuration, q_t, within a training map, and simulating the sensor from q_t to generate a map estimate, and hence a belief b_t. We then randomly select one of the actions, a_t, that is feasible given the known obstacles in b_t. Recall from equation (5) that our learned function models the probability that a collision will occur somewhere *after* completing the immediate chosen action a_t, i.e., the probability that state s_{t+1} (with configuration q_{t+1}) is an inevitable collision state. Therefore, to label this belief-action pair, we run a resolution-complete training planner from configuration q_{t+1} at the *end* of a_t. If there exists any feasible trajectory starting from q_{t+1} that avoids collision with the true hidden map (to some horizon), then $y_{\text{new}} = 0$, otherwise $y_{\text{new}} = 1$. Finally, we compute features $\phi_{\text{new}}(b_t, a_t)$ of this belief-action pair and insert $(\phi_{\text{new}}, y_{\text{new}})$ into D.

We use a horizon of three actions (6 m) for our training planner because if a collision is inevitable for our dynamics model, it will nearly always occur within this horizon. We have explored other settings for this horizon and found the results to be comparable, although if the horizon is too short, some inevitable collision states will be mis-labeled as "non-collision".

Figure 3 illustrates "collision" and "non-collision" training instances. In 3a, the hidden map includes a sharp hairpin turn, which is infeasible for our dynamics model, given that the car begins toward the inside of the turn at a relatively high speed ($\approx$ 8 m/s). Therefore, the training planner fails to find a trajectory to the desired horizon and each partial path results in a dead-end (red dot) because the robot is moving too fast to complete the hairpin turn. On the other hand, the hidden map



(a) (b)

Fig. 3: Examples of "collision" (a) and "non-collision" (b) training events. One of the immediate actions (black) is chosen for labeling. The training planner determines whether the end of this action (purple dot) is an inevitable collision state with respect to the hidden map (shown in light gray). Feasible partial paths are shown in blue. Nodes successfully expanded by the training planner are green, and nodes for which no collision-free outgoing action exists are red. In (a), all partial paths dead-end (red nodes) before reaching the desired three-action horizon because the vehicle speed is too great to complete the turn given curvature and cuvature rate limits. In (b), the training planner successfully finds a sequence of three actions.

in 3b has a straight hallway rather than a sharp turn, and the training planner succeeds in finding a feasible trajectory to the three-action horizon, aided by the car's initial configuration toward the outside of the turn. The empirical distribution of "collision" and "non-collision" labels collected from these sampled scenarios therefore implicitly captures the relevant environmental characteristics of the true hidden map distribution, and the way they interact with our dynamics model, in a much more efficient way than modeling the complex, high-dimensional map distribution explicitly. Our training procedure captures sensible patterns, for instance that it is safe to drive at high speeds in wide open areas and long straight hallways, but that slower speeds are safer when approaching a wall or navigating dense clutter.

3.2 Learning algorithm

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 [29]. The Bayesian nature of this model will enable us to provide prior knowledge to make safe (though perhaps conservative) predictions in environments where we have no training data. We model collision as a Bernoulli-distributed random event with beta-distributed parameter $\theta \sim \text{Beta}(\alpha, \beta)$, 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 data D:

$$f_c(\phi) = P(y = \text{“collision”}|\phi, D) = \frac{\alpha(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i) y_i}{\alpha(\phi) + \beta(\phi) + \sum_{i=1}^{N} k(\phi, \phi_i)}, \tag{7}$$

where $k(\phi, \phi_i)$ is a kernel function measuring proximity in feature space between our query, ϕ, and each ϕ_i in D. We use a polynomial approximation to a Gaussian kernel with finite support. We write prior pseudo-counts as functions $\alpha(\phi)$ and $\beta(\phi)$, since they may vary across the feature space. We use the effective number of nearby training data points, $N_{\text{eff.}} = \sum_{i=1}^{N} k(\phi, \phi_i)$, as a measure of data density. The prior 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. For data sets with 5×10^4 points in total, $N_{\text{eff.}}$ tends to be 10^2 or greater when the testing environment is similar to the training map, and $N_{\text{eff.}} \ll 1$ when the testing environment is fundamentally different (i.e., office building vs. natural forest). For this paper, we used $N_{\text{pr.}} = 5$, and results are insensitive to the exact value of $N_{\text{pr.}}$ as long as $N_{\text{eff.}} \gg N_{\text{pr.}}$ or $N_{\text{eff.}} \ll N_{\text{pr.}}$.

Machine learning algorithms should make good predictions for query points near their training data in input space. However, predictions that extrapolate beyond the domain of the training data may be arbitrarily bad. For navigation tasks, we want our planner to recognize when it has no relevant training data, and automatically revert to safe behaviors rather than making reckless, uninformed predictions. For instance, if the agent moves from a well-known building into the outdoors, where it has not trained, we still want the learning algorithm to guide it away from danger.

Fortunately, the inference model of equation (7) enables this capability through the prior functions $\alpha(\phi)$ and $\beta(\phi)$. 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. By specifying priors $\alpha(\phi)$ and $\beta(\phi)$ that are functions of our features, we can endow the planner with domain knowledge and formal rules about which regions of feature space are safe and dangerous. In this paper, 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. Computing this prior uses the information in features (c) and (d) described in Section 3. Therefore, as $N_{\text{eff.}}$ drops to zero (and for sufficiently large values of J_c), the learning algorithm activates a conventional stopping distance constraint, seamlessly turning our planner into one with essentially the same characteristics as the baseline planner we compare against.

Another natural choice of learning algorithm in this domain would be a Gaussian process (GP). However, classification with a GP requires approximate inference techniques that would likely be too slow to run online, whereas the inference model in equation (7) is an approximate model that allows efficient, analytical inference.

4 Results

We have obtained simulation results from randomly generated maps as well as experimental results on a real RC car navigating as fast as 8 m/s in laboratory, office and large open atrium environments at MIT. Our simulation and experimental results both show that replacing safety constraints with a learned model of collision probability can result in faster navigation without sacrificing empirical safety. The results also show that when the planner departs a region of feature space for which it has training data, our prior keeps the robot safe. Finally, our experiments demonstrate that even within a single real-world environment, it is easy to encounter some regions of feature space for which we have training data and others for which we do not, thereby justifying our Bayesian approach and use of prior knowledge.

4.1 Simulation Results

In this section, we present simulation results from hallway and forest environments that were randomly sampled from environment-generating distributions [21]. We use a Markov chain to sample hallways with a width of 2.5 m and a turn frequency of 0.4, and a Poisson process to sample 2D forests with an average obstacle rate of 0.05 trees $\cdot$ m^{-2} and tree radius of 1 m. Figure 5 shows a randomly sampled hybrid environment composed of both hallway and forest segments. To measure the benefit of our learned model, we compare against the baseline planner, illustrated in Figures 1a-1b, that enforces a safety constraint. This planner navigates as fast as is possible given that constraint, and represents the planner we would use if we did not have a learned model of collision probability.

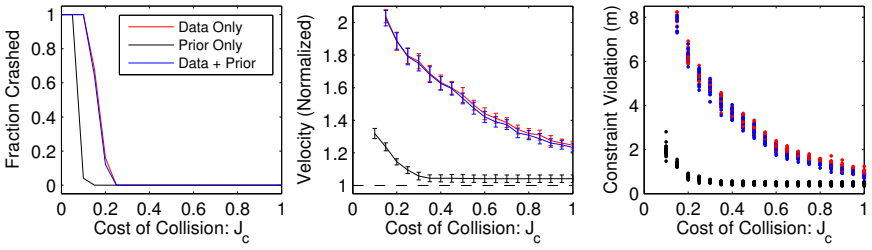


Fig. 4: Simulation results from 1600 simulations in 25 random hallway maps using our learned model with dense data + prior (blue), data alone without a prior (red), and prior alone without data (black). Velocities for each trial are normalized by the speed of the baseline planner (described in Section 1) in the same map. The right plot shows the average length by which a stopping-distance constraint was violated.

Figure 4 shows the performance of the planner using different data/prior configurations (data + prior, data only, and prior only), for different J_c values, in 25

randomly sampled hallway environments. The training data set used for these trials was collected in a separate hallway environment drawn from the same distribution. For low values of J_c all three planners crash in every trial, but become safer as J_c is increased. Above $J_c = 0.25$, all planners succeed in reaching the goal without collision 100% of the time. At $J_c = 0.25$, our solution navigates approximately 80% faster than baseline using data + prior, or data alone. In these trials, the prior contributes very little since the planner has dense training data from the same environment type. The right plot illustrates average violation of a stopping-distance constraint. For $J_c = 0.25$, the planners using training data violate the stopping-distance constraint by 5.75 m on average, indicating the degree to which this constraint is overly conservative, given our collision probability model. The planner using the prior alone chooses not to violate the stopping distance constraint by nearly as much, and essentially converges to the baseline performance as J_c is increased.

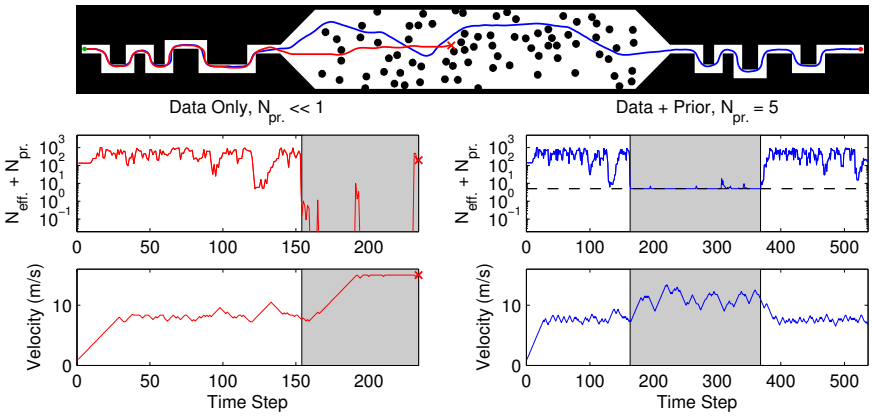


Fig. 5: Simulations from a hybrid hallway-forest map with $J_c = 0.5$. One planner uses hallway data alone with no prior (red), and another planner uses hallway data combined with the prior (blue). Without a safe prior, the planner's data density drops to zero and the robot recklessly accelerates to full speed, crashing in the forest (red '$\times$'). However, when guided by the prior while in the forest, the planner safely reaches the goal. The forest regions on the four graphs are shaded in gray.

While results with dense training data make very little use of the prior, Figure 5 illustrates the important role of a prior when transitioning from a familiar environment type (hallway) to an unfamiliar one (forest) where no data are available. For these simulation experiments, the planners had access to a training data set from a hallway environment, but not from a forest environment. If the planner has no data *and no prior*, the effective data density drops essentially to zero and it is unable to

distinguish between safe and risky behaviors[2]. Therefore the planner accelerates to full speed resulting in a crash (red '×'). However, using our Bayesian approach, the effective number of data points drops only as low as $N_{\text{pr.}} = 5$ when the agent enters the forest and the planner is safely guided by the information in the prior. In 25 trials of random hallway-forest maps, 100% succeeded using the prior, while only 12% succeeded without the prior.

4.2 Experimental Results

We conducted experiments on an RC car using a training dataset generated in simulation on a hallway-type environment. We performed state estimation by fusing a planar LIDAR and an IMU in an extended Kalman filter (EKF) [7]. We use the LIDAR to provide relative pose updates to the EKF using a scan-matching algorithm [4]. We use a Hokuyo UTM-30LX LIDAR, a Microstrain 3DM-GX3-25 IMU and an Intel dual-core i7 computer with 16GB RAM. Video of our experimental demonstrations, at speeds up to 8.2 m/s is available at: `http://groups.csail.mit.edu/rrg/bayesian_learning_high_speed_nav`.

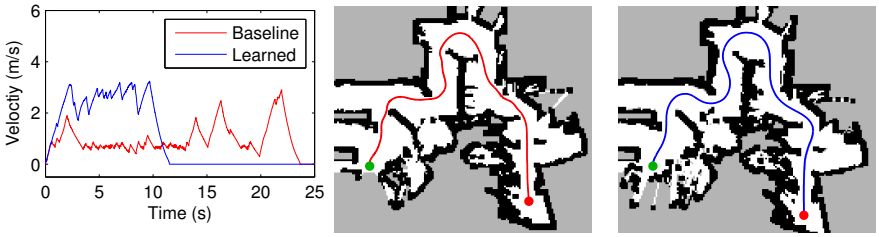


Fig. 6: Experiment in which our planner (blue) reached its goal over 2x faster than baseline (red). Velocity profiles and trajectories for each planner are shown.

We conducted tests to show that our planner can indeed navigate faster in certain real environments than the baseline planner. For the experiment shown here, we chose a narrow path within a lab space with several sharp turns and many obstacles. Figure 6 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, the baseline planner frequently applies the brakes to slow to about 1 m/s , whereas our planner uses its training data from the simulated hallway environment to predict that it is safe to travel up to about 3 m/s around most corners and therefore maintains a higher aver-

[2] We implement the no-prior case by setting prior values of α and β each to 0.0005 to ensure the solution is computable in regions of feature space with no data at all. The default prediction with no data is therefore $P(\text{“collision”}) = \alpha/(\alpha+\beta) = 0.5$, rather than undefined if $\alpha = \beta = 0$.

age speed. The baseline planner took 23.7 s to reach the goal in this case, whereas our planner took 11.5 s, representing a factor of two improvement.

We also conducted experimental trials to show that in real-world environments, the planner can safely navigate in unfamiliar regions where it has no relevant training data. Figure 7 shows an experimental trial in the Stata Center (MIT), which is composed of straight hallways and larger unstructured regions. For this experiment, training data were again provided from a simulated hallway environment, which resembles the straight hallway segments, but looks very different from the open, unstructured regions. The inset graph shows the effective data density. In the hallways, the planner uses $\approx 10^2$ effective data points per prediction. However, in the open unstructured regions, $N_{\text{eff.}}$ drops to zero, and the only information available to the planner is contributed by the prior. These open unstructured regions are marked with purple and blue diamonds. In this experiment, the weight of the prior was $N_{\text{pr.}} = 5$.

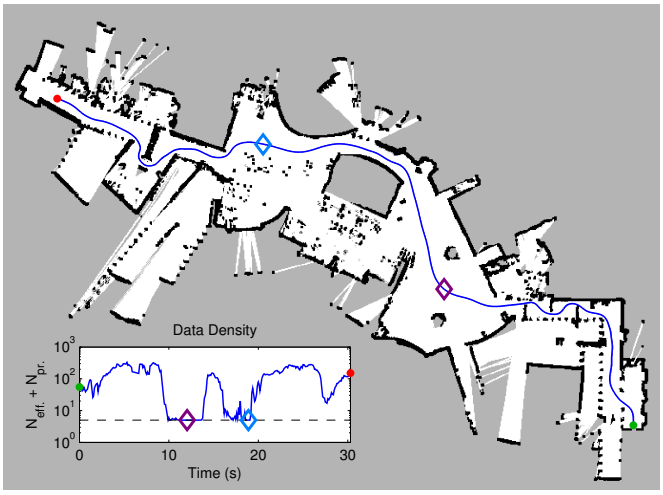


Fig. 7: Trajectory through the Stata Center (MIT), traversing hallway and large unstructured regions. Our planner was trained in a hallway environment, but not in a large unstructured environment. $N_{\text{eff.}}$ drops to zero in the open unstructured regions (purple and blue diamonds) and it must rely on the prior to navigate safely. The top speed in this environment was 8.2 m/s in a separate successful trial.

5 Related Work

A large body of work has established techniques for safe planning in static and dynamic environments [3, 8, 9, 25]. Bekris and Kavraki have shown kinodynamic navigation in unknown environments using safety constraints, without considering actions into the unknown [6]. Several examples use circling loiter maneuvers through observed free space to guarantee safety [23, 2]. Our method differs from this body

of work by relaxing absolute safety constraints and replacing them with predictions of collision probability, which can be viewed as a form of data-driven constraints. Althoff et al. propose a collision probability concept similar to $f_c(\phi(b_t, a_t))$ for dynamic environments, but they assume a known distribution over the future behavior of each moving agent, which we do not have in the case of unknown maps [1].

In the exploration literature, the primary objective is to build a map, and actions may be taken to view unseen parts of the environment [30, 28]. Some exploration work has balanced the objectives of information gain, navigation cost and localization quality from a utility or decision-theory point of view [16, 26]. Unlike exploration, our objective is to reach a goal in minimum time, which places emphasis on collision probability and high-speed dynamics, rather than map information.

A natural way to reason about planning in an unknown map is through the POMDP formalism [14]. Despite notorious complexity, various strategies have grown the size of (approximately) solvable POMDPs [19, 15, 24]. However, we cannot simply apply POMDP techniques since we assume no explicit knowledge of the environment distribution, or black-box simulation capabilities to sample the world at planning time. Instead, we must learn a function of this missing distribution offline. POMDPs have been used for aircraft collision avoidance [27, 5], where a very large negative reward discourages collisions at nearly any cost. In contrast, we use small collision penalties to drive aggressively, trading off risk and reward.

Learning has been applied to autonomous vehicle navigation in several contexts. One example from the DARPA LAGR program used deep learning to classify traversable terrain up to 100 m away [11]. Neural networks and monocular depth estimation coupled with policy search [18, 17], as well as human-pilot demonstrations [22], have been used to learn high-speed reactive controllers to avoid obstacles, however these systems map sensory input directly to actions. They are not decision-theoretic planners, and do not trade off meaningfully between risk and reward.

6 Conclusion

We have shown that by using a Bayesian learning algorithm, with safety constraints encoded as a prior over collision probabilities, our planner can detect when it lacks the appropriate training data for its environment and seamlessly revert to safe behaviors. Our strategy offers a simple and elegant probabilistic method of merging the performance benefits of training experience with the the reassurance of safety constraints when faced with an unfamiliar environment. One of the main limitations of this work is the difficulty of hand-coding feature functions that describe complex planning scenarios. We plan to address this limitation by applying recent feature-learning techniques to automatically generate feature functions from data, extending our methods to handle different scenarios, map representations and sensor types.

References

1. D. Althoff et al. Safety assessment of robot trajectories for navigation in uncertain and dynamic environments. *Autonomous Robots*, 32(3):285-302, 2012.

2. S. Arora et al. A principled approach to enable safe and high performance maneuvers for autonomous rotorcraft. *American Helicopter Society 70th Annual Forum*, 2014.
3. H. Asama and T. Fraichard. Inevitable collision states - a step towards safer robots. *Advanced Robotics*, 18:1001–1024, 2004.
4. A. Bachrach et al. RANGE - robust autonomous navigation in GPS-denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011.
5. H. Bai et al. Unmanned aircraft collision avoidance using continuous-state POMDPs. In *Proc. Robotics: Science & Systems*, 2011.
6. K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proc. ICRA*, 2007.
7. A. Bry et al. State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *Proc. ICRA*, 2012.
8. P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.
9. D. Fox et al. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.
10. T. Fraichard. A short paper about motion safety. In *Proc. ICRA*, 2007.
11. R. Hadsell, et al. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.
12. T. M. Howard et al. Model-Predictive Motion Planning: Several Key Developments for Autonomous Mobile Robots. *Robotics Automation Magazine, IEEE*, 21(1):64–73, 2014.
13. T. M. Howard et al. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25(6-7):325–345, 2008.
14. L. P. Kaelbling et al. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.
15. H. Kurniawati et al. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science & Systems*, 2008.
16. A. A. Makarenko et al. An experiment in integrated exploration. In *Proc. IROS*, 2002.
17. J. Michels et al. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proc. ICML*, 2005.
18. U. Muller et al. Off-road obstacle avoidance through end-to-end learning. In *Proc. NIPS*, 2005.
19. J. Pineau et al. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. IJCAI*, 2003.
20. C. Richter et al. High-speed autonomous navigation of unknown environments using learned probabilities of collision. In *Proc. ICRA*, 2014.
21. C. Richter et al. Markov chain hallway and Poisson forest environment generating distributions. Technical Report MIT-CSAIL-TR-2015-014, 2015.
22. S. Ross et al. Learning monocular reactive UAV control in cluttered natural environments. In *Proc. ICRA*, 2013.
23. T. Schouwenaars et al. Receding horizon path planning with implicit safety guarantees. In *Proc. ACC*, 2004.
24. D. Silver and J. Veness. Monte-carlo planning in large POMDPs. In *Proc. NIPS*, 2010.
25. R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proc. ICRA*, 1996.
26. C. Stachniss et al. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proc. Robotics: Science & Systems*, 2005.
27. S. Temizer et al. Collision avoidance for unmanned aircraft using Markov decision processes. In *AIAA Guidance, Navigation, and Control Conference*, 2010.
28. S. Thrun et al. Map learning and high-speed navigation in RHINO. *AI-based Mobile Robots: Case studies of successful robot systems. MIT Press, Cambridge, MA*, 1998.
29. W. Vega-Brown et al. Nonparametric Bayesian inference on multivariate exponential families. In *Proc. NIPS*, 2014.
30. B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. Computational Intelligence in Robotics and Automation*, 1997.