Avoidance Critical Probabilistic Roadmaps for Motion Planning in Dynamic Environments

Felipe Felix Arias[1], Brian Ichter[2], Aleksandra Faust[2] and Nancy M. Amato[1]

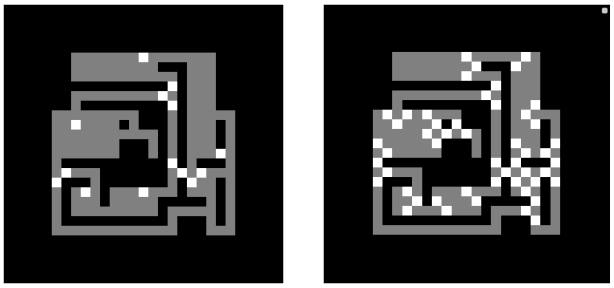
***Abstract*— Motion planning among dynamic obstacles is an essential capability towards navigation in the real-world. Sampling-based motion planning algorithms find solutions by approximating the robot's configuration space through a graph representation, predicting or computing obstacles' trajectories, and finding feasible paths via a pathfinding algorithm. In this work, we seek to improve the performance of these subproblems by identifying regions critical to dynamic environment navigation and leveraging them to construct sparse probabilistic roadmaps. Motion planning and pathfinding algorithms should allow robots to prevent encounters with obstacles, irrespective of their trajectories, by being conscious of spatial context cues such as the location of chokepoints (e.g., doorways). Thus, we propose a self-supervised methodology for learning to identify regions frequently used for obstacle avoidance from local environment features. As an application of this concept, we leverage a neural network to generate hierarchical probabilistic roadmaps termed Avoidance Critical Probabilistic Roadmaps (ACPRM). These roadmaps contain motion structures that enable efficient obstacle avoidance, reduce the search and planning space, and increase a roadmap's reusability and coverage. ACPRMs are demonstrated to achieve up to five orders of magnitude improvement over grid-sampling in the multi-agent setting and up to ten orders of magnitude over a competitive baseline in the multi-query setting.**

I. Introduction

In robotics, dynamic environment navigation refers to the set of processes by which a robot computes the trajectory it will take to arrive at its goal location without colliding with moving obstacles (e.g., people or other robots). Motion planning is one of the components of dynamic environment navigation. It computes a feasible and collision-free path given a state representation and a set of obstacles [1]. Naturally, Multi-Agent Motion Planning (MAMP) finds feasible paths for two or more robots and avoids inter-robot collisions. Due to the complexity of MAMP, most existing work on multi-agent systems solves the subproblem of Multi-Agent Pathfinding (MAPF), which requires a representation of the robot and its environment to solve a graph search problem in discrete space [2]. Sampling-based motion planning (SBMP) approaches, such as Probabilistic Roadmaps (PRMs) [3], reduce motion planning to pathfinding by constructing an implicit graph representation of a robot's configuration space through samples and local connections verified by a collision checking algorithm [1].

[1]Felipe Felix Arias and Nancy M. Amato are with Parasol Lab, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL USA {felipea2, namato}@illinois.edu

[2]Brian Ichter and Aleksandra Faust are with Google Research, Mountain View, CA, USA {ichter, faust}@google.com



a) Criticality Data b) Learned Criticality

Fig. 1: Environment annotated with a) avoidance criticality from 236 random dynamic environment problems and b) a model trained to identify critical regions. White cells are avoidance critical regions, gray cells are free space, and black cells are static obstacles.

Most sampling-based MAMP solvers use standard sampling techniques that do not account for workspace topology or obstacle avoidance, with grid-sampling [4] being the most prevalent. This oversight leads to dense roadmaps that are not suitable for multi-agent motion planning in narrow passages. As the number of agents and size of the environment increase, the computational costs of collision checking and pathfinding become prohibitive. Specifically, we note that searching through a roadmap with many vertices and edges while tracking inter-agent conflicts is the bottle-neck in sampling-based MAMP. Thus, we propose to improve the existing planning pipeline by constructing sparse roadmaps with sufficient structure to support multiple robots. By building upon ideas presented in [5], we train a model to identify regions important to MAMP. These regions include configurations critical to solution trajectories in static environments (e.g., doorways) and configurations that help construct larger motion structures for obstacle avoidance.

We show that the configurations that robots often use to wait in place, a behavior which indicates that the agent is actively avoiding collisions [6], capture a sizeable subset of the regions critical to MAMP. By extending *betweenness centrality* [7], a graph-theoretic measure of how frequently a vertex is visited in shortest paths through a graph, to dynamic environments, agents can generate training data for a model that determines whether or not a configuration is in a region critical to obstacle avoidance from a local occupancy grid. Finally, we leverage our model to sample in relevant regions and construct a learned skeleton of workspace that provides sufficient structure for obstacle avoidance.

A. Statement of Contributions

1) We formalize the concept of determining whether a configuration is useful for dynamic environment navigation from spatial context cues. Cognitive agents navigating in human environments should assess the likelihood of encountering a human or a robot. Therefore, we broadly propose that robots should be able to identify configurations that minimize collision likelihood irrespective of the dynamic obstacles' trajectories.
2) We propose a self-supervised methodology for identifying regions critical to obstacle avoidance. Our approach extends graph-theoretic concepts to generate training data for a binary classification neural network. Our data generation scheme tracks how long the agent had to wait in place. This metric captures a sizeable subset of the regions critical to obstacle avoidance and is used as a supervision signal to train the network to identify them from local topology.
3) We propose a hierarchical probabilistic roadmap strategy that pre-samples in avoidance critical regions. The roadmaps generated by this strategy contain emergent motion structures that enable efficient obstacle avoidance, reduce the search and planning space, and increase a roadmap's reusability and coverage of the environment.
4) Lastly, we demonstrate and discuss the benefits of identifying and utilizing avoidance critical configurations and show that ACPRMs outperform baselines in multi-agent and multi-query problems.

II. Related Work

A. Multi-Agent Motion Planning

Solutions to multi-agent navigation have framed the problem in a variety of ways. [8], [9], and [10], propose to compute single-agent paths and then iteratively resolve conflicts until all paths are valid. [11] solves the problem in a decentralized fashion by implicitly assuming that all agents have the same collision-avoidance behavior. Given the predicted trajectories of the dynamic obstacles, pathfinding algorithms like [12] find collision-free paths by adding the time dimension to their search space. However, most of these approaches work in discrete state spaces that may: become prohibitively large for robots with many degrees of freedom, lack support for heterogeneous robot systems, or fail to provide a framework for collision checking and navigation in three dimensions. Since SBMP is known to overcome some of these limitations, adaptations like [13] extend Multi-Agent Path Finding (MAPF) algorithms to MAMP methods that use probabilistic roadmaps. [14] and [15] generate one roadmap per agent and track constraints beyond the scope of this work in graph data structures but fail to address sampling efficiency in large environments with narrow passages. While [16] and [17] generate a single roadmap for all agents, their contributions focus on stitching together roadmaps through sensing and planning in stochastic maps under uncertainty, respectively. Comparatively, we construct a single multi-query roadmap for all agents that aims to reduce the search and planning space and increase the roadmap's reusability and coverage. [18] has a similar objective but proposes a post-processing procedure that optimizes the positions of the vertices and edges to create a directed graph well-suited for MAMP. While this is a promising approach, it does not improve sampling complexity as it requires an existing roadmap as input and keeps all of its vertices and edges. Instead, we propose to bias sampling to construct roadmaps with the desired qualities.

Fig. 2: Randomly chosen avoidance critical occupancy grids. From darkest to lightest the cells represent: static obstacles, free space, the agents' region of origin within the occupancy grid, the avoidance critical region (center of the grid), and the agent's goal region within the occupancy grid.

B. Sampling in Regions Critical to Motion Planning

In this work, we refer to spatial contextual awareness (SCA) [19] as the information inferred from the relationships between the robot, the structure of its surroundings, and environment features (e.g., location and appearance of relevant regions). While SCA has countless implications for dynamic environment navigation, we concentrate on recognizing regions useful for obstacle avoidance from local environment features. To our knowledge, ACPRM is the first sampling-based motion planning method that explicitly samples in regions important to MAMP. Several approaches that seek to bias sampling use heuristics to sample near static obstacles [20], near narrow passages [21], or near both [22]. Others like [23] and [24] seek to use workspace decompositions to identify regions of interest and guide sampling. Recent advances also show that it is possible to learn distributions to sample from, as shown in [25], [26], and [27]. [25] and [27] use pre-computed trajectories to bias sampling towards regions that may lead to an optimal solution. However, all of these solutions only consider the static-environment single-query setting, which can leverage the start and goal configurations for sampling and benefit from constructing dense roadmaps. Instead, we focus on methods that construct sparse multi-query roadmaps that contain configurations that enable efficient obstacle avoidance, can be pre-computed without previous knowledge of the start and goal configurations, and scale well to large environments with narrow passages.

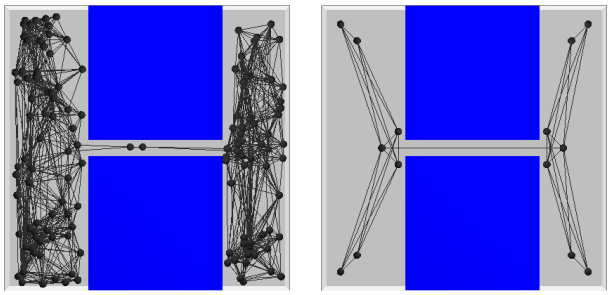
C. Learned Critical Probabilistic Roadmaps

Since we are not aware of any heuristics that can identify regions critical to obstacle avoidance, we use a framework similar to the one proposed in [5]. [5] solves many one-to-all problems in various roadmaps and tracks how often

Algorithm 1 ACPRM

Input: MAMP Problem (s_i, g_i $i \in \{0,..,N\}, h_\theta, k_a, k_p, PF$)

1: Discretize environment and generate occupancy grid inputs for h_θ
2: Identify avoidance critical regions using h_θ
3: Sample in avoidance critical regions
4: Connect critical samples to k_a nearest-neighbors
5: Sample uniformly in $\mathscr{C}_{space}$
6: Connect non-critical samples to k_p nearest-neighbors
7: Continue sampling and connecting until pathfinding algorithm, PF, can compute feasible paths for each agent



a) PRM b) ACPRM

Fig. 3: PRM (a) and ACPRM (b) used to compute motion plans for 4 agents in a narrow passage environment.

each vertex is visited. The approximated betweenness centrality values then become training labels for a regression neural network that predicts a *criticality* score. This model is then used to construct Critical PRMs, which identify a subset of the critical configurations through pre-sampling and attempt to connect all subsequent samples to them. There are two fundamental differences between Critical PRMs and our approach. First, we explicitly extend the notion of critical configurations to dynamic environment navigation. Second, rather than pre-sampling uniformly to find a subset of the critical configurations, we build learned skeletons that contain configurations in all critical regions. Notably, [5] demonstrates that neural networks can identify relevant regions from high-dimensional real-world environment local representations. Therefore, we focus on additional capabilities and consider grid-discretizations of the environments.

III. Problem Statement

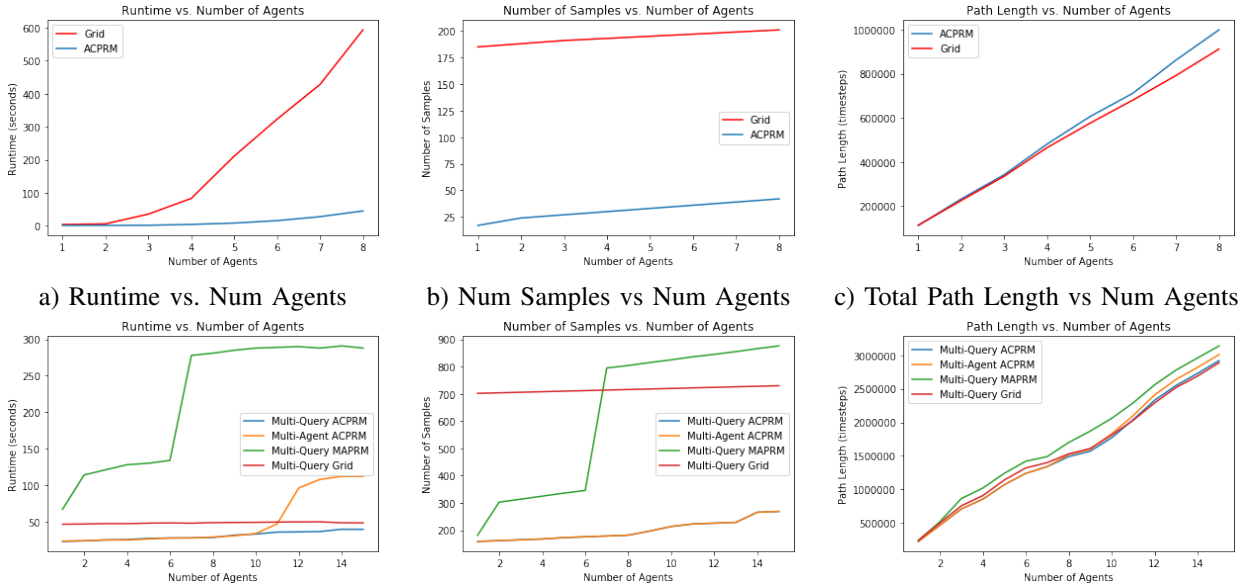
This work aims to solve the multi-agent motion planning problem efficiently by leveraging the ability to identify regions important for obstacle avoidance. We concentrate on three-dimensional planar navigation for omnidirectional robots with rigid bodies and a configuration space $\mathscr{C}_{space} = \mathbb{R}^2 \times \mathbb{S}$. Namely, the agents each have an x,y translational component, and a θ rotational component. The multi-agent motion planning problem with N such agents is defined as follows. Let $\mathscr{C}_{obst}$ be the subset of the configuration space occupied by static obstacles, $\mathscr{C}_{free} = \mathscr{C}_{space} \backslash \mathscr{C}_{obst}$ the free space, and $s_i \in \mathscr{C}_{free}$ and $g_i \in \mathscr{C}_{free}$ the initial and goal configurations of the i-th agent, respectively. We call a path p_i *collision-free* if the robot never leaves $\mathscr{C}_{free}$ or collides with another agent when executing p_i. A path p_i is *feasible* if it is collision-free and it successfully takes the i-th agent from s_i to g_i. We utilize sampling-based motion planning techniques that reduce motion planning to pathfinding by constructing an implicit representation of the configuration space through a set of samples and local connections that are verified by a collision-checking algorithm [1]. Given a roadmap that connects s_i to g_i $\forall i$, we compute feasible paths p_i $\forall i$ by using a MAPF algorithm.

IV. Developing Spatial Contextual Awareness

A. Identifying Avoidance Critical Regions

One of our goals is to define a methodology for identifying configurations critical to dynamic environment navigation. In this work, we extend the concept of betweenness centrality to dynamic environments by computing many one-to-one problems. For simplicity, we consider the case where the agent has prior knowledge of the moving obstacles' trajectories. However, this assumption is not imperative; an agent may actively predict the dynamic obstacles' trajectories during training or test time. In practice, we found that utilizing betweenness centrality (i.e., tracking the number of time steps an agent occupied a specific configuration) resulted in the obfuscation of regions critical to dynamic environment navigation by those critical to static-environment navigation. Therefore, our metric only increases when an agent is waiting in place and the dynamic obstacles are within a user-defined distance. This subtle contextual cue (the waiting in place event) is unique to dynamic environment navigation and implies that the agent is actively avoiding a collision. This metric, the amount of time an agent chose to wait at a configuration, is what we refer to as *avoidance criticality*. However, the nature of dynamic environment navigation leads to this being an ill-defined characterization of our method. There are two additional criteria:

- Avoidance criticality may be computed in environments with any number of obstacles, each with an arbitrary trajectory, size, speed, or behavior. In our data-generation procedure, we introduce three dynamic obstacles of the same size as the agent per navigation task, each of which takes the shortest path to get to its goal and moves with constant speed. One of the obstacles travels from the goal to the start of the agent's motion task, requiring the agent to avoid at least one collision. The other two obstacles are each assigned a random start and goal.
- Using pathfinding algorithms that provide no guarantees on the quality of the shortest path they find, such as A* [28], can result in unpredictable obstacle avoidance behavior. An agent waiting for a narrow passage to clear may wait at any configuration or compute a suboptimal trajectory to "waste time". For this reason, we use safe interval path planning (SIPP) [12], which



a) Runtime vs. Num Agents b) Num Samples vs Num Agents c) Total Path Length vs Num Agents

d) Runtime vs. Num Agents e) Num Samples vs Num Agents f) Total Path Length vs Num Agents

Fig. 4: Plots comparing grid-sampling and ACPRM in the three-way junction environment (a-c) and grid-sampling, ACPRM, and MAPRM in the building environment (d-f).

guarantees that the agent arrives at each configuration in the solution trajectory at the earliest time possible.

Fig. 1a shows an example of a randomly generated environment annotated with avoidance criticality.

B. Learning to Recognize Avoidance Critical Regions

We use avoidance criticality to get supervision signal for a neural network model. While betweenness centrality typically increases as it gets closer to local maxima, avoidance criticality does not display the same property. As seen in Fig. 1a, it instead consists of clusters of critical regions, with some being more critical than others due to workspace topology. As a result, we frame our problem as a classification task that bins avoidance criticality scores into a user-defined number of avoidance criticality levels. Here, we concentrate on the binary classification setting. In other words, although avoidance criticality refers to the number of timesteps agents spent waiting in a region, we choose to use our model to predict whether or not a region is likely to be used to wait in place for any amount of time.

1) Dataset Creation: We use self-supervision to train a neural network h_θ to determine whether a region is avoidance critical or not. Section IV-A outlines the procedure for annotating an arbitrary environment. However, to generalize to a wide range of workspace topologies, the training data for h_θ is collected from various environments. We generate a dataset Y of criticality label y and occupancy grid $g(x)$ pairs. Where $y \in [0,1]$ indicates whether or not a sample is avoidance critical, $x \in \mathscr{C}_{free}$ is a configuration, and $g(x)$ is the occupancy grid representation of the workspace around x. Fig. 2 shows randomly selected avoidance critical samples from the training data. In the interest of planar navigation in human environments, we use a random environment generator that constructs structured environments that consist of grid-aligned rooms and hallways of diverse shapes and sizes [29]. Figs. 1a-1b depict one such environment.

2) Learning: Using Y, we train h_θ. Which outputs $\hat{y} \in [0,1]$, an indicator of whether or not the model believes a given occupancy grid contains an avoidance critical region in its center. During training, we minimize cross-entropy loss to learn θ and use Stochastic Gradient Descent (SGD) with momentum to optimize. Namely, we minimize:

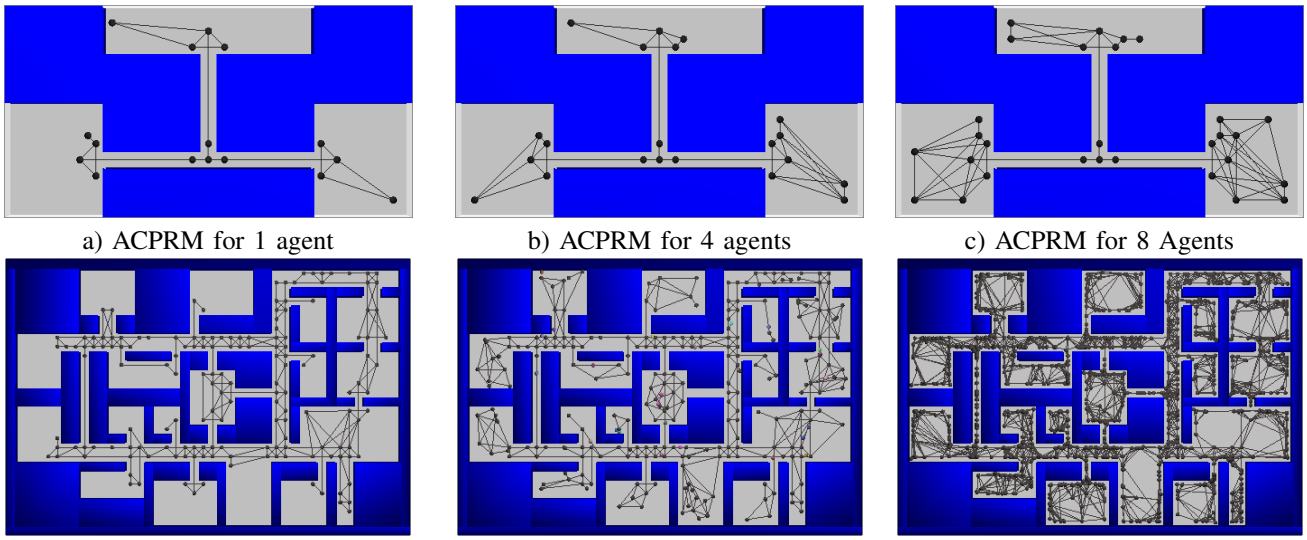
$$\mathscr{L}(\theta) = -y\log(h_\theta(g(x))) - (1-y)\log(1-h_\theta(g(x))) \quad (1)$$

Fig. 1b shows an example of a randomly generated environment annotated by h_θ through a sliding window procedure.

3) Implementation: h_θ is a fully connected neural network with 7 hidden layers, each with a ReLU activation function and a subsequent dropout layer set to 0.1. We use SGD with a learning rate of 0.01 and a momentum of 0.9 to optimize. The training data is collected from 450 randomly generated environments. For each region, we store its avoidance criticality score and an 11×11 occupancy grid with the critical region in its center. Our dataset contains 70K critical and 70K non-critical examples.

V. Avoidance Critical Probabilistic Roadmaps

This section presents a hierarchical SBMP method we call Avoidance Critical Probabilistic Roadmap (ACPRM). Algorithm 1 outlines the entire procedure. First, we compute a discretized representation of the environment to generate the occupancy grid inputs for h_θ (Line 1). Then, ACPRMs build a learned avoidance critical skeleton by sampling in all regions h_θ deems critical through a sliding window procedure. Specifically, ACPRMs sample in avoidance critical regions (Line 2-3) and then connect all samples to their k_a



a) ACPRM for 1 agent b) ACPRM for 4 agents c) ACPRM for 8 Agents

d) Learned Avoidance Critical Skeleton e) ACPRM f) MAPRM

Fig. 5: Relevant roadmaps from the three-way junction and building environments.

nearest neighbors (Line 4). An example of a learned skeleton is illustrated in Fig. 5d. Such skeletons provide motion structures and samples critical to the narrow passage problem. Although any sampling strategy may be used after generating the skeleton, we use a PRM scheme [3] that uniformly samples configurations (Line 5) and connects each sample to its k_p nearest neighbors (Line 6), which may or may not be part of the learned skeleton. After constructing the learned skeleton, ACPRMs iteratively connect each agent's start and goal configurations to the roadmap, sampling uniformly if necessary. Figs. 5a-5c depict the evolution of an ACPRM as the number of agents increases. Once the roadmap connects all of the start and goal configurations, ACPRM uses an arbitrary multi-agent pathfinding algorithm to find feasible paths for each agent (Line 7). Fig. 3 shows a qualitative comparison of two multi-agent roadmaps, where the only difference in the sampling procedure is the construction of the learned skeleton before uniform sampling.

VI. Results

A. Pathfinding in ACPRMs

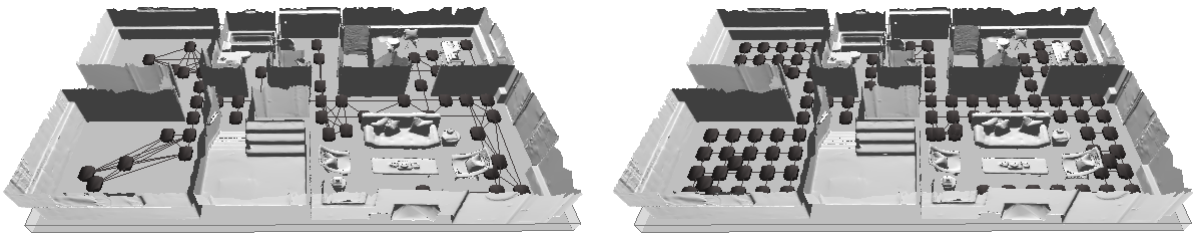
While ACPRMs will find a solution as the number of samples and connections (k_p) approach infinity, Algorithm 1 assumes access to a pathfinding algorithm that can track inter-agent collisions in the roadmap at an arbitrary resolution. In our implementation, we use a custom-made decentralized pathfinding algorithm based on SIPP [12] that allows robots to travel through edges but only wait at vertices. While our implementation is not guaranteed to find the globally optimal solution to the multi-agent motion planning problem, it finds the shortest feasible path for each agent given the obstacles already in the environment. We chose to implement a custom pathfinding algorithm because existing pathfinding algorithms for obstacle avoidance only scale well in discrete environments (i.e., planar graphs). Multi-agent pathfinding in roadmaps is not trivial as roadmaps have properties such as redundant edges (e.g., three or more vertices that lie on a line all connected to one another) and vertices that collide with one another. Since these properties vastly increase the search and planning space, developing multi-agent pathfinding algorithms that perform well on roadmaps is an open research area.

B. Motion Planning with ACPRMs

In this section, we seek solutions to the difficult problem of constructing a multi-query roadmap that scales well in large environments with narrow passages. Therefore, many standard baselines that sample uniformly [3] or bias sampling towards obstacles [20] could not find solutions without exceeding our computation time limit. Thus, we compare the performance of ACPRM with that of a grid-sampling baseline. Additionally, since ACPRMs should be useful for an arbitrary number of agents, we also compare its performance on multi-query single-agent motion planning against that of grid-sampling and Medial-Axis PRM [30], a method that retracts samples towards the medial-axis of $\mathscr{C}_{free}$ to find narrow passages and improve coverage of the environment.

C. Three-Way Junction Multi-Agent Problem

For our first multi-agent motion planning problem, we plan in the three-way junction environment shown in Figs. 5a-5c. This type of environment is known to be difficult for baselines and accentuates some of the features of ACPRMs. As depicted in Fig. 4a, ACPRM outperforms grid-sampling and has runtime improvement that increases as the number of agents increases. After eight agents, the uniform grid-sampling baseline exceeded our computation time limit of ten minutes. On the final eight-agent problem, ACPRM found a solution over six times faster. Moreover, Figures 4b and 4c show that ACPRM finds only marginally longer paths with over eight times fewer samples. Both methods



a) ACPRM b) Grid-Sampling

Fig. 6: Roadmaps in Gibson Environment

perform similarly when the number of agents is small, which illustrates that the improvements stem from the considerably smaller search space. Figs. 5a-5c depict the evolution of the ACPRM as the number of agents increases. In Fig. 5a, the learned skeleton is visible. It includes configurations critical for motion planning in narrow passages (e.g., entrances and a sample at the junction itself) and emergent motion structures that allow agents to take turns going through the narrow passages efficiently (triangle-shaped subgraphs at entrances and samples around the center of the junction).

D. Multi-Query Building Problem

To demonstrate ACPRM's scalability and ability to generalize to environments unlike those in the training data, we deploy ACPRM on an environment based on an existing building (See Figs. 5d-5e). We conduct two types of experiments. The first set seeks to study the use of ACPRM as a multi-agent roadmap. In contrast, the second evaluates ACPRM's coverage of the environment and its merit as a multi-query roadmap for single-agent motion planning. All experiments solve the same 15 randomly generated motion planning problems.

1) ACPRM for Multi-Agent Motion Planning: Fig. 4d plots the performance of an ACPRM used for MAMP (Multi-Agent ACPRM) and that of an ACPRM used to solve multiple queries (Multi-Query ACPRM) in the building environment. Both create the same roadmap and, therefore, have the same sampling complexity (Fig. 4e). However, after introducing ten agents to the environment, Multi-Query ACPRM begins to outperform Multi-Agent ACPRM by up to three orders of magnitude. These findings are expected, as planning for multiple agents requires adding the time dimension to the search space. Nonetheless, they shed some key insights. First, they illustrate the cost of pathfinding and the importance of keeping the search space small. Since the roadmaps are identical, the additional computation time stems from multi-agent pathfinding. Second, the increase in the total path length (Fig. 4f) succinctly shows that collision avoidance in this environment requires agents to wait or find paths longer than those found in the multi-query setting.

2) ACPRM for Single-Agent Motion Planning: We also compare ACPRM, grid-sampling, and MAPRM in the single-agent multi-query setting. Fig. 4d shows that ACPRM outperforms both baselines. ACPRM finds solutions up to ten times faster than MAPRM and up to two times faster than grid-sampling. These findings and the inability of other baselines to find solutions, substantiate the merits of using neural networks to bias sampling and construct sparse roadmaps. It is computationally inexpensive to annotate large environments. On the other hand, methods like MAPRM use sampling procedures that do not scale well in large and complex environments. Furthermore, as seen in Figs. 4e-4f, ACPRM uses up to three times fewer samples than MAPRM and grid-sampling but still finds shorter and roughly equivalent paths, respectively. Figs. 5d-5e, depict the learned skeleton and the roadmaps generated by both methods.

E. Real-World Environment

Lastly, we deploy an ACPRM (Fig. 6a) and the grid-sampling baseline (Fig. 6b) in an environment from the Gibson Database of 3D Spaces [31], a dataset generated by scanning indoor environments. Additionally, we use a model for the omnidirectional Clearpath Ridgeback mobile base. The environment is sufficiently small for grid-sampling to find a solution for eight agents moving in the environment simultaneously in six seconds. ACPRM finds a solution in three seconds and produces marginally shorter paths. In these experiments, both grid-sampling and ACPRM are deterministic, as they cover workspace such that there is no need for uniform sampling, agents need only connect their start and goal configurations to the roadmap. The random generation of starts and goals may make the problem infeasible or trivial, so we present the results of the first randomly generated problem that required obstacle avoidance. This experiment shows that if the training data is collected in appropriate environments and the discretization is adequate, ACPRMs are useful for motion planning in the real-world.

VII. CONCLUSIONS

We presented a method for identifying and leveraging regions critical to obstacle avoidance. Using an agent waiting in place for obstacles to pass as a contextual cue that signifies a configuration's importance to obstacle avoidance, we train a neural network to identify regions important to dynamic environment navigation from local environment features. We also introduced ACPRMs, which use the trained neural network to construct sparse probabilistic roadmaps with sufficient structure to support multi-agent motion planning in environments with narrow passages. Lastly, we demonstrated that ACPRMs enable efficient obstacle avoidance, reduce the search space, and increase a roadmap's coverage.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[2] Q. Sajid, R. Luna, and K. E. Bekris, "Multi-agent pathfinding with simultaneous execution of single-agent primitives," in *Fifth Symposium on Combinatorial Search (SoCS)*, 2012, pp. 19–21.

[3] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[4] K. Kondo, "Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free space enumeration," *IEEE Trans. Robot. Automat.*, vol. 7, no. 3, pp. 267–277, 1991.

[5] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9535–9541.

[6] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1610–1616.

[7] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[8] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[9] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2001, pp. 271–276.

[10] J. Abbenseth, F. G. Lopez, C. Henkel, and S. Dörr, "Cloud-based cooperative navigation for mobile service robots in dynamic industrial environments," in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 283–288.

[11] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2008, pp. 1928–1935.

[12] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5628–5635.

[13] I. Solis, R. Sandström, J. Motes, and N. M. Amato, "Roadmap-optimal multi-robot motion planning using conflict-based search," *arXiv preprint arXiv:1909.13352*, 2019.

[14] B. Brüggemann and D. Schulz, "Coordinated navigation of multi-robot systems with binary constraints," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 3854–3859.

[15] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap composition for multi-arm systems path planning," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 2471–2476.

[16] Z. Yao and K. Gupta, "Distributed roadmaps for robot navigation in sensor networks," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 997–1004, 2011.

[17] S. Kumar and S. Chakravorty, "Multi-agent generalized probabilistic roadmaps: Magprm," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3747–3753.

[18] C. Henkel and M. Toussaint, "Optimized directed roadmap graph for multi-agent path finding using stochastic gradient descent," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 776–783.

[19] C. Freksa, A. Klippel, and S. Winter, "A cognitive perspective on spatial context," in *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum fr Informatik, 2007.

[20] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: an obstacle-based PRM for 3d workspaces," in *Proceedings of the third Workshop on the Algorithmic Foundations of Robotics*. Natick, MA, USA: A. K. Peters, Ltd., 1998, pp. 155–168, (WAFR '98).

[21] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 141–153.

[22] D. Hsu, G. Sánchez-Ante, and Z. Sun, "Hybrid PRM sampling with a cost-sensitive adaptive strategy," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*. Barcelona, Spain: IEEE, 2005, pp. 3874–3880.

[23] H. Kurniawati and D. Hsu, "Workspace importance sampling for probabilistic roadmap planning," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, vol. 2, sept.-2 oct. 2004, pp. 1618–1623.

[24] J. Berg and M. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2004, pp. 453–460.

[25] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.

[26] M. Zucker, J. Kuffner, and J. A. Bagnell, "Adaptive workspace biasing for sampling-based planners," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 3757–3762.

[27] C. Zhang, J. Huh, and D. D. Lee, "Learning implicit sampling distributions for motion planning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3654–3661.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[29] Drow, "Random dungeon generator," Donjon, http://donjon.bin.sh/.

[30] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2, 1999, pp. 1024–1031.

[31] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: Real-world perception for embodied agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9068–9079.