In Proceedings of the Intl. Symposium on Experimental Robotics (ISER 2016).

Learning to Plan for Visibility in Navigation of Unknown Environments

Charles Richter and Nicholas Roy

Massachusetts Institute of Technology
77 Massachusetts Ave, Cambridge, MA 02139
{car,nickroy}@mit.edu

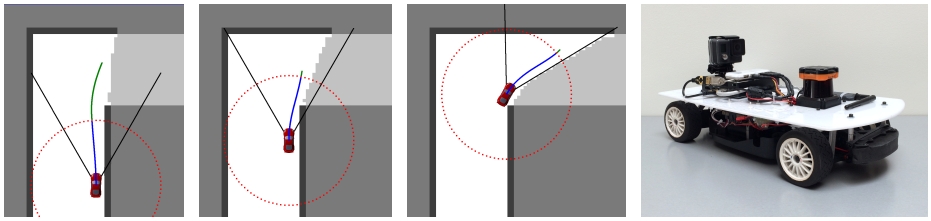
Abstract. For robots navigating in unknown environments, naïvely following the shortest path toward the goal often leads to poor visibility of free space, limiting navigation speed, or even preventing forward progress altogether. In this work, we train a guidance function to give the robot greater visibility into unknown parts of the environment. Unlike exploration techniques that aim to observe as much map as possible for its own sake, we reason about the value of future observations directly in terms of expected cost-to-goal. We show significant improvements in navigation speed and success rate for narrow field-of-view sensors such as popular RGBD cameras. However, contrary to our expectations, we show that our strategy makes little difference for sensors with fields-of-view greater than 80°, and we discuss why the naïve strategy is hard to beat.

1 Introduction

Robot navigation in unknown environments is often performed with a receding-horizon approach, where a motion planner selects a trajectory of some length that makes progress toward the goal while avoiding observed obstacles. Progress is often measured by a heuristic estimate of the remaining cost-to-goal from the end of the planned action. This approach can be represented mathematically with the following optimization problem, which is repeatedly re-solved online to select the optimal action a_t^* as the robot progresses through the environment:

$$a_t^* = \underset{a_t \in A}{\operatorname{argmin}} \; j(b_t, a_t) + h(b_t, a_t), \text{ s.t. } g(b_t, a_t) = 0. \tag{1}$$

Here, a_t is an action to be executed at time t, chosen from a set A of possible actions. The robot's belief b_t contains its knowledge of its own configuration at time t and a partial map built from sensor measurements up to time t. The total estimated cost of choosing action a_t, given b_t, is the sum of the action cost $j(b_t, a_t)$ and the heuristic estimate $h(b_t, a_t)$ of the cost remaining beyond the end of a_t. In this paper, we focus on the minimum-time navigation problem, so $j(b_t, a_t)$ returns the time duration of a_t and $h(b_t, a_t)$ estimates the remaining time to reach the goal after action a_t has been completed. Finally, to guarantee safety, we use a collision constraint $g(b_t, a_t)$, which returns 1 if action a_t intersects either obstacles or unknown regions in the belief, or leads to a state for which collision



(a) 4 m/s. (b) 1 m/s. (c) 1 m/s. (d) Autonomous RC Car.

Fig. 1: Navigating with a simple shortest-path heuristic, the robot greedily keeps to the inside of the turn since it is nearer to the goal. This position occludes the free space around the corner, forcing the robot to slow down to maintain a safe stopping distance. Chosen actions a_t are drawn in blue, emergency-stop actions (lengths proportional to the square of speed) are green, and both are constrained by $g(b_t, a_t)$ to lie within known free space. Black lines indicate the field-of-view.

or entering unknown regions would be inevitable [4], and returns 0 otherwise. We enforce this constraint by ensuring the existence of an "emergency-stop" trajectory that could bring the robot to a stop from the end of action a_t without intersecting an obstacle or unknown region of the map. This collision constraint creates a crucial interaction between the feasibility of actions and the parts of the map that have been observed thus far. The robot is constrained to select trajectories that lie entirely within the known free space of the map, at speeds that ensure it can stop before hitting an obstacle or entering the unknown.

In an unknown map, it is very difficult or impossible to accurately estimate the remaining cost-to-goal—as we will discuss in Section 2, doing so would amount to solving a difficult POMDP—so the heuristic $h(b_t, a_t)$ is inherently a simplified approximation. Common approximations are often based on following the apparent shortest path to the goal at a fixed speed. In this work, we define a shortest-path cost-to-goal estimator $d(b_t, a_t)$, which computes the time needed to reach the goal from the end of a_t, respecting the obstacles in belief b_t, assuming holonomic kinematics, and assuming that speed is held constant at the initial speed of a_t. We implement $d(b_t, a_t)$ using Dijkstra's algorithm on a 2D graph of nodes connected to their 16 nearest neighbors. Using this shortest-path function as the heuristic amounts to setting $h(b_t, a_t) = d(b_t, a_t)$ in equation (1).

To demonstrate the problems with this approach, Figure 1 illustrates a sequence of trajectories planned by a simulated car approaching a blind corner using the shortest-path heuristic. This heuristic greedily guides the robot toward the inside of the corner, where it is unable to see the free space ahead. Without visibility around the corner, the local planner is forced to select an action that slows down from 4 m/s to 1 m/s to preserve sufficient stopping distance.

This common heuristic ignores the fact that the constraints imposed by the current unknown regions of the map may actually be lifted when future observations are taken and more free space is added to the map. A more accurate

heuristic might have guided the robot along a slightly longer path, sacrificing some immediate progress in exchange for improved future visibility into the unknown regions of the map. Greater visibility might, in turn, enable faster actions in future planning steps resulting in an overall reduction in time-to-goal.

Unfortunately, reasoning explicitly about this observation-action interaction typically implies the daunting complexity of POMDPs. To avoid this complexity while retaining information-gathering behaviors, our approach is to augment the shortest-path heuristic with a learned function that models the change in cost-to-goal resulting from the next observation and subsequent action. Next, we will derive this learned model and use it to efficiently reason about the observation-action interactions that are ignored by common shortest-path heuristics.

2 Problem and Technical Approach

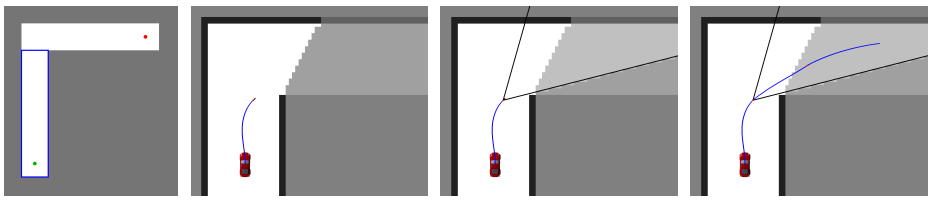
The heuristic $h(b_t, a_t)$ approximates the expected remaining cost-to-goal after taking action a_t, given belief b_t. However, the true optimal value of this quantity represents the solution to a POMDP in which the partially observable map is modeled as a random variable drawn from a distribution over environments [9]. Not only would this POMDP be intractable to solve, but it would also require us to model the distribution over maps, which, for realistic environments, would be an extremely challenging problem unto itself. Without knowing this map distribution, we cannot accurately estimate the probability of future sensor measurements in order to search forward through action-observation sequences.

Rather than employing POMDP solution techniques, our approach is instead to use machine learning to somehow approximate $h(b_t, a_t)$ from training examples. While we could consider attempting to learn the entire global heuristic from data, this is not likely to be a predictable quantity due to the wide range of possible map geometries and sizes. Instead, we observe that $d(b_t, a_t)$ gives reasonable guidance much of the time, but is missing specific information about the effects of possible map observations in the immediate future, which is more local and hence predictable than the entire heuristic. Therefore, we will model the effects of possible future observations in the form of a correction to the shortest-path heuristic. Let $h^*(b_t, a_t)$ represent the expected cost-to-goal under the optimal policy[1]. Our learned function $f_h(b_t, a_t)$ is intended to model the difference between the shortest path heuristic and this true optimal cost-to-goal:

$$f_h(b_t, a_t) \approx h^*(b_t, a_t) - d(b_t, a_t). \tag{2}$$

By capturing this effect from training data, we avoid the need to explicitly model the environment distribution and search over the vast space of observations. And, while we cannot compute $h^*(b_t, a_t)$, even offline during training, we can locally approximate it using a one-step look ahead technique, which we describe next.

[1] Following the POMDP problem formulation developed in [9], this true optimal expected cost would be defined as $h^*(b_t, a_t) = \sum_{s_{t+1}} P(s_{t+1}|b_t, a_t)V^*(s_{t+1})$. We omit a detailed discussion of the POMDP formulation of this problem for brevity, but note that it provides the basis for our mathematical approach and approximations.



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

Fig. 2: (a) Training map, with blue region from which training configurations are sampled. (b)-(d) Training sequence: (b) Random robot configuration is sampled, with belief b_0 and feasible candidate action a_0; (c) the sensor is simulated from the end of a_0, yielding belief b_1 containing additional observed free space; (d) next action a_1 is selected to minimize cost, given the newly observed free space in belief b_1. The choice of a_1 will be used to compute the label of this data point.

2.1 Training

We train the model $f_h(b_t, a_t)$ using a dataset of labeled belief-action pairs, which are intended to represent realistic scenarios the robot could encounter. We generate each data point in simulation by first randomly sampling a robot configuration within a training map. Figure 2a shows a training map we use for hallway environments. Since all corners in our hallway environments share the same geometry (for a given hallway width), it is sufficient to train on a map with a single corner. To restrict ourselves to realistic configurations the robot might encounter while approaching a turn, we sample uniformly from the blue rectangular region, with random headings in the range $[-45°, 45°]$ with respect to the hallway direction, and with random speed.

We then generate a belief b_0 from the point of view of the sampled robot configuration. To generate a realistic belief in general, it would be necessary to aggregate a history of measurements, which in turn would require us to sample a realistic history of states for each data point. To avoid this complication, we generate the sampled belief by simulating a single sensor measurement with a 360° field of view from the sampled configuration. This strategy reveals the local map around the sampled configuration while still capturing the effects of occlusions due to walls, reasonably approximating beliefs encountered at runtime.

Having sampled a configuration and generated the initial belief, we then randomly select a feasible action a_0 the robot could execute, subject to the constraint $g(b_0, a_0) = 0$, which states that the action is collision-free and allows the robot room to stop within known free space (Figure 2b). Next, we simulate the sensor, with the actual field-of-view, from the *end* of a_0 and incorporate the measurement into b_0 to form a new updated belief, b_1 (Figure 2c). Then, we select the next action, a_1, from the end of a_0, that makes the most progress toward the goal given the updated belief b_1 and subject to the constraint $g(b_1, a_1) = 0$ (Figure 2d). Progress is measured with a calculation of $d(b_1, a_1)$, from the

end of a_1. When computing $d(b_1, a_1)$, we assume that the speed of the robot returns from the terminal speed of a_1 to the initial speed of a_0 according to the acceleration limits of the robot, and then maintains that speed to the goal.

The training label is then computed using the optimal choice for a_1:

$$y(b_0, a_0) = \min_{a_1 \in A} [j(b_1, a_1) + d(b_1, a_1)] - d(b_0, a_0). \tag{3}$$

Thus, while we cannot compute $h^*(b_t, a_t)$ even offline during training, our training labels still approximate the desired result of capturing the deviation from the shortest-path heuristic that is due to potential future observations, and subsequent actions the planner could take, contingent upon those observations.

Finally, we compute a low-dimensional set of features, $\phi(b_t, a_t)$ for each belief-action pair that capture the useful predictive information. For this work, we use two features based on the boundary between known free space and the unknown, which we refer to as the "frontier": (1) the fraction of the map frontier in b_t that will be visible from the robot configurations along action a_t, and (2) the average distance from states along the action a_t to all frontier locations in the map.

2.2 Prediction and Planning with the Learned Model

Our dataset of N data points has the form $D = \{(\phi_1, y_1), \ldots, (\phi_N, y_N)\}$ where indices in this case represent different data points (not time t) and we write ϕ_i as shorthand for $\phi(b, a)$ for the i^{th} data point. We use the Nadaraya-Watson estimator to make predictions at runtime:

$$f_h(b_t, a_t) = \frac{\sum_{i=1}^{N} K(\phi(b_t, a_t) - \phi_i) y_i}{\sum_{i=1}^{N} K(\phi(b_t, a_t) - \phi_i)}. \tag{4}$$

Our proposed heuristic function is then the sum of the shortest-distance heuristic and the learned function: $h(b_t, a_t) = d(b_t, a_t) + f_h(b_t, a_t)$. Putting it all together, our planner repeatedly re-solves the following optimization problem:

$$a_t^* = \underset{a_t \in A}{\operatorname{argmin}} \; j(b_t, a_t) + d(b_t, a_t) + f_h(b_t, a_t), \;\; \text{s.t.} \;\; g(b_t, a_t) = 0. \tag{5}$$

To measure the performance of our approach, we will compare it to a "baseline" planner, which uses a simpler control law that is identical to (5), except that it omits $f_h(b_t, a_t)$ and instead uses the simpler heuristic $h(b_t, a_t) = d(b_t, a_t)$.

3 Results

To evaluate our method, we trained a separate model for each combination of sensor field-of-view (from 45° to 120°) and hallway width (2 m, 2.5 m and 3 m). For each of these combinations, we conducted simulation trials in 50 randomly-generated hallway environments [10], recording the performance of our planner as well as that of a "baseline" planner, described above. We selected the hallway

environment type because hallways are very common and also challenging for high-speed navigation due to the visual occlusion at turns. We selected the range of sensor fields-of-view to be relevant for widely used RGBD sensors such as the Microsoft Kinect and Asus Xtion, which produce point clouds of 57˚ and 58˚ horizontal field-of-view, respectively, as well as standard and wide-angle cameras.

We use a re-planning frequency of 20 hz, a planning horizon of 1.5 m for the pre-computed set A of actions, and a combination of feed-forward and feedback control to execute the planned motions. We use a 2D occupancy grid map representation, which can represent free, occupied and unknown cells, and is initialized with every cell unknown. To build maps, we simply project range measurements into the map using raycasting from the vehicle configuration, which we assume is known exactly. We do not use probabilistic or pose-graph optimizing SLAM. In simulation, we used a 10 cm map resolution, which is sufficient to capture the perfectly rectangular shapes of the simulated hallways.

3.1 Learned Models

Figure 3 shows examples of learned models for 60˚ field-of-view in a 2 m wide hallway and 120˚ field-of-view in a 3 m wide hallway. Both models illustrate the same overall trend. Intuitively, the highest cost belief-action pairs occur when actions are located very near to the map frontier while simultaneously offering low frontier visibility (lower left corners of plots in Figure 3). This scenario occurs when the robot approaches a corner hugging the inside wall of the hallway, which enables it to get close to the frontier while keeping the frontier mostly occluded by the wall. In fact, this costly behavior is characteristic of the baseline planner.

Our models predict a reduction in cost if we select actions that either increase distance from the frontier or visibility of the frontier, or both. When the robot is approaching a blind corner, these two features both suggest approaching the corner for a wide turn, not only to increase available stopping distance between the robot position and the frontier, but also to take an observation of more occluded space, thereby pushing the frontier farther ahead.

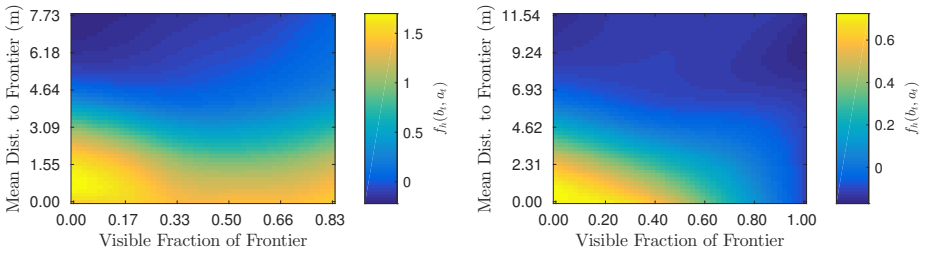


Fig. 3: Learned model $f_h(b_t, a_t)$ for (a) 60˚ field-of-view in 2 m wide hallway and (b) 120˚ field-of-view in 3 m wide hallway.

The only observable deviation from this trend is that in the 60° field-of-view model, the predicted cost begins to increase slightly as the fraction of visible frontier rises above 50%. The reason for this rise is that for narrow field-of-view sensors, the robot must drive directly toward the frontier to observe a large fraction of it, which in hallway environments is correlated with shorter stopping distances and therefore slower speeds. We believe that a more descriptive set of features would produce a more accurate model, but this one nevertheless captures the intuitive notion that keeping greater distance from the frontier and observing some portion of the frontier are both advantageous.

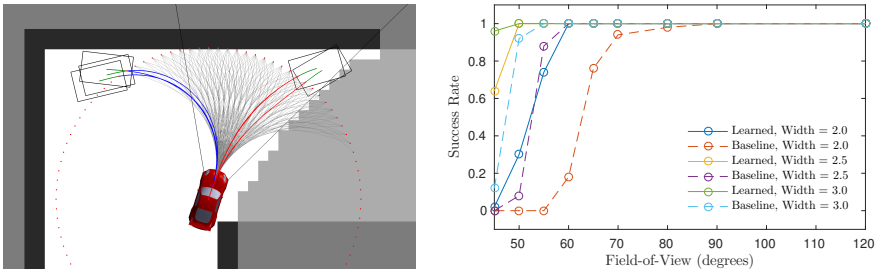
3.2 Simulation Success Rates

We observed the surprising result that for narrow fields-of-view, the baseline planner often failed to reach the goal altogether. These failures resulted from the planner becoming inadvertently trapped in a small region of known free space, such that no feasible actions would yield views of additional free space. The robot was therefore forced to stop. In these cases, a 3-point turn would be required to reorient the robot, take observations revealing free space toward the goal, and proceed. However, we consider these events to be failures.

This scenario is pictured in Figure 4a, where the gray lines represent all actions in action set A. Nearly all of them would cause the robot's bounding box (not shown for these actions) to intersect a wall or an unknown map cell and are therefore infeasible. The exceptions are highlighted in blue and red. For these actions, we then determine whether there exists a safe stopping action, illustrated in green, with the bounding box of the robot at the end of the emergency-stop action illustrated as a black rectangle. The red actions making progress toward the goal do not have safe stopping actions because the bounding box of the robot would intersect both a wall and unknown space. The only feasible choices, with stopping actions lying entirely within known free space, are illustrated in blue. However, these feasible actions neither make progress toward the goal, nor afford any useful sensor viewpoints. Hence the planner has become trapped.

While our planner was also susceptible to this failure mode, our learned guidance function was very effective at avoiding it for certain hallway widths and fields-of-view. Figure 4b shows the success rate for both planners, indicating that our planner was able to reliably reach the goal (for a given hallway width) using a field-of-view approximately 5-10° narrower than what would be required for the baseline planner. While these failures depend on the relative size of the hallway and the length of actions, using a planning horizon that is too short can lead to poor receding-horizon planning behavior in other ways, since short actions cannot span the robot's true range of turning maneuvers. Therefore, we show this dependency by varying the width of the hallway instead.

These results show that the baseline planner is likely to fail under some fairly benign and common conditions. For example, using a Microsoft Kinect sensor on an RC car robot with a 0.8 m turning radius in a 2 m wide hallway could be expected to succeed only 20% of the time, whereas our method succeeds 100% of the time under these conditions.

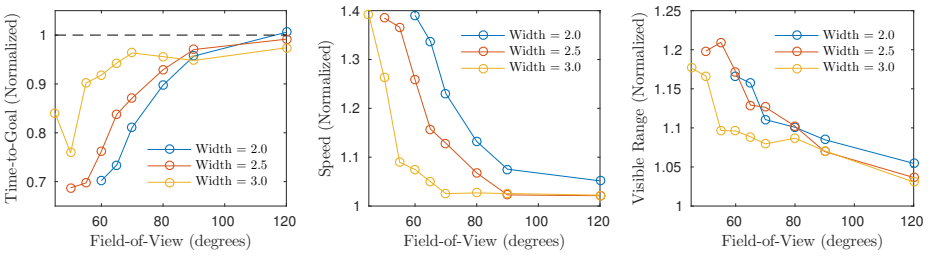


(a) Baseline trapped. (b) Success rates.

Fig. 4: (a) Baseline planner trapped, steering away from the goal, due to insufficient observed free space to the right. The only feasible actions, that are both collision-free and have a feasible emergency-stop action, are blue. See main text for discussion. (b) Success rates in reaching the goal as a function of hallway width and field-of-view, for our method and the baseline planner.

3.3 Simulation Time-to-Goal and Speed as a Function of Visibility

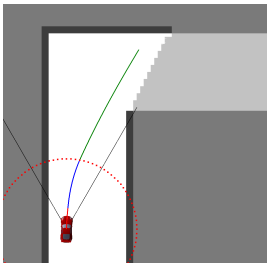
Figure 5a illustrates the time-to-goal for our planner, normalized by the time-to-goal for the baseline planner in the same environments. For a given hallway width and field-of-view, we averaged this normalized time across all trials in which both planners succeeded. By augmenting the shortest path heuristic with a learned model, we observed times-to-goal as low as 70% of the baseline value in 2 m wide and 2.5 m wide hallways, for narrow fields-of-view. We observe smaller, but still substantial improvement in time-to-goal for the 3 m wide hallway and for wider fields of view. It is intuitively sensible that the maximum improvement should occur in cases where the observable free space is most limited.



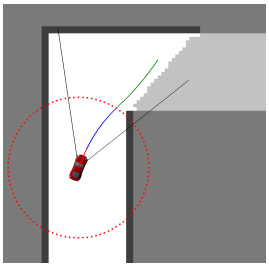
(a) (b) (c)

Fig. 5: Simulation results: (a) average time-to-goal (lower is better), (b) average speed (higher is better), and (c) average maximum visible range. Averaged results are normalized by the corresponding performance of the baseline planner in the same environments. Plot color indicates hallway width.

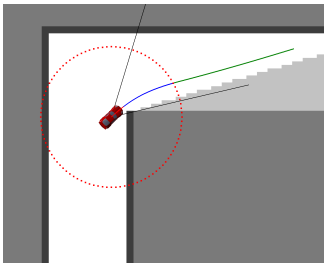
Figure 5b illustrates the average (normalized) navigation speeds corresponding to the time-to-goal results. In each hallway width, our solution is up to 1.4 times faster than the baseline planner. However, since our planner typically travels a slightly longer distance in order to project its sensor visibility around corners, the increased speed does not necessarily translate to shorter times-to-goal, as is the case for 120° field-of-view. Figure 5c quantifies the greater environment visibility achieved by our method. We recorded the maximum visible range in each simulated sensor measurement during simulation trials of our planner and compared those with the maximum visible ranges observed by the baseline planner. In every case, our method was able to view a greater distance ahead, resulting in the ability to plan higher-speed actions.



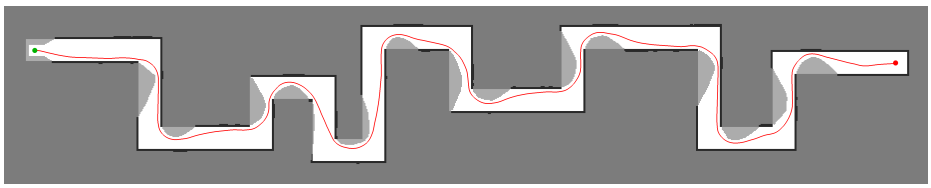
(a) 6 m/s.



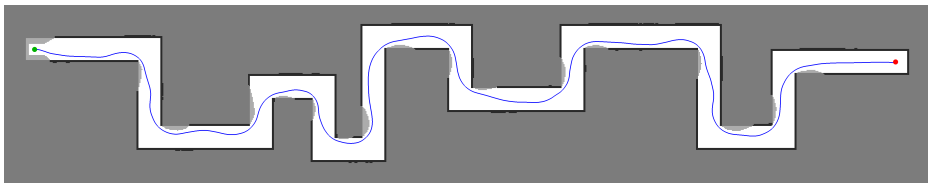
(b) 4 m/s.



(c) 6 m/s.



(d) Simulated trajectory and resulting map of the baseline planner.



(e) Simulated trajectory and resulting map of our planner.

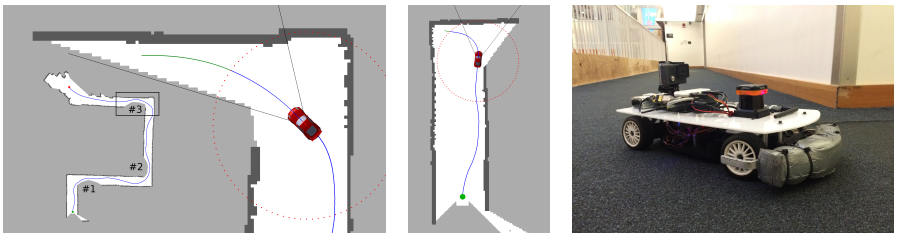
Fig. 6: (a)-(c) Example planning sequence using our method, which guides the robot wide around the corner, gaining visibility and maintaining a higher speed. (d)-(e) Simulated trajectory and resulting observed map of the baseline planner and our method, in a 2.5 m hallway with a 60° sensor field-of-view. Green and red dots indicate start and goal, respectively. Dark gray indicates the hidden map, and light gray regions are those that have not been observed.

Figures 6a-6c show that our method gives rise to the intuitive behavior of swinging wide around corners to obtain more advantageous views of free space enabling greater stopping distances. Compared to the default baseline behavior pictured in Figure 1, this behavior is qualitatively different and faster. Figures 6d and 6e show the trajectories and resulting maps produced by the baseline planner compared to our method in a 2.5 m wide hallway with a 60° field-of-view. Our planner approaches each corner from a wider angle than the baseline planner, observing more of the environment while reaching the goal in less time.

3.4 Experimental Demonstration on Autonomous RC Car

We tested our planner in a hallway environment in the MIT Stata Center on the autonomous RC car pictured in Figure 1. The car is equipped with a Hokuyo UTM-30LX LIDAR, a Microstrain 3DM-GX3-25 IMU and a Gigabyte Brix Intel dual-core i7 computer with 16GB RAM. To perform state estimation, we fuse LIDAR odometry [2] with IMU measurements in an extended Kalman filter. For these experiments, we used a 5 cm map resolution to capture the smaller irregularities of the real environment. We artificially limited the Hokuyo planar LIDAR field-of-view for the purposes of map building (but not for state estimation).

We picked a section of hallway consisting of three 90° turns, with hallway widths of approximately 1.9m, roughly matching our simulated hallway distribution. In this environment, we conducted 10 trials each for the baseline planner and our planner, using both 60° and 90° fields-of-view. Figure 7 illustrates our experimental map and autonomous RC car, and Table 1 summarizes the experimental results. In Table 1, the time-to-goal column averages across only the successful trials in each category, while the mean and maximum speed columns represent averages over all trials in order to quantify the speed differential for the 60° case where the baseline planner did not reach the goal in any trials.



(a) (b) (c)

Fig. 7: (a) Experimental trajectory of our planner through a hallway in the MIT Stata Center with numbered turns (inset) and expanded view of our planner's trajectory through turn #3 (main image), exhibiting good visibility around the corner as a result of its wide-turn behavior. (b) Baseline planner failed at turn #1. (c) Our autonomous RC car in turn #2 of the experimental environment.

Figure 7a shows a map and trajectory from one of our planner's successful trials using a 60° field-of-view. The close-up view of the third turn illustrates a characteristic example of the good visibility of free space our planner was able to obtain, hence allowing it to maintain a greater speed around the turn.

Table 1: Experimental results.

FOV	Planner	Success	Time-to-Goal (s) (successes only)	Mean Speed (m/s) (all trials)	Max Speed (m/s) (all trials)
60°	Baseline	0/10	N/A	2.25	4.29
	Learned	**8/10**	14.36	3.08	5.35
90°	Baseline	4/10	14.26	3.06	5.30
	Learned	**10/10**	13.68	3.43	5.55

With a 60° field-of-view, our planner succeeded in 8 of 10 trials, while the baseline planner failed in every trial. Figure 7b shows a characteristic failure by the baseline planner, becoming trapped as discussed in Section 3.2. The success rates and relative speeds listed in Table 1 roughly match our simulation results for 60° and 2 m hallway width, with difference likely resulting from the fact that the experimental hallway was slightly narrower than 2 m.

Using a 90° field-of-view, our planner succeeded in all 10 trials, while the baseline planner succeeded in only 4. This success rate for the baseline planner is considerably worse than the simulation results, again likely due to the narrower hallway, but our approach is robust to these differences. For those trials that did succeed, Table 1 shows that our planner reached the goal in 4% less time than the baseline planner, which is consistent with our simulation results.

4 Related Work

The planning literature has addressed the problem of safety for agile autonomous vehicles in partially-known environments [1,3,4,9,11,14]. However, these results largely do not consider the contingency between future sensor measurements and action choices, which we target in this work. POMDP algorithms explicitly address this action-observation contingency, but require knowledge of the environment distribution and even approximate methods are generally intractable online [5,6]. Several methods have efficiently captured the action-observation contingency using a discrete or topological abstraction [12,7]. However, these methods do not readily translate to realistic sensor measurements or low-level vehicle dynamics. Planning of sensor viewpoints can be found in the exploration literature [8,15,13], where the objective is to build a more complete or accurate map. However, he exploration literature assigns a fundamentally different value to measurements than the time-to-goal value we have used in this work.

5 Conclusion

Our results are surprising in several respects. First, we did not anticipate the failure mode of robots becoming trapped in a small area of known free space,

unable to make progress toward the goal. Given how often this failure mode occurs under common conditions, it warrants a solution such as the one we provide in this work. Second, we expected a more substantial benefit using our refined heuristic across a wider range of fields-of-view. Instead, there is little benefit for fields-of-view greater than 80°, as the resulting speed improvement does not outweigh the extra distance traveled to obtain better visibility. Ultimately navigation speed is limited by stopping distance, which grows quadratically with the speed of the robot. Therefore, to increase speed, the length of known free space must grow quadratically as well. However, our learned guidance function was able to produce only a modest increase in sensor visibility. Some possibilities for future work might be to quantify a bound on the maximum possible benefit that could be gained from the optimal policy, to explore the characteristics of the environment that affect that bound, and to implement or learn a richer set of predictive features to help get closer to the optimal policy.

References

1. S. Arora, et al. Emergency maneuver library–ensuring safe navigation in partially known environments. In *Proc. ICRA*, 2015.
2. A. Bachrach et al. RANGE - robust autonomous navigation in GPS-denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011.
3. K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *Proc. ICRA*, 2007.
4. T. Fraichard and H. Asama. Inevitable collision states–a step toward safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.
5. L. P. Kaelbling et al. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1):99–134, 1998.
6. H. Kurniawati et al. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. RSS*, 2008.
7. M. Likhachev and A. Stentz. PCPP: Efficient probabilistic planning with clear preferences in partially-known environments. In *Proc. AAAI*, 2006.
8. A. A. Makarenko et al. An experiment in integrated exploration. In *Proc. IROS*, 2002.
9. C. Richter et al. Bayesian learning for high-speed navigation in unknown environments. In *Proc. ISRR*, 2015.
10. C. Richter et al. Markov chain hallway and Poisson forest environment generating distributions. Technical Report MIT-CSAIL-TR-2015-014, 2015.
11. T. Schouwenaars et al. Receding horizon path planning with implicit safety guarantees. In *Proc. ACC*, 2004.
12. R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. IJCAI*, 1995.
13. C. Stachniss et al. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proc. RSS*, 2005.
14. M. Watterson and V. Kumar. Safe receding horizon control for aggressive MAV flight with limited range sensing. In *Proc. IROS*, 2015.
15. B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. Computational Intelligence in Robotics and Automation*, 1997.