2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

# PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-based Planning

Aleksandra Faust[1] Oscar Ramirez[1] Marek Fiser[1]
Kenneth Oslund[1] Anthony Francis[1] James Davidson[1]
Lydia Tapia[2]

***Abstract*— We present PRM-RL, a hierarchical method for long-range navigation task completion that combines sampling-based path planning with reinforcement learning (RL). The RL agents learn short-range, point-to-point navigation policies that capture robot dynamics and task constraints without knowledge of the large-scale topology. Next, the sampling-based planners provide roadmaps which connect robot configurations that can be successfully navigated by the RL agent. The same RL agents are used to control the robot under the direction of the planning, enabling long-range navigation. We use the Probabilistic Roadmaps (PRMs) for the sampling-based planner. The RL agents are constructed using feature-based and deep neural net policies in continuous state and action spaces. We evaluate PRM-RL, both in simulation and on-robot, on two navigation tasks with non-trivial robot dynamics: end-to-end differential drive indoor navigation in office environments, and aerial cargo delivery in urban environments with load displacement constraints. Our results show improvement in task completion over both RL agents on their own and traditional sampling-based planners. In the indoor navigation task, PRM-RL successfully completes up to $215\,\mathrm{m}$ long trajectories under noisy sensor conditions, and the aerial cargo delivery completes flights over $1000\,\mathrm{m}$ without violating the task constraints in an environment 63 million times larger than used in training.**

## I. Introduction

Long-range navigation tasks require robots to move safely over substantial distances while satisfying task constraints. For example, indoor navigation (Fig. 1a) requires a robot to navigate through buildings avoiding obstacles using noisy sensor data. As another example, an aerial cargo delivery [9] requires a suspended load-equipped unmanned aerial vehicle (UAV) to fly over long distances while avoiding obstacles and minimizing the oscillations of the load (Fig. 1b). We factor the long-range navigation task into two parts: long-range collision-free path finding and local robot control. Collision-free path finding identifies an obstacle-free path from a starting position to a distant goal while avoiding obstacles [18]. Local robot control produces feasible controls that the robot executes to perform the task while both satisfying task constraints and staying near the obstacle-free path.

Sampling-based planners, such as Probabilistic Roadmaps (PRMs) [15] and Rapidly Exploring Random Trees (RRTs) [17], [19], efficiently solve this problem by approximating the topology of the configuration space (C-space), the space of all possible robot configurations. These methods construct a graph or tree by sampling points in C-space, and connecting points if there is a collision-free local path between points. Typically this local path is created by a line of sight test or an inexpensive local planner.

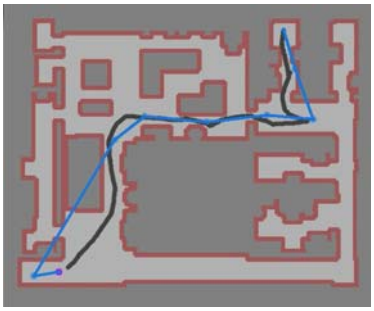
(a) Indoor navigation (b) Aerial cargo delivery

Fig. 1. Long-range navigation task case studies.

Regardless of how a collision-free path is generated, executing it introduces new complications. A robot cannot simply follow the C-space path, but must 1) satisfy the constraints of the task, 2) handle changes in the environment, and 3) compensate for sensor noise, measurement errors, and unmodeled system dynamics. Reinforcement learning (RL) agents have emerged as a viable approach to robot control [16]. RL agents have solved complex robot control problems [34], adapted to new environments [9], demonstrated robustness to noise and errors [8], and even learned complex skills [28]; however, RL agents can be hard to train if rewards are sparse [7]. This presents both opportunities and challenges in applying RL to navigation. RL agents have been successfully applied to several permutations of the navigation task and video games [27], making them good choices to deal with task constraints. Conversely, long-range navigation over complex maps has sparse rewards, making agents either difficult to train or successful only at short range because of vulnerabilities to local minima. For example, both city maps and office floorplans frequently have goals on the other side of wide barriers, which can cause local agents to become confused, or on the other side of box canyons, where local agents can become trapped.

1 Google Brain, Mountain View, CA, faust@google.com
2 Department of Computer Science, University of New Mexico, Albuquerque, NM

978-1-5386-3081-5/18/$31.00 ©2018 IEEE



(a) Training environment - 23 m by 18 m

(b) Building 1 - 183 m by 66 m

(c) Building 2 - 60 m by 47 m

(d) Building 3 - 134 m by 93 m

Fig. 2. The environments used for the indoor navigation tasks are derived from real building plans. a) The smallest environment is used to train the RL agent. b)-d) The PRMs are built using agents trained in the training environment. Red regions are deemed too close to obstacles and cause episode termination when the robot enters them; white is free space from where the start and goals are selected. Line-connected PRM waypoints (blue) and agent executed trajectory with RL agent (black).

We present PRM-RL, an approach to long-range navigation tasks which overcomes the limitations of PRMs and RL agents by using them to address each other's shortfalls. In PRM-RL, an RL agent is trained to execute a point-to-point task locally, learning the task constraints, system dynamics and sensor noise independent of the long-range environment structure. Then, PRM-RL builds a roadmap using this RL agent to determine connectivity, rather than the traditional collision-free straight-line interpolation in C-space. PRM-RL connects two configuration points only if the RL agent can consistently perform the local point-to-point task between them and all configurations along the produced trajectory are collision-free. The roadmap thus learns the long-range environment structure that can be navigated using that RL agent. Compared to roadmaps constructed based on pure C-space linear connectivity, PRM-RL roadmaps can obey robot dynamics and task constraints. The roadmap also learns to avoid local minima that cause failures of the RL agent. The resulting long-range navigation planner thus combines the planning efficiency of a PRM with the robustness of an RL agent, while avoiding local-minima traps, and providing resilience in the face of moderate changes to the environment.

To evaluate the approach, we focus on two problems: indoor navigation (Fig. 1a), and aerial cargo-delivery (Fig. 1b). The indoor navigation problem requires a differential drive robot to navigate inside buildings (Fig. 2) while avoiding static obstacles using only its LIDAR sensor (Fig. 3). We build PRMs from blueprints of target buildings, training RL agents on the smallest map using a simulator with noisy sensors and dynamics designed to emulate the unprocessed, noisy sensor input of the actual robot. We evaluate planning and execution of the PRMs in environments that the agent was not trained in. The aerial cargo delivery problem requires a quadrotor UAV with a suspended load to transport the cargo with minimum residual oscillations while maintaining load displacement below a given upper bound. The combined quadrotor-load system is non-linear and unstable. We evaluate the planner in a simulated urban environment and in two experimental environments, assessing adherence to task and dynamic constraints. We show that in environments with static obstacles, the planner maintains task constraints over long trajectories (over 1 km in length). Finally, results on physical robots for both problems show the PRM-RL approach maintains system dynamic constraints while producing feasible trajectories.

## II. Related Work

PRMs have been used in a wide variety of planning problems from robotics [13], [24] to molecular folding [3], [31], [33]. They have also been integrated with reinforcement

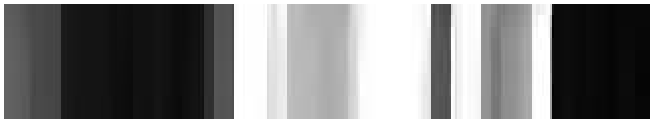


Fig. 3. Lidar observation that the differential drive robot uses for navigation. This observation corresponds to a hallway with a clear path ahead, and walls to the left and to the right. The white rays mean there are not objects within $5\,\mathrm{m}$, and black rays mean that there is an obstacle near. Notice the sensor noise in the center.

learning for state space dimensionality reduction [22], [31] by using PRM nodes as state space for the reinforcement learning agent. In contrast, our work applies reinforcement learning on the full state space as a local planner for PRMs. In prior work, for an aerial cargo delivery task, we trained RL agents to track paths generated from PRMs constructed using a straight line local planner [9]. In this work, the RL agents themselves act as the local planner, eliminating the need to train separate tracking agents. PRMs have also been modified to work with moving obstacles [14], [32], noisy sensors [23], and localization errors [2], [4]. Safety PRM [23] uses probabilistic collision checking with straight-line planner, associating with all nodes and edges a measure of potential collision. In our method, the RL local planner does Monte Carlo path rollouts with deterministic collision checking but noisy sensors and dynamics. We only add edges if the path can be consistently navigated.

Reinforcement learning has recently gained popularity in solving motion planning problems for systems with unknown dynamics [16], and has enabled robots to learn tasks that have been previously difficult or impossible [1], [6], [20]. Deep Deterministic Policy Gradient (DDPG) [21] is a current state-of-the-art algorithm that works with very high dimensional state and action spaces and is able to learn to control robots based on unprocessed sensor observations [20]. Continuous Action Fitted Value Iteration (CAFVI) [10] is a feature-based continuous state and action reinforcement algorithm that has been used for problems such as multi-robot tasks [10], flying inverted pendulums [11], and obstacle avoidance [8]. In this work, we use DDPG as the local planner for the indoor navigation task, and CAFVI as the planner for the aerial cargo delivery task. Model-predictive control (MPC) [12], action filtering [9], and hierarchical policy approximations [25], like RL, provide policies which respect robot dynamics and task constraints. However, they are computationally more expensive than RL at the execution time, making them not practical for building computationally-demanding roadmaps.

## III. Methods

PRM-RL works in three stages: RL agent training, roadmap creation, and roadmap querying. Fig. 4 shows the overview of the method. A key feature which makes PRM-RL transferable to new environments is that it learns task and system dynamics separately from the deployment environment. In the first stage, we train an RL agent to perform a task on an environment comparable to the deployment environment, but with a small state space to make learning more tractable. The RL agent training stage is a Monte Carlo simulation process: regardless of the learning algorithm used (DDPG, CAFVI or another), we train multiple policies and select the fittest one for the next stage of PRM-RL. This best policy, or value function, is then passed to the roadmap creation stage. The PRM builder learns a roadmap for a particular deployment environment using the best RL agent as a local planner using Algorithm 1. Constructed roadmap can be used for number of queries in the environment, as long as the same RL agent is used for execution. While the RL agent is robot/task dependent, it is environment independent, and it can be used to build roadmaps for a multitude of environments, as we show in the results.

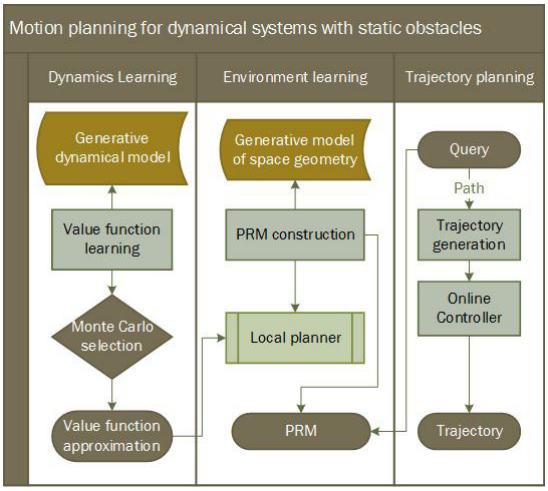



Fig. 4. PRM-RL flowchart.

### A. RL agent training

PRM-RL takes a start state $\boldsymbol{s}$ and a goal state $\boldsymbol{g}$ as two valid points in the robot's state space $S$. The robot state space $S$ is set of all possible robot observations in the control space for the robot, and is thus a superset of the configuration space, C-space. A state space point $\boldsymbol{s} \in S$ is valid if and only if it satisfies the task constraints for some predicate $L(\boldsymbol{s})$, and the point's projection onto C-space $p(\boldsymbol{s})$ belongs in C-free, a partition of C-space consisting of only collision-free points. When either of the conditions is violated, the system's safety cannot be guaranteed, and the task cannot be performed. The task is completed when the system is sufficiently close to the goal state, $\|p(\boldsymbol{s}) - p(\boldsymbol{g})\| \leq \epsilon$ in the configuration space. Our goal is to find a transfer function

$$\dot{\boldsymbol{s}} = f(\boldsymbol{s}, \boldsymbol{a}). \tag{1}$$

that leads the system to task completion. Formally, in the Markov Decision Process (MDP) setting, our goal is to find a policy $\pi : S \to A$, such that for an initial state $\boldsymbol{s}$, there is $n_0 > 0$, such that $\forall n \leq n_0\, \boldsymbol{s}_n = \pi^n(\boldsymbol{s})$ and $L(\boldsymbol{s}_n)$ holds, and $\|p(\boldsymbol{s}_{n_0}) - p(\boldsymbol{g})\| \leq \epsilon$.

The RL agent is trained to perform point to point navigation task without memory of the workspace topology. Reinforcement learning finds a solution to a discrete time

MDP with continuous multi-dimensional state and action spaces, given as a tuple of states, action, transitions, and rewards, $(S, A, P, R)$. $S \subset \mathbb{R}^{d_s}$ is the state or observation space of the robot. For the indoor navigation task, we use the combined space of the robot's position relative to the goal in polar coordinates, $\boldsymbol{g}$, and all possible LIDAR observations (Fig. 3), $\boldsymbol{o}$, casting 64 rays over $220^\circ$ field-of-view with up to $5\,\mathrm{m}$ depth. Overall, the state is $\boldsymbol{s} = (\boldsymbol{g}, \boldsymbol{o}) \in \mathbb{R}^{66}$. In the case of the aerial cargo delivery the state space is the joint position-velocity vector of the quadrotor's and load's centers of the masses, $\boldsymbol{s} = [\boldsymbol{s}_p\, \boldsymbol{s}_v\, \boldsymbol{\eta}\, \dot{\boldsymbol{\eta}}] \in \mathbb{R}^{10}$, where $\boldsymbol{s}_p = [x\, y\, z]^T$ is the center of the mass of the UAV, its linear velocities are $\boldsymbol{s}_v = [\dot{x}\, \dot{y}\, \dot{z}]^T$, and the angular position $\boldsymbol{\eta} = [\psi\, \phi]^T$ of the suspended load in the spherical coordinates system originating at the quadrotor's center of mass, and its angular velocities $\dot{\boldsymbol{\eta}} = [\dot{\psi}\, \dot{\phi}]^T$.

The action space, $A \subset \mathbb{R}^{d_a}$ is the space of all possible actions that the robot can perform. For the indoor navigation the action space is a two-dimensional vector of wheel speeds, $\boldsymbol{a} = (v_l, v_r) \in \mathbb{R}^2$, while for the aerial cargo delivery the action space is an acceleration vector applied to the quadrotor's center of the mass $\boldsymbol{a} \in \mathbb{R}^3$.

The transition probability, $P : S \times A \to \mathbb{R}$ is a probability distribution over state and actions. Like many reinforcement learning systems, we assume a presence of a simplified black-box simulator without knowing the full non-linear system dynamics. The simulator for the indoor navigation task is a kinematics simulator operating at 5 Hz. To simulate imperfect real-world sensing, the simulator adds Gaussian noise, $\mathcal{N}(0, 0.1)$, to its observations. The simulator for the aerial cargo delivery task operates at 50 Hz because of the inherent instability of the quadrotor, and is a simplified model of the quadrotor-load model described in [9].

During training the agent observes a scalar reward, $R : S \to \mathbb{R}$, and learns a policy $\pi(\boldsymbol{s}) = \boldsymbol{a}$, that, given an observed state $\boldsymbol{s} \in S$, returns an action $\boldsymbol{a} \in A$ that the agent should perform to maximize long-term return, or value,

$$\pi^*(\boldsymbol{s}) = \underset{\boldsymbol{a} \in A}{\operatorname{argmax}}\, E\left(\sum_i \gamma^i R(\boldsymbol{s}_i)\right). \tag{2}$$

We reward the agent for reaching the goal, with task specific reward shaping terms for each of the two tasks. For the indoor navigation task, we reward the agent for staying away from obstacles, while for the aerial cargo delivery task we reward minimizing the load displacement.

We train the agent with a continuous action RL algorithm in a small environment. We choose continuous action RL because, although more difficult to train, they provide more precise control [10], [21], and faster policy evaluation time [10]. We train indoor navigation tasks with DDPG [21] in a small space (Fig. 2a). The aerial cargo delivery agent is trained with CAFVI [10] using only $1\,\mathrm{m}$ around to goal for the training. Trained once, the agents can be used in different environments to plan a trajectory by observing the world to receive an observation or state, evaluating the policy (2), and applying the recommended action to the state. Note that agent plans in the state space, which is generally higher dimensionality than the C-space.

### B. PRM construction

The basic PRM method works by performing a uniform random sampling of robot configurations in the the robot's configuration space, retaining only collision-free samples as nodes in the roadmap. PRMs then attempt to connect the samples to their nearest neighbors using a local planner. If there is an obstacle-free path between two nodes, the edge is added to the roadmap.

We modify the basic PRM by changing the way nodes are connected. Since, we are primarily interested in robustness to noise and adherence to the task, we only connect two configurations if the RL agent can consistently perform the point-to-point task between the two points. Because the state space $S$, is a superset of the C-space, we sample multiple variations around the start and goal configuration points, and add an edge only if the success-rate exceeds a threshold. Note that this means that PRM trajectories cannot guarantee to be collision-free paths. That is not a limitation of the method, but rather the results of sensor noise. Nevertheless, later when discussing the results, we estimate a lower bound on the probability of collision.

Algorithm 1 describes how PRM-RL adds edges to the PRMs. We sample multiple points from the state space, which correspond to the start and goal in the configuration space, and attempt to connect the two points. An attempt is successful only if the agent reaches sufficiently close to the goal point. To compute the total length of a trajectory, we sum the distances for all steps plus the remaining distance to the goal. The length we associate with the edge is the average of the distance of successful edges. The algorithm recommends adding the edge to the roadmap if the success rate is above a predetermined threshold. If too many unsuccessful trials are attempted, the method terminates. The number of collision checks in Algorithm 1 is $O(max_{steps} * num_{attempts})$, because there are multiple attempts for each edge. Each trial of checking the trajectory can be parallelized with $num_{attempts}$ processors.

### C. PRM-RL Querying

To generate long-range trajectories, we query a roadmap, which returns a list of waypoints. A higher-level planner then invokes a RL agent to produce a trajectory to the next waypoint. When the robot is within the waypoint's goal range, the higher-level planner changes the goal with the next waypoint in the list.

## IV. Results

In this Section we evaluate the performance of PRM-RL for the indoor navigation and aerial cargo delivery tasks.

### A. Indoor Navigation

We work with four maps depicted in Fig. 2. We train the RL agent to avoid the obstacles in a $14\,\mathrm{m}$ by $17\,\mathrm{m}$ environment (Fig. 2a) using the DDPG algorithm; our setup

follows [21] but with two hidden layers in our actor networks (34, 55), two joint hidden layers in the critic network (163, 33) with an additional hidden layer of (261) for states and a weight decay of 0.01, trained with batch size 124, with the Adam optimizer with alpha 0.9, beta 0.999, epsilon 1e-08, learning rate 7.37e-05 for the actor and 1.14e-04 for the critic, with a target network updated every 13 training steps, and with a replay buffer with 200K entries. When training and using the RL agent, the goal tolerance is $0.5\,\mathrm{m}$. This enables the RL agent to train in the presence of noisy sensors and dynamics and is necessary because the agent does not rely on external localization. Recall that the only input to the agent is position of the goal, and the noisy LIDAR data (Fig. 3). The evaluation environments (Fig. 2b, 2c, and 2d) are between 12 and 52 times larger than the training one.

We evaluate the PRM-RL by comparing them with PRMs built with a straight line planner (PRM-SL). We do not compare with RRTs because they are one-time planners and are prohibitively expensive for building on-the-fly. Each roadmap is evaluated on 100 queries selected from the C-free space. We examine 1) the cost of building the roadmaps; 2) the qualities of the planner trajectories; 3) the actual performance of the agent in simulation; and 4) the experimental results.

**Algorithm 1** PRM-RL Add edge

**Input:** $s,\, g \in C_{space}$: Start and goal.
**Input:** $p_{success} \in [0, 1]$ Success threshold.
**Input:** $num_{attempts}$: Number of attempts.
**Input:** $\epsilon$: Sufficient distance to the goal.
**Input:** $max_{steps}$: Maximum steps for trajectory.
**Input:** $L(\boldsymbol{s})$: Task predicate.
**Input:** $\pi$: RL agent's policy.
**Input:** $D$ Generative model of system dynamics.
**Output:** $add_{edge}, success_{rate}$ , $length$

$success \leftarrow 0$, $length \leftarrow 0$
$needed \leftarrow p_{success} * num_{attempts}$
**for** $i = 1, \cdots num_{attempts}$ /* Run in parallel.*/ **do**
  $\boldsymbol{s}_s \leftarrow s.SampleStateSpace()$ // Sample from the
  $\boldsymbol{s}_g \leftarrow g.SampleStateSpace()$ // state space
  $success_{rate} \leftarrow 0$, $steps \leftarrow 0$, $\boldsymbol{s} \leftarrow \boldsymbol{s}_s$
  $length_{trial} \leftarrow 0$
  **while** $L(\boldsymbol{s}) \wedge steps < max_{steps} \wedge \|p(\boldsymbol{s}) - p(\boldsymbol{s}_g)\| > \epsilon \wedge p(\boldsymbol{s}) \in$ C-free **do**
    $\boldsymbol{s}_p \leftarrow \boldsymbol{s}$, $\boldsymbol{a} \leftarrow \pi(\boldsymbol{s})$
    $\boldsymbol{s} \leftarrow D.predictState(\boldsymbol{s}, \boldsymbol{a})$
    $num_{steps} \leftarrow num_{steps} + 1$
    $length_{trial} \leftarrow length_{trial} + \|\boldsymbol{s} - \boldsymbol{s}_p\|$
  **end while**
  **if** $\|p(\boldsymbol{s}) - p(\boldsymbol{s}_g)\| < \epsilon$ **then**
    $success \leftarrow success + 1$
  **end if**
  **if** $needed > success \wedge i > needed$ **then**
    **return** False, 0, 0 // Not enough success, we can terminate.
  **end if**
  $length_{trial} \leftarrow length_{trial} + \|p(\boldsymbol{s}) - p(\boldsymbol{g})\|$
  $length \leftarrow length + length_{trial}$
**end for**
$length \leftarrow \frac{length}{success}$, $success_{rate} \leftarrow \frac{success}{i}$
**return** $success_{rate} > p_{success}, success_{rate}, length$

*1) Roadmap construction evaluation:* To build the PRMs, we use an $85\%$ success rate to connect the edges, over 20 trials. The PRMs attempts to connect all the nearest neighbors with within $10\,\mathrm{m}$ from a node. We construct roadmaps for three different node densities: $0.1$, $0.2$, and $0.4$ samples per meter squared.

Table I summarizes the roadmap characteristics. We examine the number of nodes in the roadmap, number of edges, and collision checks performed to build the roadmap, and include percent of success for 100 randomly generated queries. As expected, across all environments and roadmap construction methods, the higher sampling density produces larger maps and more successful queries. The number of nodes in the map does not depend on the local planner, but the number of edges and collision checks do. The number of collision checks is approximately between 10 and 20 times higher for the RL local planner using Algorithm 1, because we terminate collision checks early for the edges that consistently fail. The SL planner does not use noisy sensor observations, and therefore, requires a single trial to add or reject an edge. We observe that roadmaps built with the RL local planner are more densely connected with $15\%$ and $50\%$ more edges. This is expected because the RL agent is able to go around the corners and small obstacles where the straight line planner cannot.

*2) Expected trajectory characteristics:* Now we look at the expected performance of PRM-RL across the four environments with respect to 100 randomly selected queries. We focus only on the roadmaps with 0.4 samples per meters squared density. Table II summaries the expected number of waypoints, trajectory length, and duration. The SL local planner is more optimistic, expecting 100% success on the planner trajectories. The RL agent computes the expected probability of success as a joint probability of each edge, since each edge success is an independent random event. Thus, the lower bound on the expected success rate over multiple waypoints is $0.85^{n_w}$, where $n_w$ is number of waypoints. So, the lower bounds on trajectory success for Building 2 for example should be $0.85^{6.05} = 37\%$, while the lower bound for Building 3 is $0.85^{12.65} = 13\%$. Given that the expected success rates in Table II are above 90%, that means that most of the edges added to the roadmap had 95-100% success rate during the roadmap construction time. The actual success rates for the RL local planner are between the lower bounds. This RL agent does not require the robot to come to rest at the goal region, therefore the robot experiences some inertia when the waypoint is switched. This causes some of the failures. That said, the actual success rates for the PRM-SL are significantly lower. What's more so, the PRM-SL planner has no estimate of a path risk. We also see that the PRM-RL paths contain more waypoints, with the exception of Building 3. Building 3 is the largest, and the paths through it require

TABLE I

ROADMAP CONSTRUCTION SUMMARY DIFFERENT NODE SAMPLING DENSITIES (0.1, 0.2, AND 0.4 SAMPLES PER METER SQUARED). ENVIRONMENT, METHOD, SUCCESS RATE ON 100 QUERY EVALUATION, NUMBER OF NODES, NUMBER OF EDGES, NUMBER OF COLLISION CHECKS.

| Sampling density | Method | Query success rate (%) | | | Nodes | | | Edges | | | Collision Checks | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0.1 | 0.2 | 0.4 | 0.1 | 0.2 | 0.4 | 0.1 | 0.2 | 0.4 | 0.1 | 0.2 | 0.4 |
| Training | PRM-RL | 0.32 | 0.36 | 0.50 | 16 | 32 | 63 | 33 | 166 | 663 | 13009 | 38292 | 223898 |
| | PRM-SL | 0.06 | 0.09 | 0.29 | 16 | 32 | 63 | 15 | 123 | 464 | 914 | 3649 | 17264 |
| Building 1 | PRM-RL | 0.14 | 0.31 | 0.43 | 436 | 871 | 1741 | 3910 | 15632 | 59856 | 1476931 | 5755744 | 23303949 |
| | PRM-SL | 0.06 | 0.17 | 0.15 | 436 | 871 | 1741 | 3559 | 13937 | 52859 | 156641 | 622257 | 2393841 |
| Building 2 | PRM-RL | 0.17 | 0.22 | 0.38 | 116 | 232 | 463 | 403 | 1602 | 6833 | 294942 | 1174655 | 5218619 |
| | PRM-SL | 0.06 | 0.11 | 0.18 | 116 | 232 | 463 | 276 | 1190 | 5365 | 18297 | 72850 | 312859 |
| Building 3 | PRM-RL | 0.19 | 0.39 | 0.56 | 441 | 881 | 1761 | 2962 | 11850 | 45623 | 1152524 | 4492144 | 17947728 |
| | PRM-SL | 0.07 | 0.11 | 0.08 | 441 | 881 | 1761 | 1852 | 7570 | 30267 | 97088 | 375304 | 1493816 |

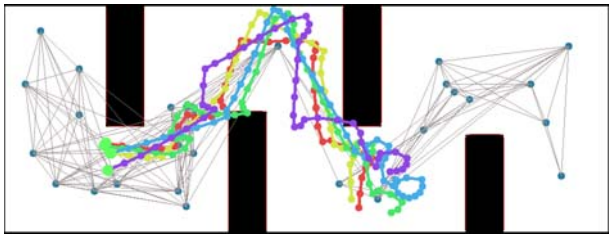


Fig. 5. Trajectories for a single query, executed on the real differential drive robot over five trails (green, blue, red, purple, and yellow). The grey straight lines indicate graph edge connectivity of the PRM-RL. The trajectories are captured with a mocap system.

more turns. The RL agent can execute some of the these turns without adding a waypoint. Expected trajectory length and duration are longer for the RL agent, because the agent uses more realistic estimates of what the robot can achieve.

*3) Actual trajectory characteristics:* To evaluate the actual indoor navigation task performance, we look at the query characteristics for successful versus unsuccessful queries. Table III summarizes the differences in number of waypoints, trajectory length, and duration of the successful and unsuccessful trajectories. The RL agent produces higher success rate than the straight line planner. PRM-RL performs the best in the Building 3, which is the largest, while PRM-SL performs the worst in that environment. The longest successfully executed trajectory is $216\,\mathrm{m}$ meters long, taking 400 seconds to complete, and passing through 45 waypoints (see enclosed video submission).

The successful trajectories have fewer waypoints than the expected waypoints from Table II, which means that the shorter queries are more likely to succeed, as is expected. We also see that the unsuccessful queries fail after only few waypoints. The PRM-RL has higher success rate overall and performs the best in the largest environment (Building 3), while the PRM-SL performs the worst in it. Fig. 2 depicts some of the PRM-RL trajectories the test environments.

*4) Physical robot experiments:* To test the effectiveness and transfer of our approach on a real robot, we created a simple slalom-like environment with four obstacles distributed over an $8\,\mathrm{m}$ by $3\,\mathrm{m}$ space. Fig. 5 illustrates a single query variance due to sensor noise. Each of the trial trajectories reached within the goal region of $0.5\,\mathrm{m}$ with a mean distance of $0.37\,\mathrm{m}$.

### B. Aerial Cargo Delivery

A model of a city, our simulation environment, allows us to test PRM-RL over longer distances, while experimental environments allow us verify the algorithm on a UAV quadrotor equipped with a suspended load. The city model (Fig. 6a), is $450\,\mathrm{m}$ by $700\,\mathrm{m}$, with $200\,\mathrm{m}$ height. Its bounding box is 250 million times larger than the quadotor's bounding box, and 63 million larger than the RL agent training environment. Requiring the load displacement to be under $45°$, the evaluation trajectories are result of 100 queries between a fixed origin (depot), and randomly selected goal. We measure load displacement, and trajectory duration, and compare to PRM-SL. The aerial cargo delivery tasks is deterministic, and we build roadmaps requiring one sampling trail, and 100% success rate. Additionally, this RL agent requires the robot to come at rest at waypoints, resulting in no failures due to inertia.

*1) Simulation results:* Fig. 6 shows the load displacement and trajectory duration results. The $xy$-plane contains the city projection. The red area marks the origin location. The data points' $x$ and $y$ coordinates are the queries' goal location in the city projected onto $xy$-plane. The data points' $z$-coordinates in Fig. 6b are the maximum load displacement throughout the trajectory, and in Fig. 6c the time it takes to complete the trajectory. Points represented as squares (green) are the result PRM-SL, and the triangle points (blue) are generated with PRM-RL. The longest trajectory executed is over 1 kilometer in length. The maximum load displacement in the continuous case is consistently below the load displacement exhibited with the discrete planner and stays under the required $45°$. Action filtering [9] guarantees load displacement constrains, but given that it is a discrete action planner with over 1.7 millions action, it adds 1.7 million collision checks per step in the local planner execution. This makes this method computationally prohibitive to build PRMs with. To offset the load displacement control, the PRM-RL trajectories take a longer time to complete the task. They generally contain on average 5 times more waypoints. Fig. 6c shows that in the vicinity of the origin, PRM-RL and PRM-SL trajectories have similar duration. As the goal's distance increases, the discrepancy becomes more significant.

*2) Experimental results:* The experiments evaluate PRM-RL on a physical robot to validate the simulation results. We look for discrepancies in the length, duration, and load

TABLE II

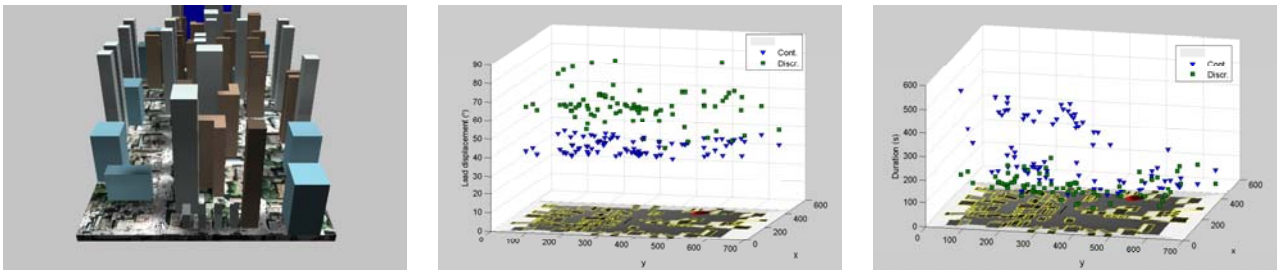
EXPECTED PATH AND TRAJECTORY CHARACTERISTICS OVER 100 QUERIES. ENVIRONMENT, METHOD, ACTUAL AND EXPECTED SUCCESS PERCENT, NUMBER OF WAYPOINTS IN THE PATH, EXPECTED TRAJECTORY LENGTH IN METERS, AND DURATION IN SECONDS.

| Environment | Method | Success (%) | | Number of waypoints | | Trajectory length (m) | | Duration (s) | |
|---|---|---|---|---|---|---|---|---|---|
| | | Actual | Expected | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Training | PRM-RL | 50 | 90 | 7.11 | 2.88 | 18.68 | 11.80 | 36.28 | 36.28 |
| | PRM-SL | 28 | 100 | 5.44 | 3.51 | 9.39 | 9.69 | 8.29 | 8.29 |
| Building 1 | PRM-RL | 43 | 91 | 12.09 | 10.68 | 56.88 | 63.37 | 107.78 | 107.78 |
| | PRM-SL | 15 | 100 | 11.99 | 8.88 | 46.69 | 43.85 | 43.07 | 43.07 |
| Building 2 | PRM-RL | 38 | 95 | 6.05 | 4.46 | 21.54 | 25.82 | 41.69 | 41.69 |
| | PRM-SL | 18 | 100 | 6.98 | 5.69 | 18.75 | 23.05 | 16.97 | 16.97 |
| Building 3 | PRM-RL | 56 | 92 | 12.62 | 5.12 | 64.94 | 33.96 | 122.31 | 122.31 |
| | PRM-SL | 8 | 100 | 15.58 | 8.02 | 59.03 | 35.47 | 54.00 | 54.00 |

TABLE III

CHARACTERISTICS OF THE SUCCESSFUL AND UNSUCCESSFUL TRAJECTORIES. ENVIRONMENT, METHOD, ACTUAL AND EXPECTED SUCCESS PERCENT, NUMBER OF WAYPOINTS IN THE PATH, EXPECTED TRAJECTORY LENGTH IN METERS, AND DURATION IN SECONDS.

| Environment | Method | Success (%) | Number of waypoints | | | | Trajectory length (m) | | | | Duration (s) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Successful | | All | | Successful | | All | | Successful | | All | |
| | | | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Training | PRM-RL | 50 | 4.29 | 2.38 | 4.32 | 2.88 | 13.88 | 8.62 | 14.54 | 9.46 | 36.74 | 26.06 | 37.04 | 37.04 |
| | PRM-SL | 28 | 2.45 | 3.02 | 2.49 | 2.73 | 6.71 | 7.10 | 6.39 | 6.32 | 6.71 | 7.10 | 6.39 | 6.39 |
| Building 1 | PRM-RL | 43 | 10.76 | 11.01 | 8.44 | 9.77 | 51.17 | 56.66 | 43.16 | 51.74 | 117.46 | 108.60 | 112.29 | 112.29 |
| | PRM-SL | 15 | 4.37 | 4.69 | 4.09 | 4.51 | 20.19 | 21.78 | 18.32 | 20.46 | 20.19 | 21.78 | 18.32 | 18.32 |
| Building 2 | PRM-RL | 38 | 8.08 | 3.67 | 3.96 | 4.53 | 32.93 | 21.58 | 23.64 | 20.78 | 70.18 | 45.87 | 77.33 | 77.33 |
| | PRM-SL | 18 | 3.89 | 5.63 | 3.41 | 4.71 | 15.31 | 20.79 | 12.29 | 17.20 | 15.31 | 20.79 | 12.29 | 12.29 |
| Building 3 | PRM-RL | 56 | 9.74 | 5.60 | 9.61 | 5.79 | 58.71 | 32.09 | 57.60 | 34.78 | 130.79 | 63.06 | 130.06 | 130.06 |
| | PRM-SL | 8 | 5.66 | 5.33 | 5.21 | 4.46 | 22.46 | 19.67 | 21.30 | 17.50 | 22.46 | 19.67 | 21.30 | 21.30 |




(a) City (b) Load displacement (c) Duration

Fig. 6. (a) Areal cargo delivery city environment. (b) Load displacement and (c) trajectory duration for PRM-RL (triangle) vs. PRM-SL (square) in the city projected on the $xy$-plane.

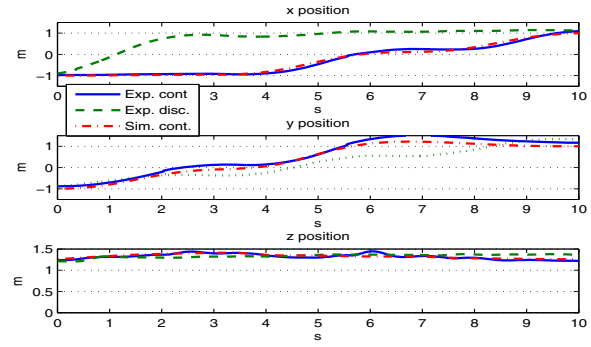
displacement over the entire trajectory. The experiments were performed on AscTec Hummingbird quadrotor, carrying a 62-centimeter suspended load weighing 45 grams, in a MARHES aerial testbed [29]. The experimental environments are $3\,\mathrm{m}$ by $4\,\mathrm{m}$ by $2\,\mathrm{m}$, and contain 3 and 5 obstacles, $2\,\mathrm{m}$ tall (see enclosed video attachment). The quadrotor and load position, tracked via a motion capture system at 100 Hz, require load displacement not to exceed $10°$.

Fig. 7 shows that the vehicle and load trajectories match closely between simulation and experiment, and that the load displacement stays under $10°$, even in experiments. Further, the discrepancy between simulation and experiments of $10°$ remains, due to the unmodelled load turbulence, is consistent with our previous results from [9].

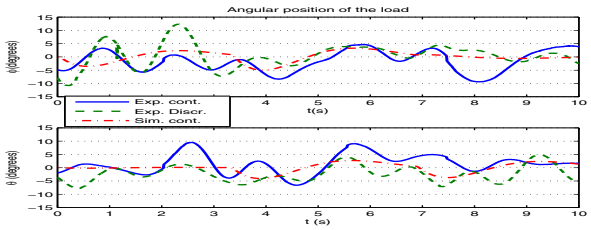
Next, we create a PRM-SL trajectory in the same environment. The vehicle trajectory differs in this case, because the PRMs in the two cases differ. The PRM-SL does not directly control the load displacement, while PRM-RL rejects edges that exceed the maximum allowed load displacement. The load trajectory (Fig. 7 (b)) indicates that the load displacement exceeds the $10°$ limit 2.5 seconds into the flight. The video attachment contains the experiment footage.

## V. CONCLUSIONS

We presented PRM-RL, a hierarchical planning method for long-range navigation tasks, that combines sampling-based path planning with RL agents to complete tasks in very large environments. Evaluated on two case studies, one with noisy sensor feedback task, and the other on a complex unstable dynamics, we showed that PRM-RL expands the capabilities of both RL agents and sampling-based planners. The indoor navigation task successfully completes trajectory over $210\,\mathrm{m}$ long, and the aerial cargo delivery creates flights over 1 kilometer long, in the planning space 63 million times larger than the agent's training space. Both tasks are verified experimentally on the physical robots.




(a)




(b)

Fig. 7. Experimental PRM-RL trajectory with three edges, on the quadrotor demonstrating quadrotor (a) and load (b) trajectories. PRM-RL with continuous action RL from a hardware experiment is shown in comparison to the simulated trajectory. Also shown for comparison is RL with discrete actions following the PRM-SL path from a hardware experiment.

## ACKNOWLEDGEMENTS

The authors thank Patricio Cruz (MARHES [26]) for helping with experiments, Torin Adamson (Tapia Lab [5]) for creating the virtual environments, and Parasol Lab [30] for the motion planning library. Tapia is supported in part by the National Science Foundation under Grant Numbers IIS-1528047 and IIS-1553266.

## REFERENCES

[1] D. Abel, A. Agarwal, F. Diaz, A. Krishnamurthy, and R. E. Schapire. Exploratory gradient boosting for reinforcement learning in complex domains. *arXiv preprint arXiv:1603.04119*, 2016.

[2] A. Agha-mohammadi, S. Chakravorty, and N. Amato. FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *Int. J. Robot. Res.*, 33(2):268–304, 2014.

[3] I. Al-Bluwi, T. Simon, and J. Corts. Motion planning algorithms for molecular simulations: A survey. *Computer Science Review*, 6(4):125 – 143, 2012.

[4] R. Alterovitz, T. Simeon, and K. Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. page 246 253, Atlanta, GA, USA, June 2007.

[5] Adaptive Motion Planning Research Group, Department of Computer Science, University of New Mexico. https://cs.unm.edu/amprg/, 2017.

[6] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

[7] A. Faust, H.-T. Chiang, and L. Tapia. Pearl: Preference appraisal reinforcement learning for motion planning. *Under submission*.

[8] A. Faust, N. Malone, and L. Tapia. Preference-balancing motion planning under stochastic disturbances. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3555–3562, 2015.

[9] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia. Automated aerial suspended cargo delivery through reinforcement learning. *Artificial Intelligence*, 247:381 – 398, 2017. Special Issue on AI and Robotics.

[10] A. Faust, P. Ruymgaart, M. Salman, R. Fierro, and L. Tapia. Continuous action reinforcement learning for control-affine systems with unknown dynamics. *Automatica Sinica, IEEE/CAA Journal of*, 1(3):323–336, 2014.

[11] R. Figueroa, A. Faust, P. Cruz, L. Tapia, and R. Fierro. Reinforcement learning for balancing a flying inverted pendulum. In *Proc. The 11th World Congress on Intelligent Control and Automation*, July 2014.

[12] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering. Springer, 1st ed. edition, 2011.

[13] K. Hauser, T. Bretl, J. Claude Latombe, and B. Wilcox. Motion planning for a sixlegged lunar robot. In *The Seventh International Workshop on the Algorithmic Foundations of Robotics*, pages 16–18, 2006.

[14] D. Hsu, R. Kindel, J.-C. Latombe, and S. M. Rock. Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robot. Res.*, 21(3):233–256, 2002.

[15] 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.*, 12(4):566–580, August 1996.

[16] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[17] J. Kuffner and S. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 995 –1001 vol.2, 2000.

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

[19] S. M. Lavalle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[20] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[21] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[22] N. Malone, A. Faust, B. Rohrer, R. Lumia, J. Wood, and L. Tapia. Efficient motion-based task learning for a serial link manipulator. *Transactions on Control and Mechanical Systems*, 3(1), 2014.

[23] N. Malone, K. Manavi, J. Wood, and L. Tapia. Construction and use of roadmaps that incoroporate worksapce modeling errors. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 1264–1271, Nov 2013.

[24] N. Malone, B. Rohrer, L. Tapia, R. Lumia, and J. Wood. Implementation of an embodied general reinforcement learner on a serial link manipulator. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 862–869, May 2012.

[25] C. Mansley, A. Weinstein, and M. Littman. Sample-based planning for continuous action markov decision processes. In *Proc. of Int. Conference on Automated Planning and Scheduling*, 2011.

[26] MARHES. Multi-Agent, Robotics, Hybrid, and Embedded Systems Laboratory, Department of Computer and Electrical Engineering, University of New Mexico. http://marhes.ece.unm.edu, February 2013.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[28] K. Mülling, J. Kober, and J. Peters. A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359–376, 2011.

[29] I. Palunko, P. Cruz, and R. Fierro. Agile load transportation : Safe and efficient load manipulation with aerial robots. *IEEE Robotics Automation Magazine*, 19(3):69 –79, sept. 2012.

[30] Parasol. Parasol Lab, Department of Computer Science and Engineering, Texas A&M University. https://parasol.tamu.edu/, May 2017.

[31] J.-J. Park, J.-H. Kim, and J.-B. Song. Path planning for a robot manipulator based on probabilistic roadmap and reinforcement learning. In *International Journal of Control, Automation, and Systems*, pages 674–680, 2008.

[32] S. Rodríguez, J.-M. Lien, and N. M. Amato. A framework for planning motion in environments with moving obstacles. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, pages 3309–3314, 2007.

[33] L. Tapia, S. Thomas, and N. M. Amato. A motion planning approach to studying molecular motions. *Communications in Information and Systems*, 10(1):53–68, 2010.

[34] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine. Collective robot reinforcement learning with distributed asynchronous guided policy search. *CoRR*, abs/1610.00673, 2016.