Map-Predictive Motion Planning in Unknown Environments

Amine Elhafsi[1], Boris Ivanovic[1], Lucas Janson[2], Marco Pavone[1]

***Abstract*— Algorithms for motion planning in unknown environments are generally limited in their ability to reason about the structure of the unobserved environment. As such, current methods generally navigate unknown environments by relying on heuristic methods to choose intermediate objectives along frontiers. We present a unified method that combines map prediction and motion planning for safe, time-efficient autonomous navigation of unknown environments by dynamically-constrained robots. We propose a data-driven method for predicting the map of the unobserved environment, using the robot's observations of its surroundings as context. These map predictions are then used to plan trajectories from the robot's position to the goal without requiring frontier selection. We applied this map-predictive motion planning strategy to randomly generated winding hallway environments, yielding substantial improvement in trajectory duration over a naïve frontier pursuit method. We also experimentally demonstrate similar performance to methods using more sophisticated frontier selection heuristics while significantly reducing computation time.**

I. Introduction

As robots move from meticulously-organized research labs and factory floors into natural or human-built environments, they must be capable of safely and efficiently planning trajectories through unexplored spaces. This is particularly important for a variety of robotic applications, such as search and rescue operations in uncharted areas, autonomous driving with occluded perception, and planetary exploration.

A common approach to planning in unknown environments relies on repeatedly replanning trajectories towards intermediate objectives within the robot's known environment, with the expectation that the robot will eventually arrive at its desired goal. However, such a planning strategy is shortsighted in that it does not reason about the map beyond the observed environment, resulting in greedy and inefficient trajectories. For example, a robot navigating an indoor environment may suddenly observe a perpendicular corridor, calling for an abrupt maneuver to complete the turn—especially if traveling at higher speeds as conceptualized by Fig. 1. Similarly, the pursuit of a frontier point, without reasoning about what may lie beyond, may lead the robot towards an occluded obstacle, requiring the execution of some emergency maneuver.

We begin by enumerating four desiderata of a motion-planning algorithm for unknown environments. First, the method must be capable of performing planning for robots with dynamic constraints. Second, robot safety must be guaranteed with respect to observed obstacles and the unobserved environment, where unseen obstacles may reside. The definition of safety in this work requires that the algorithm does not produce trajectories in collision with these regions of the workspace. Additionally, the robot should not put the robot into a state where future collisions may be unavoidable. Third, this method should be capable of incorporating reasoning about the unobserved environment in a manner that is invariant to the prediction model. Finally, the algorithm must lend itself to rapid replanning such that it is amenable to real-world applications. We motivate our proposed method by considering these desiderata among prior work related to this problem.

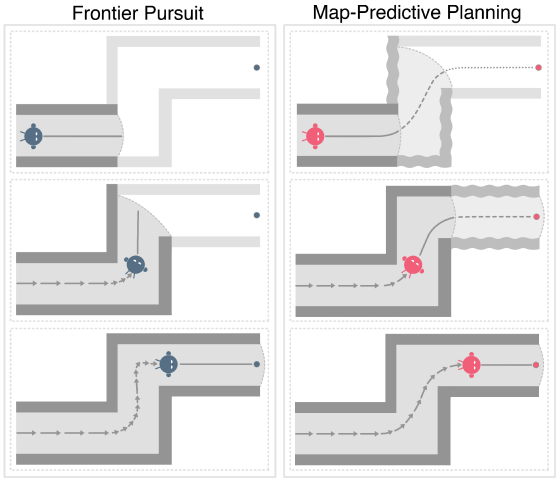

Fig. 1. **Left:** A frontier pursuit method plans without reasoning about the unobserved environment. As a result, it is not prepared for the turn and must slow down excessively and turn sharply. **Right:** Our motion planning method accounts for potential upcoming turns via map prediction, enabling a smoother and more efficient trajectory through the hallway.

II. Related Work

Traditionally, motion planning strategies in unknown environments have relied on repeatedly replanning trajectories towards intermediate objectives until the final goal region is reached [1], [2], [3]. This receding-horizon style of planning usually relies on unsophisticated heuristics to choose an intermediate objective (e.g. pursuing the frontier point nearest the goal at each iteration [4]; more sophisticated heuristics may be found in [5]). However, these heuristics often yield greedy, short-sighted behaviors as previously described.

Several works have proposed techniques for reasoning about the unobserved environment to navigate through them more efficiently. One class of works seek to use the robot's current knowledge of the environment to better inform high-level routing decisions through topologically complex environments (e.g. indoor building floor plans). [6] proposes

*This work was supported in part by NSF, Award Number: 1931815, by NASA under the NSTRF program, and by TRI. This article solely reflects the opinions and conclusions of its authors and not of NSF, NASA, TRI or any other Toyota entity.

[1]Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA {amine, borisi, pavone}@stanford.edu

[2]Department of Statistics, Harvard University, Cambridge, MA 02138, USA ljanson@fas.harvard.edu

a method of navigating partially-revealed environments in minimum distance by modeling the problem as a Partially Observable Markov Decision Process (POMDP), where optimal actions correspond to frontiers that lead directly to the goal rather than lengthy detours or dead ends. Towards a similar objective, [7] passes an occupancy grid encoding a robot's current knowledge of the environment through a Convolutional Neural Network (CNN) and uses the output to weigh frontiers based on their likelihood of leading to a point of interest. While these works enable shorter-distance navigation through unknown environments to a desired goal, they crucially do not consider robot dynamics.

Another class of such works look to directly predict the structure of the unknown environment. [8] represents building floor plans as graphs, where nodes correspond to rooms and edges indicate traversable paths, and proposes a method of predicting extensions to the known environment topology. Such an approach would be useful for improving high-level routing, yet the graph representation of the environment is not amenable to low-level trajectory planning for dynamic robots. [9], [10] more explicitly attempt to predict an occupancy grid of the unknown environment by comparing the robot's current observations against a collection of previously stored maps. The prediction process compares the surroundings of an unexplored region with the built map of explored regions. However, the accuracy and feasibility of this prediction depends on the robot having previously observed similar structure, hampering generalization. Drawing inspiration from image inpainting literature (e.g. [11], [12]), [13], [14] treat the problem of occupancy grid prediction as one of image completion. These works propose using CNNs which take as input an occupancy grid of the observed environment and produce occupancy predictions for the unknown regions as outputs. However, a CNN-based approach requires that the inputs and outputs be of some fixed sizes set at training time. Due to the dynamic nature of navigation in unknown environments, available information and desired outputs may vary. For example, in confined environments a robot's sensing is limited and only a small subset of the CNN input would be required whereas sparse environments would require much larger inputs to provide enough context. Similarly, high-speed navigation would require a significantly larger prediction region than low-speed travel.

Several works have also sought to combine reasoning about the unknown environment with planning. [15] proposes a neural network architecture that is trained to learn a mapping from first-person views to a discrete movement action set (i.e. stay still, move forward, turn left/right), and is capable of navigating novel environments. Yet, beyond the limited action space, such an end-to-end learning-based approach is effectively a black-box function which provides little insight behind the robot's reasoning for choosing a particular action. [16] introduces a planner for dynamic, high-speed navigation through unknown environments. The planning problem is formulated as a POMDP where actions are chosen from a library of motion primitives minimizing time and collision probability. Reasoning about the unobserved environment is implicitly performed via the prediction of collision probabilities which are learned as a function of the robot action and a set of hand-coded features encoding the robot's immediate observations. [17] proposes a similar planner for visual navigation by mapping images and robot actions to collision probabilities using a neural network. While these works demonstrate significant performance gains in terms of trajectory time, they crucially lack formal safety guarantees.

Along the lines of combining map prediction and planning, [18] introduces a general framework for tackling this problem and is closely related to this work. As such, we formulate our motion problem according to this framework and experimentally compare to a baseline algorithm which is an instantiation of this framework.

Statement of Contributions: To address the problem of motion planning in unknown environments, we make the following three contributions: (1) We introduce a data-driven approach to predicting structured environments (such as a grid or maze-like hallway environment) in regions that are occluded or beyond sensor range. In particular, our method produces human-interpretable probabilistic map predictions that can be passed to any general motion planning algorithm. (2) We propose a method of incorporating these predictions within a motion planning framework that guarantees robot safety. This framework makes use of the predictions to penalize trajectories passing through regions with a high probability of occupancy. (3) We demonstrate, with extensive numerical experiments, that using these predictions yields a substantial improvement in trajectory time over a naïve frontier pursuit method and significant computation time reduction over methods using more sophisticated frontier selection heuristics. Our method uniquely addresses all four aforementioned desiderata; it is capable of planning safe, dynamically-constrained trajectories with low computation times amenable to real-time performance.

III. Problem Formulation

In order to more efficiently navigate unknown environments, we seek to predict the probability of occupancy at some set of unobserved points in the environment given some nearby observations as context. We then formulate a motion planning problem seeking a safe trajectory minimizing a desired cost function through these environments.

A. Map Prediction

We begin with the assumption that robot perception is achieved via some range-limited line-of-sight mechanism such as a lidar. We further assume that the robot's perception is incorporated within a deterministic occupancy grid representation of the environment, encoding regions of free, occupied and unknown space. Finally, we make the assumption that the environment is static.

We consider the environment's occupancy grid to be an n-tuple random variable $(Y_1, \ldots, Y_n)$ whose elements represent the occupancy at grid cell i and are described by some unknown distribution. We denote $y_i \in \mathcal{Y} = \{0, 1\}$ to be a realization of Y_i. We also define a map's set of spatial coordinates as the set $\mathcal{X} \in \mathbb{R}^2$, with the observed and unobserved regions denoted by $\mathcal{X}_{obs} \subset \mathcal{X}$ and $\mathcal{X}_{un} = \mathcal{X} \setminus \mathcal{X}_{obs}$, respectively. We let the context set $\mathcal{C} = \{(x_i, y_i)\}_{i=1:c}$, correspond to the set of c observed locations $x_j \in \mathcal{X}_{obs}$ for

$j = 1, \ldots, c$ paired with their known occupancy state y_i. Finally, the target set is denoted by $\mathcal{T} = \{x_i\}_{i=(c+1):(c+t)}$ and corresponds to the t spatial coordinates $x_k \in \mathcal{X}_{un}$ for $k = c+1, \ldots, c+t$ for which we seek the occupancy value.

It is then assumed that there exists some stochastic process P over functions $f : \mathcal{X} \to \mathcal{Y}$. For some function $f \sim P$, we set $y_i = f(x_i)$. As such, the stochastic process defines the joint distribution $P(\{f(x_i)\}_{i=1:(c+t)})$ and thus the conditional distribution $P(f(\mathcal{T}) \mid \mathcal{C}, \mathcal{T})$. Given this formulation, the objective is to determine this conditional distribution.

B. Motion Planning with Map Predictions

The motion planning problem is concerned with planning trajectories to minimize some objective function of a trajectory σ, subject to initial and terminal boundary conditions, while satisfying dynamic, control and safety constraints. We define a trajectory, σ, as a tuple of states, $\mathbf{s}(t)$, control inputs, $\mathbf{u}(t)$, and duration, T. In order to make use of map predictions, we focus on strategies that repeatedly replan trajectories as new information is observed. However, rather than computing partial trajectories to intermediate objectives at each iteration, we choose to plan all the way to the goal using predictions to guide the trajectory computation in unknown regions following the framework presented by [18]. In effect, at each iteration we seek trajectories solving the optimization problem:

$$
\begin{aligned}
\underset{\sigma_\mathrm{k}, \sigma_\mathrm{u}}{\text{minimize}} \quad & c(\sigma_\mathrm{k}, \sigma_\mathrm{u}) + h(\sigma_\mathrm{u}, \Phi) \\
\text{subject to} \quad & \mathbf{s}_\mathrm{k}(t_\text{initial}) = s_\text{initial} \\
& \mathbf{s}_\mathrm{k}(t_\text{frontier}) = \mathbf{s}_\mathrm{u}(t_\text{frontier}) \\
& \mathbf{s}_\mathrm{u}(t_\text{final}) \in \mathcal{S}_\text{final} \\
& \sigma_\mathrm{k}, \sigma_\mathrm{u} \in \Sigma_\mathcal{D} \\
& \mathbf{u}_\mathrm{k}(t), \mathbf{u}_\mathrm{u}(t) \in \mathcal{U} \ \forall \ t \\
& \mathbf{u}_\mathrm{k}(t_\text{frontier}) = \mathbf{u}_\mathrm{u}(t_\text{frontier}) \\
& \mathbf{s}_\mathrm{k}(t) \in \mathcal{S}_\text{free} \ \forall \ t \\
& \mathbf{s}_\mathrm{k}(t) \notin \mathcal{S}_\text{unsafe} \ \forall \ t.
\end{aligned} \tag{1}
$$

In this formulation, we partition the full trajectory σ into subtrajectories σ_k and σ_u. We designate σ_k as the immediate trajectory in the known region for the robot to execute, terminating at a frontier. We use σ_u to represent a tentative future trajectory in the unknown region beginning from the end point of σ_k. The objective function consists of the terms $c(\sigma_\mathrm{k}, \sigma_\mathrm{u})$ the primary cost we wish to minimize, and $h(\sigma_\mathrm{u}, \phi)$, a penalty term to discourage σ_u from passing through regions of high predicted occupancy. States and controls share the same subscript convention. We enforce initial and final state constraints with s_initial representing the initial robot state and $\mathcal{S}_\text{final}$ representing the set of acceptable goal states. The set of dynamically feasible trajectories and admissible control actions are denoted $\Sigma_\mathcal{D}$ and $\mathcal{U}$, respectively. Finally, $\mathcal{S}_\text{free}$ represents the set of collision-free states while $\mathcal{S}_\text{unsafe}$ represents any chosen set of states that compromise the robot's safety, such as Inevitable Collision States (ICS) [19]. Within this setting, an ICS is defined as a state for which the robot will either eventually collide with an obstacle or enter unobserved space (where obstacles may reside), despite any control sequence the robot may execute.

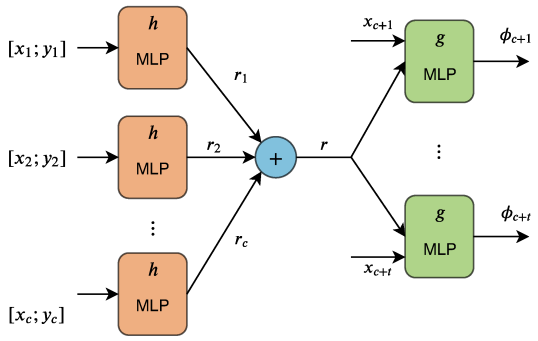

Fig. 2. Our map prediction CNP architecture. Context points and their occupancies $\{x_i, y_i\}_{i=1:c}$ are first encoded by h, a multilayer perceptron (MLP), to generate representation vectors $\{r_i\}_{i=1:c}$. These representation vectors are then averaged to produce an overall scene encoding r. The vector r is then concatenated with each of the target points $\{x_i\}_{i=(c+1):(c+t)}$ and fed through our decoder g, another MLP, which produces occupancy probabilities $\{\phi_i\}_{i=(c+1):(c+t)}$ for each of the target points.

Note that state and control continuity are enforced at t_frontier, the time at which the trajectory first crosses a frontier into unknown space. The last two constraints guarantee safety by requiring σ_k to be collision free and outside of any ICS.

Rather than thresholding predictions and enforcing that σ_u be collision free with respect to the predicted map, our formulation seeks to simply bias σ_u away from regions of high occupancy probability. We find that this formulation is more forgiving in the event that predictions are inaccurate, e.g. in cases where there is only one route to a goal and inaccurate thresholded predictions would wall it off, leading to infeasibility. In practice, we solve this problem at each iteration, execute a portion of σ_k, update the robot's map of the environment, and repeat this process until the goal is reached. Although the robot should never execute any part of σ_u, its presence in the optimization influences σ_k to navigate the known environment in a more informed manner, e.g., better situating the robot for a predicted turn in the future.

IV. Map-Predictive Motion Planning with Conditional Neural Processes

A. Map Prediction

Machine learning-based approaches are uniquely primed to model the distribution over maps, especially given the variety and complexity of real-world environments. Machine learning models can implicitly capture the salient features of environments from data, rather than formulating their distribution explicitly. Specifically, we choose a Conditional Neural Process (CNP) [20] architecture to directly parametrize the conditional stochastic process approximating $P(f(\mathcal{T}) \mid \mathcal{C}, \mathcal{T})$. In contrast to CNNs, CNPs are flexible in that they accept arbitrarily many inputs, as determined by the size of $\mathcal{C}$, as well as query arbitrarily many predictions, as determined by $\mathcal{T}$. This approach also has similarities to Gaussian process regression, yet is more scalable as it overcomes the need for costly matrix inversions.

The CNP architecture consists of an encoding neural network, an aggregation operation and a decoding neural network to produce parameters of the approximating conditional distribution $Q(f(\mathcal{T}) \mid \mathcal{C}, \mathcal{T})$. The CNP architecture is illustrated in Fig. 2.

The encoding procedure involves using the encoding neural network, $h(x_i, y_i)$, to produce embeddings r_i for each of the c context pairs $(x_i, y_i) \in \mathcal{C}$. These embeddings are then aggregated into a single conditioning representation vector r. In the CNP formulation, this aggregation operation, $a(r_{1:c})$, may take the form of any commutative operation mapping multiple vectors in $\mathbb{R}^d$ to a single vector in $\mathbb{R}^d$. Finally, the decoding network, $g(x_i, r)$ produces a vector of parameters, ϕ, of a distribution over the occupancy of some $x_i \in \mathcal{T}$.

The encoding network was designed as a four-layer fully-connected feedforward network to produce a 256-dimensional r_i. Each layer consists of 256 neurons with Rectified Linear Unit (ReLU) activations. For the aggregation operation, we choose to simply average the embeddings since this operation weighs all information equally and ensures a similar magnitude between the embeddings and r—characteristics beneficial to network stability once deployed.

Finally, the decoding network consists of another four-layer fully-connected network with each layer consisting of 256 ReLU activated neurons. The final layer feeds into a sigmoid output representing a scalar-valued ϕ. The model was trained to minimize a negative log-likelihood loss function such that ϕ could be interpreted as parametrizing a Bernoulli distribution over a target point's occupancy. With the predicted occupancy values in hand, we now describe how we these predictions are incorporated within our motion planning framework.

B. Motion Planning

For this work, we are interested in navigating unknown environments in minimum-time and as such take $c(\sigma_\mathrm{k}, \sigma_\mathrm{u})$ to represent the overall trajectory duration, T. We also choose

$$h(\sigma_\mathrm{u}, \Phi) = \alpha \int_{\mathbf{s}_\mathrm{u}} \frac{1}{1 - \Phi(\mathbf{s}_\mathrm{u}) + \epsilon} \, d\mathbf{s}_\mathrm{u}$$

which yields a higher cost for σ_u passing through regions of high occupancy probability. Here, $\Phi(\mathbf{s}_\mathrm{u})$ is a function mapping the state $\mathbf{s}_\mathrm{u}$ to a predicted occupancy, α is a scaling parameter chosen to be 0.25 experimentally and ϵ is a small, positive constant introduced to avoid singularities. In this work we present one possible selection of c and h, but we emphasize that the framework is general and alternative choices may be selected depending on designer preferences.

We take a simple friction circle car as our system of interest with dynamics

$$m\ddot{\mathbf{s}}(\mathbf{t}) = \begin{bmatrix} \cos\theta(t) & -\sin\theta(t) \\ \sin\theta(t) & \cos\theta(t) \end{bmatrix} \mathbf{u}(t)$$

where m is the car's mass, θ is the car's instantaneous heading. The state $\mathbf{s} \in \mathbb{R}^2$ represents the robot position. The control input $\mathbf{u}$ may be explicitly written as $[\mathbf{u}_\text{long} \ \mathbf{u}_\text{lat}]^\intercal$ with the first and second components respectively representing longitudinal and lateral forces in the vehicle body frame. The friction circle constraint is expressed as

$$\|\mathbf{u}\| \leq \mu m g$$

where μ is the friction coefficient and g is the gravitational constant. A constraint specifying the minimum turning radius, R_min, is also enforced as

$$\mathbf{u}_\text{lat} \leq m \frac{\|\dot{\mathbf{s}}\|^2}{R_\text{min}}.$$

We modeled our system after a radio-controlled (RC) car with $m = 2.5$ kg, $\mu = 0.9$, R_min = 0.5 m.

To approximate solutions to (1) at each iteration, we perform a grid search to find a reference path from the robot's position to the goal, smooth this path to be feasible under our system's dynamics and then optimize the speed profile for minimum-time traversal. As compared to [18]'s use of the sampling-based planner FMT* [21], we choose this decomposed motion planning strategy with practical realizability (i.e. real-time computation requirements) in mind.

We use the A* search algorithm to find the path from the robot's position to the goal. We use the Euclidean distance heuristic and treat obstacles as impenetrable within the known region. Within the unknown region, we attempt to account for the effect of h in the objective function by multiplying the Euclidean distance heuristic by $\alpha/(1-\phi_i+\epsilon)$.

Using the A* path as a reference, we then solve the convex optimization problem formulated by [22] to produce a dynamically-feasible trajectory. This smoothing operation begins by placing a sequence of "bubbles" about the points constituting the A* reference path defining a collision-free "tube." Within this tube, the optimization problem is formulated as to minimize the curvature of the path subject to dynamic constraints. As an implementation consideration, we compute bubble radii for the A* path in the known region using the distance from the corresponding reference point to the nearest obstacle. Since we do not know the locations of the obstacles in the unknown regions, we select a constant, relatively small bubble radius[1] to ensure that the smoothed trajectory does not excessively stray from the reference path.

Finally, the speed profile of this trajectory is optimized for minimum time traversal by solving the convex optimization problem presented by [23]. In order to ensure safety, we enforce the additional constraint that the robot velocity must be zero at the end of σ_k. We reason that, provided the robot begins in a safe state and succeeds in planning a collision-free trajectory in the manner described, it will always be able to safely return to zero velocity within the known region of the environment. If any of the optimizations returns an infeasible solution at a particular iteration i, it would suffice to execute the remainder of the trajectory $\sigma_{\mathrm{k},i-1}$ planned at the previous iteration to safely bring the robot to rest.

V. Experiments

We have trained a CNP model using data obtained from a set of 75 randomly-generated, single-path maze environments consisting of frequent corners and U-turns with 2.5 m hallways spanning a 25 m $\times$ 25 m area. To generate training data, we sample 504 unoccupied points from each map and simulate a 5 m range lidar scan. The spatial coordinates of the visible occupancy grid cells, expressed relative to the robot, and their occupancies then make up the context set $\mathcal{C}$. The target set $\mathcal{T}$ consists of the coordinates corresponding to points in $\mathcal{C}$ in addition to the coordinates of unobserved points within a 7.5 m radius about each frontier. Although our goal is to teach the model to predict the map beyond the frontiers, including observed points in $\mathcal{T}$ was found to benefit prediction accuracy. The model was trained for 1,000,000 iterations using a batch size of 4.

[1]We have found values similar to the vehicle's turn radius to perform well as demonstrated in our experiments.

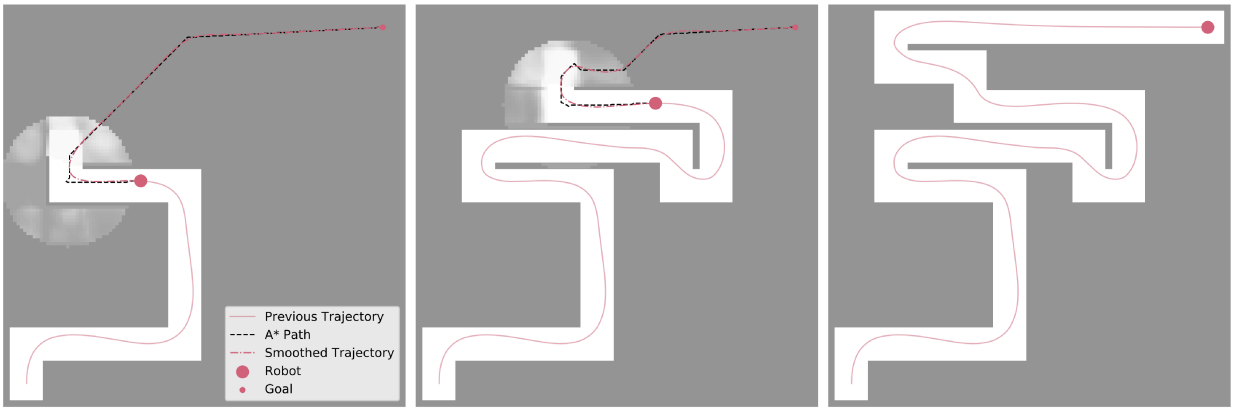

Fig. 3. **Left:** Our method plans from the robot's current position to the goal, taking into account the predicted occupancy probabilities outside of the known environment. Our CNP's predictions are visualized centered at the current frontier point. Occupancy probabilities range from white (free space) to dark grey (occupied). **Middle:** A* plans through a higher occupancy probability region, even though our model correctly predicted a hallway to the left. This occurs because our A* heuristic trades off a higher occupancy probability penalty for a lower distance to the goal. **Right:** This is mitigated by our algorithm's frequent replanning; there is no detriment to the executed trajectory.

We evaluate our methodology[2] on a set of 30 randomly-generated mazes not seen during training. We compare against a naïve frontier pursuit baseline and [18]'s method. The naïve baseline simply plans trajectories within the known map to the frontier centroid nearest the goal. Our implementation of [18] faithfully recreates their heuristics, with two exceptions: (1) We use the method described in Section IV-B for planning trajectories rather than FMT* (to reduce runtime complexity), and (2) we plan over an occupancy grid as opposed to a polygonal representation of the environment (to better match real-world hardware). Our CNP model was written in TensorFlow [24] with training and experimentation performed on a desktop computer running Ubuntu 18.04 equipped with an AMD Ryzen 1800X CPU and two NVIDIA GTX 1080 Ti GPUs. Path smoothing and trajectory optimization were performed with MOSEK [25] interfaced through Convex.jl [26] in Julia [27]. In this section, any stated P values are obtained from two-tailed t-tests on the means of the two quantities differing.

A. Planning with Map Predictions

A key component of our methodology is the ability to predict what might lie in the unknown region, conditioned on current observations, and incorporate that prediction in motion planning. Fig. 3 shows a robot navigating through one of our mazes, visualizing our predictions of the unknown region. In it, we can see how our soft penalty encourages A* to plan paths through predicted free space for some distance into the unknown region, leading to smoothed paths that better prepare the robot for turns.

We can also see emergent behavior as a result of planning ahead with map predictions in Fig. 3. Specifically, Fig. 3 (left) shows robot turning away from the second corner, prior to reaching it, enabling it to better hit the apex and maintain speed through the turn.

B. Maze Completion Time Analysis

We quantitatively compare our method to the naïve baseline and [18] in terms of completion time relative to the optimal trajectory. In Fig. 4, we can see that our method significantly outperforms the naïve baseline ($P < .001$) for all maximum speed values. This is especially true as maximum speed increases, showing the efficacy of map predictions for maintaining path smoothness and speed through turns. Additionally, our method generally matches the performance of [18]. For maximum speeds of 3 and 4 m/s, our method is directly comparable to [18] ($P = \{0.17, 0.46\}$). For lower maximum speeds, our method is only 2% worse in relative completion time (105% vs. 107%, $P = \{.042, .046\}$).

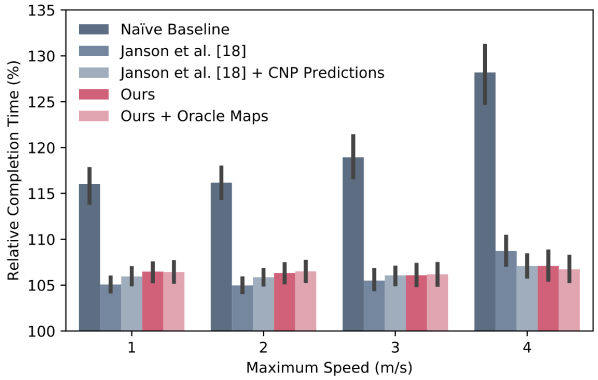

Fig. 4. Mean completion times for each method relative to the optimal completion time (lower is better). Error bars show bootstrapped 95% confidence intervals.

To determine if there is any performance that a better map prediction model could gain for our algorithm, we compare our method with an ideally-augmented version of itself (using perfect map predictions), which is referred to as "Oracle Maps" in Fig. 4. Our method performs equally as well as it would with perfect map predictions for all maximum speed values ($0.74 \leq P \leq 0.96$). This confirms that there is no performance to be gained by switching our CNP model for another type of map predictor.

Since our map predictions can be paired with any motion planner, we augmented [18] with CNP predictions to determine how probabilistic map knowledge aids performance. As can be seen in Fig. 4, the methods perform similarly ($0.12 \leq P \leq 0.43$). However, as maximum speed increases, [18] with map predictions demonstrates more consistent performance compared to [18], even achieving a better average value at the

[2]All of our source code, trained models, and data are publicly available online at `https://github.com/StanfordASL/MapPredictiveMotionPlanning`.

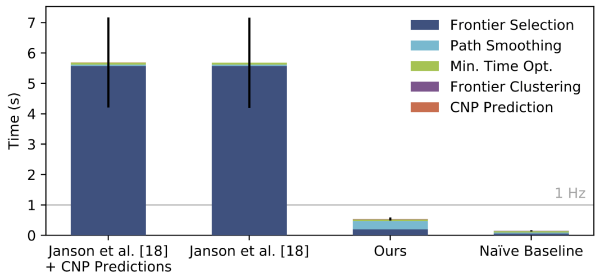

Fig. 5. Mean runtime of one planning step for each method, with subcomponent contribution shown as proportions of each bar. Our method consistently runs at $\sim$ 2 Hz. Error bars show 95% confidence intervals.

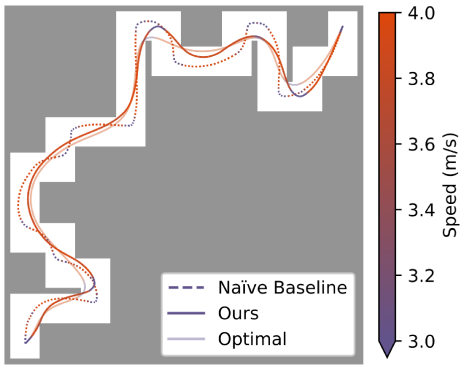

Fig. 6. Our method, the naïve baseline, and the optimal trajectory visualized with speed indicated by color. The baseline sees harsher braking and takes sharper turns than our method and the optimal trajectory.

highest maximum speed. This consistency in performance is also shared by our method as it maintains only a 7-8% deficit to the optimal trajectory across maximum speed values, and is especially visible when compared to the naïve baseline's worsening trend in Fig. 4.

C. Computation Time

While our method and [18] share similar performance in maze completion time, our method executes significantly faster. Fig. 5 shows a comparison of the runtime of both methods, as well as [18] with CNP predictions and the naïve baseline. Our method, with an unoptimized Julia implementation and general-purpose solver, consistently achieves 0.5 s (2 Hz) execution time per planning iteration ($\sim 11\times$ faster than [18]). This is fast and predictable enough for real-time replanning on, e.g., a robot moving at 4 m/s with sensor and prediction ranges of 7.5 m and 5 m, respectively. The longest-running parts of our method are path smoothing and frontier selection, i.e., executing A* from the robot to the frontier centroids and from the frontier centroids to the goal, and smoothing the resulting path. Notably, obtaining map predictions by forward propagation of the CNP model yields negligible overhead, executing in 10 ms on average.

Beyond very long execution times, [18] also suffers from a high runtime variance. This is undesirable in practice as it is difficult to estimate how far ahead to plan in order to ensure a real-time replanning frequency.

D. Path and Speed Profiles

Fig. 6 shows the trajectories and speed profiles obtained by the naïve baseline, our method, and the optimal trajectory. As the baseline cannot foresee corners before they are visible,

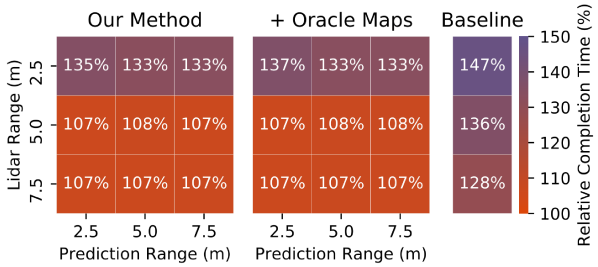

Fig. 7. The relative completion times of our model, our model with perfect predictions, and the naïve baseline for different sensor and prediction ranges with a maximum speed of 4 m/s.

it acts late and needs to slow down heavily to make most turns. Conversely, our method closely follows the optimal trajectory and its speed profile through the map. This is especially noticeable in the first set of zig-zags where the baseline must slow down and turn for each corner. Our method predicts that the zig-zags will continue and plans a smooth, straight trajectory through them, nearly matching the optimal trajectory.

E. Performance under Different Conditions

Since there are a variety of sensors used in robotics, there are many sensor ranges for which this method could be used. Our CNP was trained to predict 5 m into the unknown region from context points observed with a 7.5 m range laser scanner. To verify that the CNP can generalize to a variety of sensor and prediction ranges, we evaluate our method on different lidar and prediction ranges. Fig. 7 shows the results of this evaluation.

As before, our method generally outperforms the naïve baseline and matches the augmentation with ideal map predictions. Notably, all methods perform poorly when the sensor's range is 2.5 m. Such a small observation range means the robot does not have enough time to accelerate to maximum speed before needing to slow down to maintain safety, degrading completion time. Importantly, we maintain algorithm performance for a prediction range of 7.5 m, which is larger than our CNP's trained prediction range of 5.0 m.

VI. Conclusion and Future Work

In this work, we present a unified method that combines a map prediction and motion planning method to enable safe, time-efficient autonomous navigation of unknown environments with real-time performance. However, one limitation of this work is that the CNP model had difficulty producing predictions in more sophisticated, real-world environments (such as [28]). As a result, a direction for future research will be to explore new map prediction algorithms that perform well in a wider and more realistic class of environments. A potential modification of the current prediction approach could involve changing the encoding network, h, to be a CNN that takes a small, contiguous grid of points as context as opposed to a single point. This may allow the model to exploit local spatial correlations in the map, albeit at the expense of the flexibility of the context encoding scheme. Other possible extensions include investigating the performance of map predictions using other forms of perception, such as images from cameras.

REFERENCES

[1] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *Proc. IEEE Conf. on Robotics and Automation*, 2007.

[2] M. Pivtoraiko, D. Mellinger, and V. Kumar, "Incremental micro-uav motion planning for exploring unknown environments," in *Proc. IEEE Conf. on Robotics and Automation*, 2013.

[3] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Proc. IEEE Conf. on Robotics and Automation*, 2016.

[4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation*, 1997.

[5] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[6] G. J. Stein, C. Bradley, and N. Roy, "Learning over subgoals for efficient navigation of structured, unknown environments," in *Conf. on Robot Learning*, 2018.

[7] J. A. Caley, N. R. J. Lawrance, and G. A. Hollinger, "Deep learning of structured environments for robot search," *Autonomous Robots*, vol. 43, no. 7, pp. 1695–1714, 2019.

[8] A. Aydemir, P. Jensfelt, and J. Folkesson, "What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2012.

[9] H. J. Chang, C. S. G. Lee, Y. Lu, and Y. C. Hu, "P-slam: Simultaneous localization and mapping with environmental-structure prediction," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 281–293, 2007.

[10] D. P. Ström, F. Nenci, and C. Stachniss, "Predictive exploration considering previously mapped environments," in *Proc. IEEE Conf. on Robotics and Automation*, 2015.

[11] G. Liu, F. A. Reda, K. J. Shih, T. C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *European Conf. on Computer Vision*, 2018.

[12] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huan, "Generative image inpainting with contextual attention," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[13] R. Shrestha, F. P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in *Proc. IEEE Conf. on Robotics and Automation*, 2019.

[14] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager, "Uncertainty-aware occupancy map prediction using generative networks for robot navigation," in *Proc. IEEE Conf. on Robotics and Automation*, 2019.

[15] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.

[16] C. Richter, W. Vega-Brown, and N. Roy, "Bayesian learning for safe high-speed navigation in unknown environments," in *Int. Journal of Robotics Research*, 2018.

[17] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," in *Robotics: Science and Systems*, 2017.

[18] L. Janson, T. Hu, and M. Pavone, "Safe motion planning in unknown environments: Optimality benchmarks and tractable policies," in *Robotics: Science and Systems*, 2018.

[19] T. Fraichard and H. Asama, "Inevitable collision states – a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[20] M. Garnelo, C. M. Rosenbaum, D., T. Ramalho, S. M. Saxton, D., Y. W. Teh, D. Rezende, and S. M. A. Eslami, "Conditional neural processes," in *Int. Conf. on Machine Learning*, 2018.

[21] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast Marching Tree: a fast marching sampling-based method for optimal motion planning in many dimensions," *Int. Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[22] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. IEEE Conf. on Decision and Control*, 2015.

[23] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *Int. Journal of Control*, vol. 87, no. 6, pp. 1297–1311, 2014.

[24] M. Abadi *et al.* (2015) TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[25] Mosek APS. The MOSEK optimization software. Available at http://www.mosek.com.

[26] M. Udell, K. Mohan, D. Zeng, J. Hong, S. Diamond, and S. Boyd, "Convex optimization in Julia," in *High Performance Technical Computing in Dynamic Languages*, 2014.

[27] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[28] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D semantic parsing of large-scale indoor spaces," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.