Whole-Body Control of a Mobile Manipulator using End-to-End Reinforcement Learning

Julien Kindle, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Roland Siegwart and Juan Nieto

Abstract— Mobile manipulation is usually achieved by sequentially executing base and manipulator movements. This simplification, however, leads to a loss in efficiency and in some cases a reduction of workspace size. Even though different methods have been proposed to solve Whole-Body Control (WBC) online, they are either limited by a kinematic model or do not allow for reactive, online obstacle avoidance. In order to overcome these drawbacks, in this work, we propose an end-to-end Reinforcement Learning (RL) approach to WBC. We compared our learned controller against a state-of-the-art sampling-based method in simulation and achieved faster overall mission times. In addition, we validated the learned policy on our mobile manipulator RoyalPanda in challenging narrow corridor environments.

I. INTRODUCTION

According to the International Federation of Robotics, 16.3M service robots for personal and domestic use were sold in 2018, which is a 59% increase compared to the previous year [1]. The majority of these commercially deployed robots target applications with a focus on mobility such as autonomous vacuum cleaner robots [2] and industrial transportation systems [3]. However, there exist far fewer mobile robots that incorporate advanced manipulation capabilities. Several research platforms [4, 5] have been developed for mobile manipulation tasks such as opening doors [6]. These typically decouple the movement of the base from the movement of the manipulator to simplify the control problem. However, in doing so, these approaches miss out on any potential synergies that arise from controlling the entire robot as a whole. Indeed, Whole-Body Control (WBC) aims at improving the efficiency and performance of such mobile manipulation robots by planning for and controlling all Degrees of Freedom (DoF) of the robot simultaneously.

There are several classes of approaches to whole-body trajectory planning and control. Sampling-based methods [7–10] grow a tree of collision-free joint configurations. However, these methods not only suffer from the curse of dimensionality but also require full knowledge of the environment at the planning stage. In order to operate in an unknown and possibly dynamic environment, which is observed only via sensors mounted to the robot, offline trajectory planning is often insufficient.

Another approach to tackle WBC is Model Predictive Control (MPC). In this field, the target is to solve the planning problem online by finding the sequence of controls
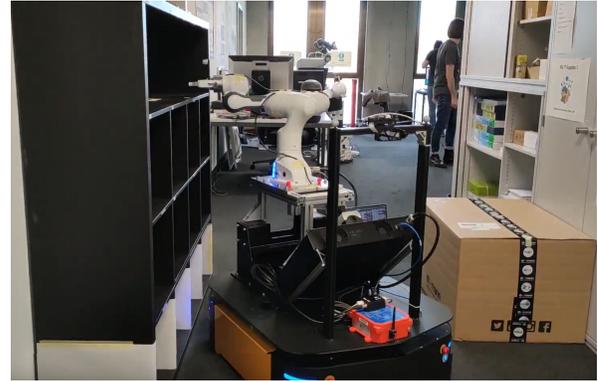
Fig. 1: RoyalPanda executing a learned WBC policy for reaching a setpoint in the shelf.

that optimize a given objective function, e.g. minimizing deviation from a target trajectory. Even though multiple MPC formulations have been proposed for solving WBC online, these are either limited in their obstacle avoidance capabilities [11] or only used a kinematic model [12].

Recent works have shown that learning-based methods can compete against classical algorithms for local path planning and manipulation tasks [13–19]. In particular, Reinforcement Learning (RL) has shown potential in fields where data collection and labelling for supervised learning is expensive.

In this work, we hypothesize that learning-based methods may be viewed as an alternative to sampling-based offline trajectory planning and online MPC. We use RL to train a controller for a mobile manipulator which observes its surroundings using two LiDAR scans and steers all DoF made available to the robot. For that, we created a simulation of our mobile manipulator RoyalPanda (Franka Emika Panda arm mounted on a Clearpath Ridgeback platform) in randomized corridor environments and trained a WBC policy to successfully steer the robot to commanded end-effector setpoints within it.

We make the following contributions:

- We present a learning-based approach to WBC that runs online at a rate of at least 100 Hz on an embedded device (Raspberry Pi 3B).
- We demonstrate the faster overall mission times (planning plus execution) of our trained controller as compared to a sampling-based method in simulation, and
- We show that the learned policy can be directly transferred to a real robot and achieve similar performance as in simulation.

II. Related Work

A simple yet widely used approach to mobile manipulation is the sequential execution of base and arm movements as in [20–22]. However, this approach is slow and, depending on the robot and workspace configuration, may require multiple iterations in order to reach a given end-effector goal.

A way to improve execution time and work space size is WBC, the simultaneous control of both the base and arm on a mobile manipulator. Classical sampling-based methods such as those offered by MoveIt! [23] can be modified to account for a (holonomic) moving base of the manipulator. Yang et al. [24] adapted the random sampler together with the sampling space and the interpolation function for common sampling-based algorithms [7–9, 25–27] to generate collision-free and balanced whole-body trajectories for humanoid robots. However, these algorithms generate an offline trajectory and therefore require a static and known environment.

On the other hand, MPC models the dynamics of a system together with the state boundaries as constraints over a fixed horizon and aims to minimize a cost function which is dependent on the states and inputs of the system. Examples in robotics include high-level control for fixed-wing guidance [28] and multi-MAV collision avoidance and trajectory tracking [29]. For mobile manipulation, several different approaches have been presented. Minniti et al. [30] propose an optimal control framework which allows WBC for end-effector tasks while simultaneously stabilizing the unstable base of a balancing robot in a single optimization problem. A Sequential Linear Quadratic (SLQ) method described by Farshidian et al. [11] was used as a solver for the optimal control problem. However, the final controller does not incorporate obstacles. Avanzini et al. [12] formulated an MPC which drives to a reference point while avoiding obstacles. The controller keeps the arm in a safe configuration by dynamically varying a weight matrix of the optimization problem as a function of goal distance. Their model formulation, however, is limited to a kinematic model.

RL has also been a popular choice for learning manipulation skills as well as navigation policies. Novkovic et al. [16] proposed an RL approach to object finding in environments which require physical interaction with the end-effector to expose the searched object. Breyer et al. [17] use RL with curriculum learning to train a controller from sparse rewards for end-effector object grasping. Popov et al. [18] used a modified variant of Deep Deterministic Policy Gradient (DDPG) to learn a control policy for grasping and stacking objects with a manipulator, while Gu et al. [19] developed an approach to train a controller with asynchronous RL directly on multiple robot arms.

In terms of learning navigation policies, Tai et al. [13] trained a mapless motion planner to drive a differential base to a setpoint given only 10 distance measurements. Pfeiffer et al. [14] generated data from the ROS navigation stack to train a controller in a supervised fashion to steer a differential drive robot from LiDAR scans. Iriondo et al. [15] trained a controller for the base of a mobile manipulator that drives the platform to a feasible position for grasping an object on a table. However, in their paper, the tests were only conducted in simulation, the absolute position of the robot was given to the controller and the structure of the environment was fixed throughout the training.

Recently, progress has been made in the field of MPC in combination with machine learning, especially RL. Farshidian et al. [31] developed an algorithm called Deep Value Model Predictive Control (DMPC) that uses an actor-critic agent whose critic is a neural network modelling a value function and whose actor is an MPC that uses the value function as part of its optimization cost. The MPC is then solved by using SLQ where the Hessian and Jacobian of the value function network are computed using an automatic differentiation library. In their work, the controller is trained for a single static environment.

Our approach is based on the work of Pfeiffer et al. [14] and Popov et al. [18]. We combine the information of two LiDAR scans, the system's internal states and the setpoint in end-effector frame in a deep neural network. The network is trained in simulation using RL as a WBC policy to drive the end-effector to the setpoint while avoiding obstacles.

III. Method

In this work, we use RL to discover WBC policies that can steer a mobile manipulator to target end-effector poses while avoiding obstacles. To do so, our system uses raw 2D LiDAR and the target position in end-effector frame as input to the RL algorithm. The agent (Section III-C) is trained in a simulated randomized corridor environment (Section III-B).

A. Observation and action space

We model our mobile manipulator as shown in Fig. 2. The observation of the complete system consists of two LiDAR scans (S_f, S_r), the joint positions (φ), joint and base velocities ($\dot{\varphi}$, $\dot{x}_b$, $\dot{y}_b$, $\dot{\theta}_b$) and the setpoint position in end-effector frame (P). The actions are the (discretized) accelerations of the base ($\ddot{x}_b$, $\ddot{y}_b$, $\ddot{\theta}_b$) and the joints ($\ddot{\varphi}$). The action discretization factor and the kinematic and dynamic limits used are listed in Table I.

B. Simulation

Due to the large amounts of data needed to train a (deep) neural network, an agent is first trained in simulation. For that, we created a 3D simulation of our mobile manipulator RoyalPanda (Fig. 1) using PyBullet [32] while offering an OpenAI Gym interface [33].

In order to train an agent that is able to control the system in a variety of environments, we randomize the simulation at each episode. In this work, we focus on scenarios where a mobile manipulator needs to operate in a corridor-like environment. We prevent overfitting to a specific setup by varying the corridor width, number and location of shelves, walls and doors and their respective dimensions. Similarly, special care was taken to ensure that the trained agent will work in a realistic environment with imperfect sensors. We added Gaussian noise to the LiDAR scans and randomly set measurements to the maximum distance ($5\,\mathrm{m}$).

C. The agent

We have chosen to use Proximal Policy Optimization (PPO) [34], a policy-based RL algorithm which is often used in the field of robotics [34–39] due to the ease of hyperparameter-tuning while retaining the stability and reliability of Trust Region Policy Optimization (TRPO) [40]. PPO is an actor-critic method which ensures that the policy does not change dramatically between batch updates, resulting in smoother, more stable and more robust learning behavior.

The architecture used to train the agent is depicted in Fig. 3. Data from the two LiDAR scans are compressed in the same branch composed of convolutional, max pooling and fully connected layers to 64 floats. These are then combined and compressed to 64 floats by three fully connected layers. At this point, the setpoint pose $\mathbf{P}$, base velocities $\dot{x}_b, \dot{y}_b, \dot{\theta}_b$, joint positions φ and joint velocities $\dot{\varphi}$ are concatenated and further compressed by eight fully connected layers to 32 floats. The model then separates into an actor and a critic, which result in the policy (5×5 floats) and the value function (1 float), respectively. The actions are then chosen by evaluating the argmax of each block of 5 floats.

D. Training procedure

PPO is an actor-critic method and can therefore be trained on data generated from multiple workers that run in parallel. We used Stable Baselines [41], a collection of RL algorithms that offers an implementation called PPO2 with multiple workers as well as GPU and Tensorboard support.

Training was conducted with 32 workers in parallel on an Intel Xeon E5-2640 v3 (8 Core @ 2.6 GHz) and the network was updated on an Nvidia GeForce GTX Titan X. The agent was trained with 60 M steps in simulation which took about 20 h. The learning parameters used in our experiments are listed in Table I.

In order to speed up the training process and robustly achieve convergence, we used a simplified variant of Automatic Domain Randomization (ADR) [42] on the size of the tolerance sphere around the setpoint (d_h).
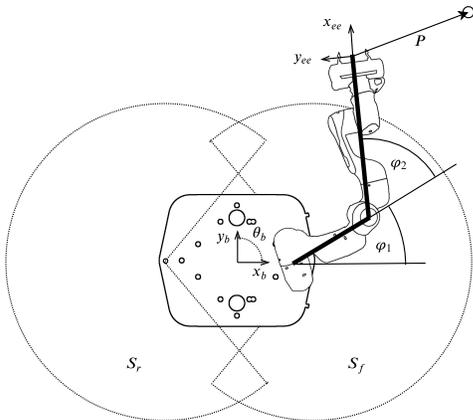

Fig. 2: Top down view of our mobile manipulator Royal-Panda. We lock the arm configuration such that the arm can only move in a 2D plane parallel to the floor with two actuated joints.
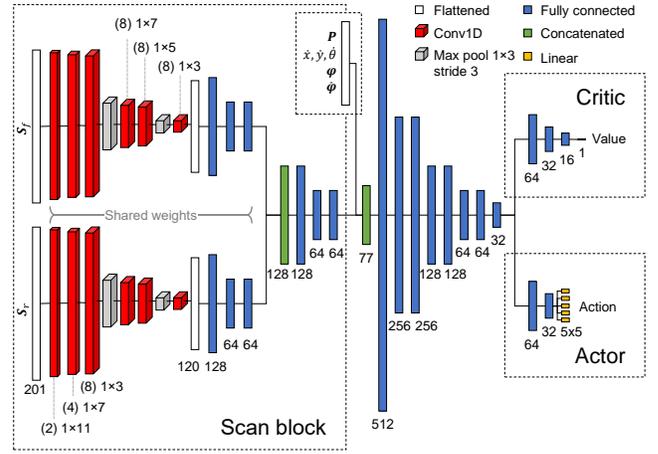

Fig. 3: The network architecture used in our experiments. All activations are of type LeakyReLU, except for the last layers of the actor and critic which are linear.

E. Reward function

We use a handcrafted reward function:

$$r(\Delta d_{pd}, \Delta d_{pt}, \vec{v}, d_{sm}, d_g, d_h, \vec{D}, I_h) =$$

$$w_t \frac{\tau}{T_t} + w_{pd}\Delta d_{pd} + w_{pt}\frac{\Delta d_{pt}}{d_{pt,init}} + w_{sm}\|\vec{v}\|\tau L(d_{sm}, d_{th})$$

$$+ (w_{ht} + w_{hd}L(d_g, d_h))\frac{\tau}{T_h} + D_c + D_l + D_h - I_h, \quad (1)$$

where $L(x, y) = 1 - \min(1, x/y)$. Individual components of r are described in the following subsections and the weights used in our experiments are listed in Table I.

1) Time penalty, w_t: We introduce a timeout T_t after which the episode is terminated. We penalize time in each step to favour actions that lead to reaching the setpoint faster, summing up to a maximum (negative) value of w_t.

2) Goal distance, w_{pd}, w_{pt}: We use a Harmonic Potential Field (HPT) to generate a collision-free path for the end-effector to the goal (Fig. 4) at the beginning of each episode. We then penalize deviation from the path (Δd_{pd}) by w_{pd} and reward progress made along the path (Δd_{pt}) such that the maximum total reward is w_{pt}. Compared to a simple metric

TABLE I: Learning and reward function parameters

Learning parameters			
noptepochs	30	cliprange	0.2
cliprange_vf	−1 (off)	ent_coeff	0.00376
gamma	0.999	lam	0.8
n_steps	2048	nminibatches	8
Learning rate	Linear drop from 1e−3 to 0.15e−3		

Reward function parameters			
w_t	−15	T_t	120
w_{pd}	−10	w_{pt}	30
w_{sm}	−1	d_{th}	0.3
w_{ht}	20	w_{hd}	40
T_h	1.5	τ	0.04
D_c	−60	D_l	−20
D_h	10		

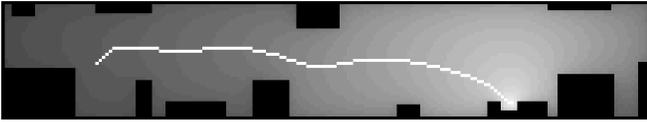| Observation and action space limits			
$\overline{S}$	5 m	$n_{\text{action\_discr}}$	5
$\overline{|\dot{\varphi}|}$	0.5 rad s^{-1}	$\overline{|\ddot{\varphi}|}$	0.8 rad s^{-2}
$\overline{|\dot{x}_b|}, \overline{|\dot{y}_b|}$	0.1 m s^{-1}	$\overline{|\ddot{x}_b|}, \overline{|\ddot{y}_b|}$	0.15 m s^{-2}
$\overline{|\dot{\theta}_b|}$	0.2 rad s^{-1}	$\overline{|\ddot{\theta}_b|}$	0.3 rad s^{-2}

Fig. 4: A Harmonic Potential Field which was used to generate a collision-free path for the end-effector to the goal position.

based on Euclidean distance to goal, our reward formulation allows the agent to easily learn to drive around obstacles and avoid getting stuck behind them in a local optima.

3) Safety margin, w_{sm}: We enforce a safety margin from obstacles for the base, by adding a term that linearly reduces the reward per driven distance with decreasing smallest distance to obstacle d_{sm}, starting with value 0 at threshold distance d_{th} and maximal reduction w_{sm} at distance 0.

4) Holding goal position, w_{ht}, w_{hd} and I_h: Many tasks such as grasping an object require the end-effector to keep still. Thus, instead of ending an episode as soon as the end-effector reaches the goal, we require it to be within a tolerance sphere of radius d_h (set by ADR, Section III-D) for a sustained interval T_h. The reward is incremented each time step the end-effector remains in the sphere, summing up to a total of w_{ht}. Additionally, we desire the goal distance d_g to be as small as possible, therefore rewarding a smaller distance with a maximum total reward of w_{hd} in a similar fashion as for the safety margin. To prevent the agent from exploiting the situation by repeatedly entering and leaving the sphere, we store the accumulated holding reward in I_h and subtract it immediately if the agent leaves the sphere.

5) Episode termination, D_c, D_l and D_h: Collision with an obstacle is penalized by D_c and reaching the joint limits is penalized with D_l. Otherwise, the agent is rewarded with D_h as soon as it achieves a sustained holding time of T_h. All three cases result in the termination of the current episode.

F. Sampling-based motion planning baseline

We compared our implementation to a baseline sampling-based algorithm. In particular, we used the implementation of RRTConnect [7] in the MoveIt! motion planning framework. Since MoveIt! does not natively support planning movements of a mobile base platform, we introduced two prismatic and one continuous joint to model the movement of a holonomic platform in a plane. The trajectory received by the planner can then be followed with a simple controller. However, an accurate localization system is required for tracking the mobile base trajectory.

Due to the fact that MoveIt! is an offline motion planning library, we generated an octomap of our environment by using a Gazebo plugin from RotorS [43]. This technique allows us to plan the trajectory to a setpoint in a single run but comes with the drawback that all obstacle positions must be static and known a priori. Further, to ensure reliable collision avoidance for the prismatic joints modelling the base platform, we reduced the search resolution of the kinematics solver to $1e-5$. We set the number of planning attempts to 20 and limited the planning time to $180\,\text{s}$.

IV. RESULTS

Figure 5 shows the accumulated reward per episode, the success rate and the ADR values during training. We conducted our experiments on a Raspberry Pi Model 3B [44] with an ARM Cortex-A53 (4 Core @ $1.2\,\text{GHz}$) and 1GB of RAM. Except for active cooling, we did not add any equipment (such as a Neural Compute Stick) to the device and therefore run the network directly on the CPU which required about 160MB of RAM. We achieved a network prediction rate of $104\,\text{Hz}$.
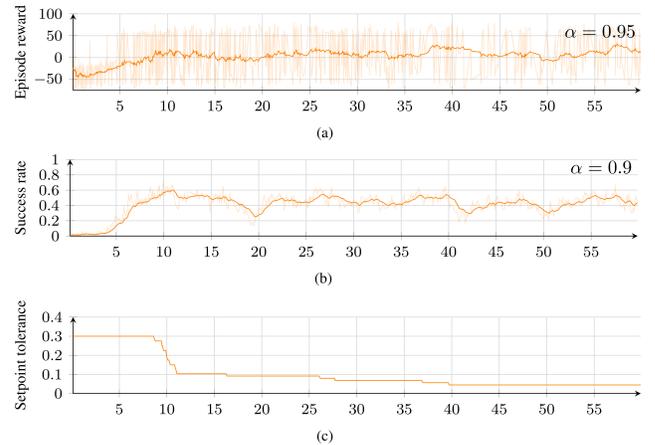

Fig. 5: Accumulated reward per episode and success rate with their respective smoothed plots (exponential moving average) and setpoint tolerance during training.

The trained agent was first evaluated in a Gazebo simulation (run on a separate machine) which modelled various corridor environments before we deployed it to the real system. We compared the performance against the RRTConnect planning baseline (Section III-F). The metrics used for comparing the two approaches are the total mission time, as a sum of the planning and execution time, and the total travelled base and joint distances. We additionally highlight the planning and execution times and the success rate for the two methods. Note that failure for the RL agent occurs due to collision or execution timeout ($180\,\text{s}$), while RRTConnect failure occurs when the planner is unable to find a collision-free path in the given time. The setpoint tolerance was fixed to a value of $0.07\,\text{m}$.

A. Simulation results

Figure 6 shows four different scenarios in which we evaluated the RL agent and RRTConnect. The four scenarios vary in difficulty according to the placement of obstacles in the environment. We randomly sampled a feasible initial configuration inside the blue ellipse and a setpoint in the red shelf and ran both algorithms. Table II shows the averaged metrics from 100 runs for each task.

Even though RRTConnect achieved a lower execution time in all tasks, due to the long planning time required, our approach had a consistently lower total mission time. In Task 2, our approach not only achieved a lower total
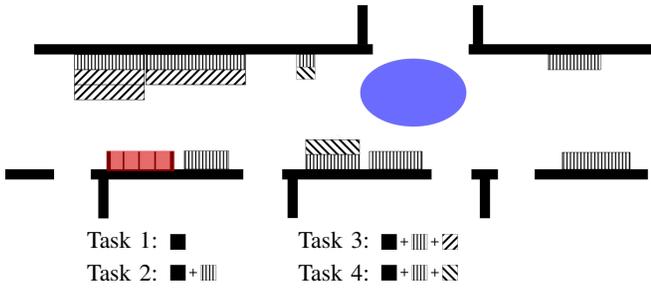
Fig. 6: Collision map of the simulated environments for tasks 1-4 used in our evaluation. The red square denotes the shelf in which the setpoints are spawned, the blue ellipse marks the region of initial poses.

mission time but also a higher success rate. In the remaining tasks, however, the success rate was notably lower. This might come from the fact that our procedure for environment randomization during training was more likely to produce scenes similar to Task 2 compared to the other tasks.

In all tasks, RRTConnect achieved smaller base and joint distances compared to the RL agent. The learned agent tended to exhibit small oscillatory behaviors whose presence may be due simply to the fact that we did not penalize base and joint distances in our reward function.

Note that even though the evaluations were done on static scenes, unlike RRTConnect, our approach can also handle dynamic scenes since we are controlling the system in a closed-loop fashion.

B. Real-world experiments

We verified the trained RL agent through 20 experimental runs on our mobile manipulator RoyalPanda. Maplab [45] was used for the estimation of the setpoint pose $\mathbf{P}$ in the end-effector frame. The agent ran on a laptop with an Intel i7-8550U (4 Core @ 1.8GHz) with 16GB of RAM. We deployed the robot in a corridor of our lab and tested the limitations of the agent with different initial robot configurations and a cardboard box to change the corridor width in different locations (see Fig. 1).

The experiments showed that the agent—which was solely trained in simulation—can be transferred to a real system. For a normal corridor environment similar to Task 2 (Fig. 6), the agent was consistently reaching the setpoint. However, for more difficult scenarios such as a tight corridor, the base platform sometimes collided into our cardboard box. A possible explanation is that the robot has two blind spots which are invisible to the LiDAR sensors. As soon as an obstacle corner is inside these, the agent seemed to cut the corner in order to reach the goal as quickly as possible, thus colliding with the box. This issue could be addressed by adding a memory structure to the network to help retain information from previous scans. Also, depending on the angle of the end-effector to the setpoint, the manipulator sometimes collided with the shelf. This may be due in part to an inaccurate manipulator-to-base calibration.

The accompanying video, which was also uploaded to

Youtube[1], shows some of the real-world experiments.

V. CONCLUSION

In this work, we have shown a method to learn WBC of a mobile manipulator by training a (deep) neural network with RL. We used ADR to automatically increase the complexity of the problem with growing success rate to speed up training and achieve robust convergence. We compared our agent against RRTConnect on an embedded device in simulation and showed that our approach achieves consistently shorter total mission times. Furthermore, we showed that the trained agent can be directly transferred to a real robot.

In future work, the success rate of our approach could be increased by adding a safety controller which separately limits the action space for the base platform and the manipulator to avoid collisions. Furthermore, the used DoF of the manipulator could be increased by adding a depth sensor as observation, potentially in an encoded form generated by an Auto-Encoder network.

REFERENCES

[1] I. F. of Robotics, "Executive summary world robotics 2019 service robots," https://ifr.org/downloads/press2018/Executive_Summary_WR_Service_Robots_2019.pdf, 2019.

[2] iRobot. (2020) iRobot: Vacuum, mop, & lawn mower. [Online]. Available: https://www.irobot.com

[3] R. D'Andrea, "Guest editorial: A revolution in the warehouse: A retrospective on Kiva systems and the grand challenges ahead," *IEEE Trans. on Automat. Sci. and Eng.*, vol. 9, no. 4, pp. 638–639, 2012.

[4] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mösenlechner, W. Meeussen, and S. Holzer, "Towards autonomous robotic butlers: Lessons learned with the PR2," in *2011 IEEE Int. Conf. on Robot. and Automat.*, 2011, pp. 5568–5575.

[5] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on Autonomous Mobile Service Robots*, 2016.

[6] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote *et al.*, "Autonomous door opening and plugging in with a personal robot," in *2010 IEEE Int. Conf. on Robot. and Automat.*, 2010, pp. 729–736.

[7] J. Kuffner and S. LaValle, "RRT-Connect: An efficient approach to single-query path planning." in *IEEE Int. Conf. on Robot. and Automat.*, vol. 2, 01 2000, pp. 995–1001.

[8] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

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

[10] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Trans. on Robot.*, vol. 26, no. 4, pp. 635–646, Aug 2010.

[11] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th Int. Conf. on Humanoid Robot. (Humanoids)*, 2017, pp. 577–584.

[12] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constraint-based model predictive control for holonomic mobile manipulators," in *2015 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*, 2015, pp. 1473–1479.

[13] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*, 2017, pp. 31–36.

[14] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end

---

[1]https://youtu.be/3qobNCMUMV4

TABLE II: Average simulation results of the RL agent and RRTConnect over 100 runs. Standard deviations are provided in brackets.

Task	Method	Total mission time [s]	Base distance [m]	Joint distance [rad]	Planning time [s]	Execution time [s]	Success rate
1	RL agent	**77.26 (9.82)**	6.71 (0.71)	16.88 (3.56)	-	77.26 (9.82)	0.55
	RRTConnect	82.07 (20.93)	**5.93 (1.61)**	**2.72 (1.35)**	22.83 (11.75)	**59.24 (16.15)**	**0.85**
2	RL agent	**79.31 (14.84)**	6.63 (1.02)	18.71 (4.24)	-	79.31 (14.84)	**0.79**
	RRTConnect	89.73 (14.82)	**5.96 (0.76)**	**3.07 (1.45)**	30.13 (12.85)	**59.61 (7.69)**	0.77
3	RL agent	**76.7 (13.74)**	6.4 (0.93)	16.98 (3.66)	-	76.7 (13.74)	0.56
	RRTConnect	102.05 (39.95)	**5.75 (1.06)**	**3.33 (1.66)**	44.77 (39.32)	**57.28 (10.66)**	**0.66**
4	RL agent	**81.45 (16.66)**	**6.86 (1.0)**	19.09 (3.94)	-	81.45 (16.66)	0.61
	RRTConnect	104.86 (42.84)	7.17 (4.0)	**3.94 (1.84)**	33.24 (13.4)	**71.62 (38.43)**	**0.85**
Avg.	RL agent	**78.8 (14.26)**	6.65 (0.95)	18.02 (4.02)	-	78.8 (14.26)	0.63
	RRTConnect	94.59 (33.4)	**6.26 (2.45)**	**3.28 (1.66)**	32.1 (22.45)	**62.48 (23.79)**	**0.80**

motion planning for autonomous ground robots," in *IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2017, p. 15271533.

[15] A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, A. Fernandez, and J. Molina, "Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning," *Applied Sciences*, vol. 9, p. 348, 01 2019.

[16] T. Novkovic, R. Pautrat, F. Furrer, M. Breyer, R. Siegwart, and J. Nieto, "Object finding in cluttered scenes using interactive perception," in *2019 IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2019, to appear.

[17] M. Breyer, F. Furrer, T. Novkovic, R. Siegwart, and J. Nieto, "Comparing task simplifications to learn closed-loop object picking using deep reinforcement learning," *IEEE Robot. and Automat. Lett.*, vol. 4, no. 2, pp. 1549–1556, 2019.

[18] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[19] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE Int. Conf. on Robot. and Automat. (ICRA)*, 2017, pp. 3389–3396.

[20] M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. Le, A. Wellman, and A. Y. Ng, "High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening," in *2009 IEEE Int. Conf. on Robot. and Automat.*, 2009, pp. 2816–2822.

[21] Y. Wang, H. Lang, and C. W. De Silva, "A hybrid visual servo controller for robust grasping by wheeled mobile robots," *IEEE/ASME Trans. on Mechatronics*, vol. 15, no. 5, pp. 757–769, Oct 2010.

[22] Z. Li, T. Zhao, F. Chen, Y. Hu, C.-Y. Su, and T. Fukuda, "Reinforcement learning of manipulation and grasping using dynamical movement primitives for a humanoidlike mobile manipulator," *IEEE/ASME Trans. on Mechatronics*, vol. 23, no. 1, pp. 121–131, 2017.

[23] I. A. Sucan and S. Chitta, "Moveit," https://github.com/ros-planning/moveit, 2020, gitHub repository.

[24] Y. Yang, V. Ivan, W. Merkt, and S. Vijayakumar, "Scaling sampling-based motion planning to humanoid robots," in *2016 IEEE Int. Conf. on Robot. and Biomimetics (ROBIO)*, Dec 2016, pp. 1448–1454.

[25] I. A. Şucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," in *Algorithmic Foundations of Robot. VIII*, ser. Springer Tracts in Adv. Robot. Springer, Berlin, Heidelberg, 2009, vol. 57, pp. 449–464.

[26] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of Int. Conf. on Robot. and Automat.*, vol. 3, April 1997, pp. 2719–2726.

[27] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Springer Tracts in Adv. Robot.* Springer, Berlin, Heidelberg, 2003, vol. 6, pp. 403–417.

[28] T. Stastny and R. Siegwart, "Nonlinear model predictive guidance for fixed-wing UAVs using identified control augmented dynamics," in *2018 Int. Conf. on Unmanned Aircraft Syst. (ICUAS)*, 2018, pp. 432–442.

[29] M. Kamel, J. Alonso-Mora, R. Siegwart, and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," in *2017 IEEE/RSJ Int. Conf. on Intell. Robots and Syst. (IROS)*, Sep. 2017, pp. 236–243.

[30] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body MPC for a dynamically stable mobile manipulator," *IEEE Robot. and Automat. Lett.*, vol. 4, pp. 3687–3694, 2019.

[31] F. Farshidian, D. Hoeller, and M. Hutter, "Deep value model predictive control," in *3rd Conf. on Robot Learning*, 2019, pp. 1–15.

[32] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[33] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," OpenAI, arXiv preprint arXiv:1606.01540, 2016.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," OpenAI, arXiv preprint arXiv:1707.06347, 2017.

[35] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," DeepMind, arXiv preprint arXiv:1707.02286, 2017.

[36] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *2nd Conf. on Robot Learning*, 2018, pp. 1–31.

[37] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Proc. of Robot.: Sci. and Syst.*, Pittsburgh, Pennsylvania, June 2018.

[38] Y. Chen and L. Ma, "Rocket powered landing guidance using proximal policy optimization," in *Proc. of the 2019 4th Int. Conf. on Automat., Control and Robot. Eng.* Association for Computing Machinery, 2019, pp. 1–6.

[39] G. C. Lopes, M. Ferreira, A. da Silva Simões, and E. L. Colombini, "Intelligent control of a quadrotor with proximal policy optimization reinforcement learning," in *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robot. (SBR) and 2018 Workshop on Robot. in Education (WRE)*, Nov 2018, pp. 503–508.

[40] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Int. Conf. on Machine Learning*, 2015, pp. 1889–1897.

[41] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018, gitHub repository.

[42] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas *et al.*, "Solving rubik's cube with a robot hand," OpenAI, arXiv preprint arXiv:1910.07113, 2019.

[43] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS a modular Gazebo MAV simulator framework," in *Robot Operating System (ROS)*, ser. Studies in Computational Intelligence. Springer, Cham, February 2016, vol. 625, pp. 595–625.

[44] Rasperry Pi Foundation. (2020) Raspberry Pi: Teach, learn and make with raspberry pi. [Online]. Available: https://www.raspberrypi.org

[45] T. Schneider, M. T. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robot. and Automat. Lett.*, 2018.