Learning a Meta-Controller for Dynamic Grasping

Yinsen Jia [*, 1], **Jingxi Xu** [*, 2], **Dinesh Jayaraman** [3], **Shuran Song** [2]
[*]Equal contribution
[1]Department of Electrical Engineering, Columbia University
[2]Department of Computer Science, Columbia University
[3]GRASP Lab, University of Pennsylvania

Abstract

Grasping moving objects is a challenging task that combines multiple modules such as object pose predictor, arm motion planner, etc. Each module operates under its own set of meta-parameters, for example, the prediction horizon in the pose predictor and the time budget for planning motion in the motion planner. Many previous works assign fixed values to these parameters either heuristically or through grid search; however, at different time steps within a single episode of dynamic grasping, there should be different optimal values for each parameter, depending on the current scene. In this work, we learn a meta-controller through reinforcement learning to control the prediction horizon and time budget dynamically at each time step. Our experiments show that the meta-controller significantly improves the grasping success rate and reduces grasping time, compared to baselines whose parameters are fixed or random. Our meta-controller learns to reason about the reachable workspace and through dynamically controlling the meta-parameters, it maintains the predicted pose and the planned motion within the reachable region. It also generalizes to different environment setups and can handle various target motions and obstacles.

1 Introduction

Grasping moving targets without prior knowledge of their motion remains a challenging task and due to its complexity, roboticists usually decompose the entire task into several modules. For example, [1] identifies two key components in dynamic grasping as below.

Object Pose Predictor [12, 11, 13, 3]. This component models the unknown object motion using its past trajectory and predicts a future pose of the object according to a specific *prediction horizon* (i.e., how far into the future the predicted object pose should be). Prediction is critical in dynamic grasping because, without prediction, the robot arm can never catch up with the target object.

Arm Motion Planner [4, 9, 6, 7]. This component attempts to generate a collision-free path towards the planned grasp pose. It operates with respect to a *time budget*. If no successful path is planned within the time budget, the motion planning is terminated with a failure. Otherwise, a solution is returned as soon as a successful path is planned.

The prediction horizon in the pose predictor and the time budget in the motion planner bring important trade-offs into the systems, and they impact each other significantly. The prediction horizon highly affects the predicted object pose and then consequently the success of the motion planning. For example, if the object has moved into a cluster of obstacles, the pose predictor should use a large horizon to generate a future pose outside these obstacles so that a collision-free motion can be planned earlier. On the other hand, the time budget brings the trade-off between planning success and delay. The larger the time budget, the more likely a successful path can be found, but at the same time, if this time budget is used up, the target object might have already moved away, outside the prediction horizon. Many previous works [1, 14, 5, 8, 2] assign a fixed value to these parameters

NeurIPS 2022 Workshop on Robot Learning: Trustworthy Robotics, Virtual, Virtual

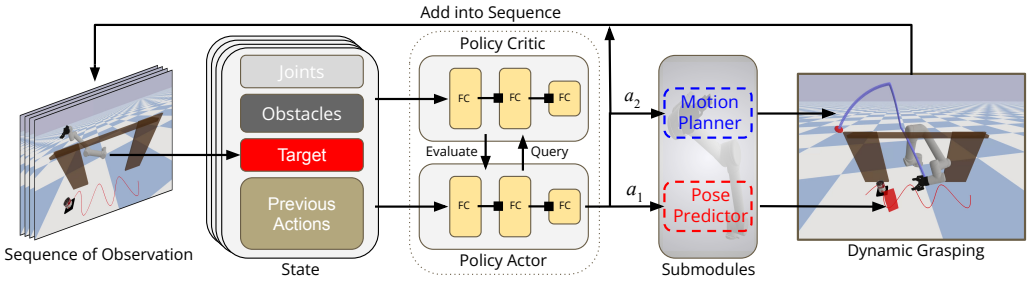


Figure 1: Learning a dynamic grasping meta-controller. We stack a sequence of scene information as the state input to our meta-controller. The meta-controller is modeled as a PPO agent trained with RL and a sparse reward. Our meta-controller generates a prediction horizon a_1 which is used by the object pose predictor, and a time budget a_2 which is used by the motion planner. The red cuboid indicates the predicted object pose, the red sphere indicates the end-effector initial pose and the blue curve indicates the planned path.

either heuristically or through grid search; however, during dynamic grasping, their optimal values should differ at each time step, depending on the current scene/environment.

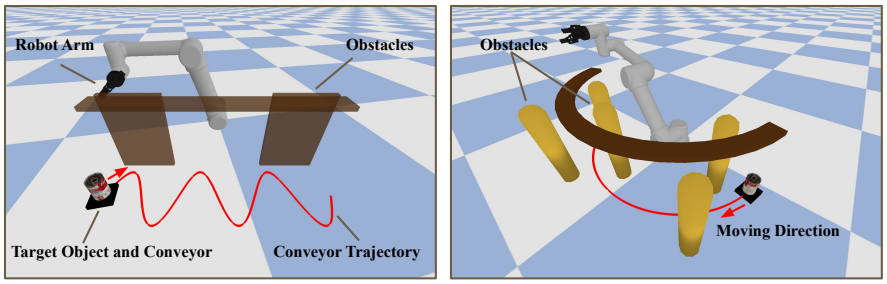
In this work, we focus on controlling *meta-parameters* dynamically. Meta-parameters are the parameters of individual submodules within a sophisticated system. Different from hyperparameters, meta-parameters from different modules affect each other and their optimal values differ at each time step. To the best of our knowledge, we are the first to learn a meta-controller through reinforcement learning (RL) and demonstrate our method on dynamic grasping. With the meta-controller, our method significantly improves the success rate and reduces the grasping time of grasping moving objects compared to baselines whose meta-parameters are fixed or random. We believe our method can be applied to many complex systems consisting of individual modules with meta-parameters.

2 Method

The goal of our method is to dynamically assign meta parameters based on the current scene. We choose the dynamic grasping task to demonstrate our method, in which we focus on learning a meta-controller for prediction horizon and time budget as discussed in Sec. 1. Assigning the correct values to the prediction horizon and time budget is a difficult task and even humans can not provide straightforward intuitions. The assignment of these meta-parameters can have a delayed effect, and the interplay between prediction horizon and time budget is hard to model. In our proposed method, the meta-controller is modeled as an RL agent. Without any prior knowledge, our meta-controller learns to control these parameters through trials and errors with a sparse reward.

Background. We use a similar dynamic grasping pipeline as proposed in [1]. At each time step, a predicted pose is output by the object pose predictor. The grasp defined in the object frame is then transformed according to the predicted object pose. Grasp planning is outside the scope of this work and we use a fixed grasp throughout the whole episode. If the transformed grasp pose is reachable, the motion planner plans a motion to reach that pose within the time budget. If a collision-free path is found, the arm starts moving towards that grasp pose, and then the loop continues until the distance between the arm end-effector and the current object pose is smaller than a predefined threshold.

State and Action Space. It is critical that our meta-controller has access to all the necessary information about the current scene as the optimal meta-parameters highly depend on the current state of the environment. We identify the below important information to learn a meta-controller. (1) Joints: the robot arm joint values $J \in \mathbb{R}^n$, where n is the number of degrees of freedom. (2) Target: this includes the target object moving speed $v \in \mathbb{R}$ modeled by our pose predictor, the distance from the object to our arm end-effector $d \in \mathbb{R}$, the target position $P \in \mathbb{R}^3$, the target quaternion $Q \in \mathbb{R}^4$, and target bounding box dimension $B \in \mathbb{R}^3$. (3) Obstacles: similar to target, we model each obstacle with a 10-dimension vector (position + quaternion + bounding box dimension). (4) Previous Actions: this contains all the information about the meta-parameters in the previous time step, including the prediction horizon, time budget, motion planning success, inverse kinematics (IK) success, IK time, motion planning time, and whether the arm is close enough to the target for grasp execution. At each



(a) Easy Setup **(b) Hard Setup**

Figure 2: Dynamic grasping in highly-cluttered environments. The UR5 arm attempts to pick up a tomato soup can on a moving conveyor. Obstacle locations are partially randomized. (a) Easy setup. (b) Hard setup.

time step, a history of such information from the past 5 time steps is stacked as the state input to our meta-controller. At each time step, our meta-controller outputs two continuous actions, prediction horizon $a_1 \in [0s, 8s]$ and motion planning time budget $a_2 \in [0s, 4s]$.

Training. We use the Proximal Policy Optimization (PPO) algorithm [10] to train our meta-controller. Both the critic and actor networks are a 4-layer multilayer perceptron (MLP) where the first three layers have 128, 64, and 64 neurons. The last layers of critic and actor have output dimensions 1 and 2 respectively. For each episode, a reward of 1 is given only if the robot arm successfully grasps the object. The episode is terminated if an unwanted collision happens, the object/obstacles are knocked over, or the object has reached the end of the trajectory.

3 Experiments

We train and evaluate our meta-controller in a comprehensive range of environment setups with different target motions and obstacles, for better generalization abilities. The metrics that we are most interested in are (1) success rate - the ratio of episodes that the robot successfully picks up and lifts the object over a height of 3cm, and (2) grasping time - the time taken for the object to be lifted up.

Experimental Setups. We design two setups (easy and hard) in the PyBullet simulator (Fig. 2). For both setups, the trajectories of the object are covered and surrounded by obstacles, resulting in a limited workspace, to mimic conveyor belts in industrial warehouses. The object speed is sampled between 2 - 6cm/s and stays unchanged for the whole episode. The distance between the trajectory and the robot base is randomized between 35 - 65cm. Our arm motion planner uses rapidly-exploring random tree (RRT) algorithm and similar to [1], we use an LSTM-based object pose predictor.

In the easy setup, there are three static shelves (one top shelf and two side shelves) between the robot arm and the object trajectory. For each episode, the trajectory is sampled from linear, sinusoid, and rectangular trajectories. The top shelf height is randomized between 40 - 60cm and the side shelf locations are randomized so that an empty middle space of size 45 - 85cm is available. In the hard setup, the object moves following a circular trajectory and there are 4 cylinder obstacles surrounding the trajectory and a top circular shelf obstacle covers the trajectory. Positions of these cylinder obstacles are randomly sampled. This is a harder and more cluttered setup.

Baselines. We compare our approach to two baselines. (1) *Random*: the prediction horizon and time budget are sampled randomly. (2) *Grid-search*: we choose 8 values evenly spaced in the action space for both prediction horizon and time budget. We evaluate all possible (prediction horizon, time budget) pairs. The pair with the highest performance is chosen as fixed meta-parameters for this method. We find (2s, 1s) and (1s, 2s) give the best performance for easy and hard setups respectively.

Performance Analysis and Discussion. The results are shown in Table 1. With the learned meta-controller, we achieve a higher success rate and lower dynamic grasping time compared to all baselines. Our meta-controller learns to reason about the reachable workspace and through dynamically controlling the prediction horizon and time budget, it maintains the predicted pose and the planned motion within the most reachable region. For example, in the easy setup, there is a limited reachable area between the two side shelves. Our meta-controller tends to output a large prediction horizon in the beginning so that the predicted pose is within the reachable space between

Table 1: Performance analysis of different methods under the easy and hard setups. Each number is reported over 200 trials. We report the success rate (higher values are better) and the grasping time (mean $\pm$ one standard deviation, lower values are better). The best performance is in bold.

Method	*Easy Setup*		*Hard Setup*	
	Success Rate	**Grasping Time**	**Success Rate**	**Grasping Time**
Random	0.49	14.88 ± 11.07	0.37	23.21 ± 20.64
Grid-search	0.73	11.42 ± 11.43	0.53	19.49 ± 22.05
Meta-controller (ours)	**0.80**	$\mathbf{9.751 \pm 7.240}$	**0.61**	$\mathbf{14.21 \pm 13.10}$

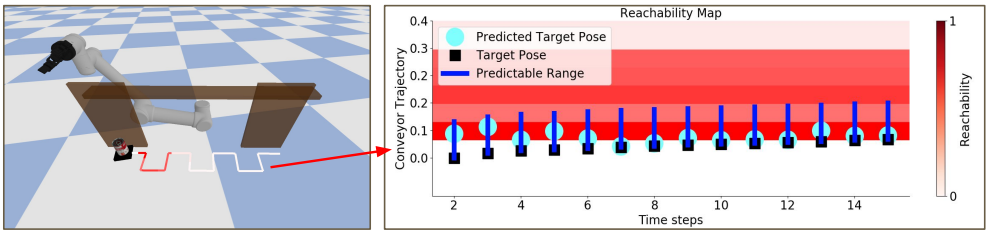


Figure 3: Prediction horizon visualization on trajectory reachability map. (a) Easy setup with rectangular trajectory. (b) The step-by-step prediction horizon from our meta-controller. The trajectory is vertically projected on the Y-axis with reachability visualization. The reachability value is proportional to how fast a motion can be found with repetitive offline motion planning.

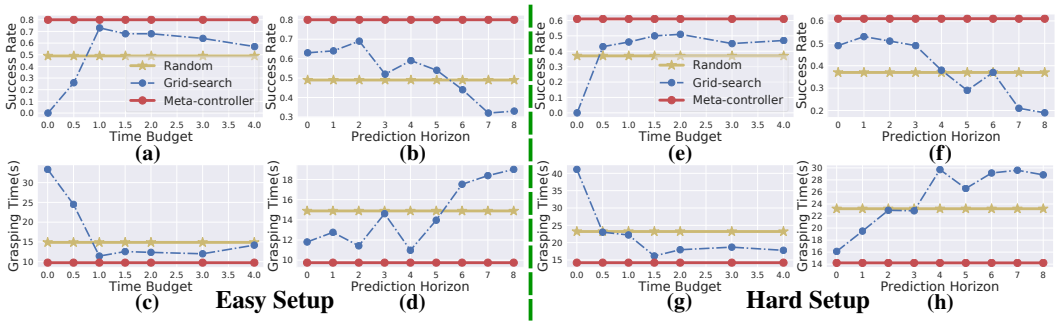


Figure 4: Comparison with different fixed meta-parameters in *Grid-search*. Top row: success rate, the higher the better. Bottom row: grasping time, the lower the better. (a)(b)(c)(d) easy setup. (e)(f)(g)(h) hard setup.

the two side shelves. After the object has entered that space, our meta-controller tends to reduce the prediction horizon to make the predicted pose remain in that reachable area. This behavior can be visualized in Fig. 3. On the contrary, if we use a fixed large prediction horizon, the predicted pose quickly goes outside that reachable area; if we use a fixed small prediction horizon, a collision-free path can be found only when the object is close to the reachable region, but since the reachable area is quite restricted, the object will have already moved out of it as the arm moves closer.

We further compare our method to different fixed prediction horizons and time budgets in the *Grid-search* baseline, as shown in Fig. 4. We keep one of the meta-parameters in the optimal pair and then vary the other. These results show that even with a fixed meta-parameter, different values can impact performance significantly. However, our meta-controller outperforms all fixed values by changing the meta-parameters at different time steps, based on the scene information.

4 Conclusion

We train a meta-controller through reinforcement learning to control the prediction horizon and time budget dynamically at each time step. The meta-controller drastically improves the performance of dynamic grasping and can generalize to different environment setups. Potential future work includes generalizing to different target objects and grasp poses.

References

[1] Iretiayo Akinola, Jingxi Xu, Shuran Song, and Peter K Allen. Dynamic grasping with reachability and motion awareness. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9422–9429. IEEE, 2021.

[2] Peter K Allen, Aleksandar Timcenko, Billibon Yoshimi, and Paul Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, 1993.

[3] Jan Issac, Manuel Wüthrich, Cristina Garcia Cifuentes, Jeannette Bohg, Sebastian Trimpe, and Stefan Schaal. Depth-based object tracking using a robust gaussian filter. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 608–615. IEEE, 2016.

[4] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[5] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018.

[6] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[7] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[8] Naresh Marturi, Marek Kopicki, Alireza Rastegarpanah, Vijaykumar Rajasekaran, Maxime Adjigble, Rustam Stolkin, Aleš Leonardis, and Yasemin Bekiroglu. Dynamic grasp and trajectory planning for moving objects. *Autonomous Robots*, 43(5):1241–1256, 2019.

[9] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494. IEEE, 2009.

[10] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[11] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[12] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3343–3352, 2019.

[13] Manuel Wüthrich, Peter Pastor, Mrinal Kalakrishnan, Jeannette Bohg, and Stefan Schaal. Probabilistic object tracking using a range camera. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3202. IEEE, 2013.

[14] Xinyu Ye and Shan Liu. Velocity decomposition based planning algorithm for grasping moving object. In *2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 644–649. IEEE, 2018.