Efficient Object Manipulation Planning with Monte Carlo Tree Search

Huaijiang Zhu*, Ludovic Righetti*†
*New York University, New York, USA
†Max-Planck Institute for Intelligent Systems, Tübingen, Germany

***Abstract*—This work presents an efficient approach to object manipulation planning using Monte Carlo Tree Search (MCTS) to find contact sequences and an efficient ADMM-based trajectory optimization algorithm to evaluate the dynamic feasibility of candidate contact sequences. To accelerate MCTS, we learn a goal-conditioned policy-value network used to direct the search towards promising nodes. Further, manipulation-specific heuristics enable to drastically reduce the search space. Systematic object manipulation experiments in a physics simulator demonstrate the efficiency of our approach. In particular, our approach scales favorably for long manipulation sequences thanks to the learned policy-value network, significantly improving planning success rate.**

I. Introduction

The ability to plan sequences of contacts and movements to manipulate objects is central to endow robots with sufficient autonomy to perform complex tasks. This remains, however, particularly challenging as it typically leads to intractable combinatorial and nonlinear problems. One common formulation of such problem is via Mixed-integer Programming (MIP). In the context of robot manipulation, one representative work is the Contact-Trajectory Optimization proposed in [1], where contact scheduling is modeled as binary decision variables and the non-convexity due to cross product is relaxed by McCormick envelopes. However, the approach has only been demonstrated on 2D object manipulation with very short manipulation sequences.

In principle, we can employ a brute-force approach to MIP problems: search over all possible combinations of the discrete variables and for each such combination solve the resulted continuous optimization problem. In general, such strategy is not practical due to the factorial complexity. However, it can be made more efficient if 1) the search space can be notably reduced, 2) good search heuristics are available, and 3) the non-convex continuous optimization problem can be solved efficiently.

In this work, we show that all these three requirements can be achieved. In particular, our contributions are 1) we adapt learning-based Monte Carlo Tree Search (MCTS) to discrete contact planning problems for robotic manipulation, 2) we formulate the resulted continuous optimization problem as a biconvex program to allow efficient solution via the Alternating Direction Method of Multipliers (ADMM) [2], and 3) we learn a policy-value network from data collected on short-horizon tasks which provides good heuristics for long-horizon tasks and significantly decreases the overall solution time. To our best knowledge, this is the first application of learning-based MCTS to contact planning for manipulation.

II. Problem Statement and Method Overview

A. Inputs

We aim to solve an object manipulation task similar to the Contact-Trajectory Optimization problem proposed in [1] where the following quantities are given: 1) a rigid object with known geometry and dynamics, and N_Ω pre-defined touchable regions, 2) a trajectory of length T that consists of the desired object pose, velocity, and acceleration, 3) an environment with known geometry and friction coefficient μ_e, and 4) a manipulator with known kinematics that can make at most N_c contacts with the object.

B. Outputs

For each time step t, we aim to find the following: 1) the contact region $\Omega_c(t) \in \{0, 1, \dots, N_\Omega\}$, the contact force $f_c(t)$ and the contact location $r_c(t)$ for each contact point c of the manipulator; $\Omega_c(t) = 0$ indicates that the c-th contact point is not in contact, and 2) the environment contact force $f_e(t)$ such that the forces and torques sum to the desired ones which can be computed from the object motion.

C. Continuous Contact Optimization via ADMM

If the discrete contact regions were known, the problem could be reduced to a continuous optimization problem with an interesting feature: the only non-convex constraint due to the cross product $r_c \times f_c$ is in fact biconvex. When we group the decision variables into two sets $x = [r_c(t)]_{t=0}^{T-1}$ and $z = [f_c(t), f_e(t)]_{t=0}^{T-1}$, all other standard constraints such as linearized friction cone and sticking contact are convex and separable in x and z; hence, the problem can be treated as a biconvex program and solved via ADMM, which only entails solving a handful inequality-constrained Quadratic Programs (QPs).

D. Discrete Contact Planning via MCTS

A family of learning-based MCTS algorithms, which we will refer to as Policy-Value Monte Carlo Tree Search (PVMCTS), has been proposed in [3, 4] for the chess-playing agents AlphaGo and AlphaZero. In our framework, the PVMCTS searches for contact sequences that are kinematically feasible, persistent and only allow contact switches at

TABLE I: Task performance for motions interpolated from randomly sampled poses with various lengths. Pose errors are calculated only for successful tasks.

# Object motions	Trajectory length T	Model	Success rate	Error [cm,°]		# Evaluation		Time [s]	
				Average	Worst	Average	Worst	Average	Worst
1	48	Untrained	**20/20**	0.16, 1.18	0.57, 5.89	4.65	11	2.09	4.88
		Trained	**20/20**	**0.15, 0.39**	**0.24, 0.83**	**1.5**	**4**	**0.71**	**1.73**
2	96	Untrained	**20/20**	0.35, 1.23	0.79, 2.24	8.15	25	8.54	21.88
		Trained	**20/20**	**0.32, 0.88**	**0.48, 1.78**	**2**	**5**	**1.96**	**4.68**
3	144	Untrained	12/20	0.48, 1.86	0.91, 5.98	29.85	50	46.23	84.63
		Trained	**20/20**	**0.43, 1.81**	**0.58, 4.84**	**2.3**	**8**	**3.18**	**9.43**
4	192	Untrained	5/20	**0.61, 1.95**	**0.74, 2.13**	43.05	50	93.57	137.31
		Trained	**20/20**	0.65, 2.56	1.59, 6.92	**2.8**	**16**	**6.12**	**31.02**

zero velocity and acceleration; when the angular acceleration is nonzero, we require at least 3 contact points to be in contact. The candidate sequences are evaluated by solving the resulted continuous optimization problem via ADMM, where the solution is then integrated to give a pose error normalized between $[0, 1]$ as the reward function. The collected rewards, the search decisions and the goal pose of the task are then used to learn a goal-conditioned policy-value network to guide future search. To avoid biasing the learned value function, we train it only on data with positive rewards. We additionally train a binary classifier to determine if the contact sequence should be fed into the policy-value network.

III. Experiments

We conduct simulation experiments to show that our method 1) is capable of finding dynamically feasible solutions to manipulation planning problems defined in Sec II, and 2) scales to long-horizon tasks even when trained only on data collected from short-horizon tasks.

A. Experiment Setup

Throughout all experiments, we consider a manipulator with $N_c = 2$ contact points, composed of two modular robot fingers similar to the ones used in [5] and a $10\,\mathrm{cm} \times 10\,\mathrm{cm} \times 10\,\mathrm{cm}$ cube with mass $m = 0.5\,\mathrm{kg}$ on an infinitely large plane. The cube and the plane have the same friction coefficient $\mu = \mu_e = 0.8$. The manipulation tasks are defined by interpolating planar object motions with randomly sampled desired position and rotation around the z-axis. The trajectories in the training data all have a length of $T = 96$.

B. Metrics

We examine three performance metrics to evaluate the effectiveness and efficiency of our method 1) **Pose error:** the error between the desired pose and the one integrated from the solution. 2) **Number of evaluations:** the number of continuous optimization problems the PVMCTS needs to solve until it finds the first feasible solution. 3) **Solution time:** the total time needed to find the first feasible solution.

C. Results

We evaluate the trained and untrained models on tasks that are generated by the same procedure yet have different trajectory lengths. Each task category with the same trajectory length has 20 different randomly generated tasks. A task is considered failed if no feasible solution within the error threshold is found after evaluating 50 contact sequences. Table I reports the performance metrics of the untrained and trained model for each task category. We see that the trained model consistently solve all the tasks, regardless of the trajectory length, while the untrained model struggles in long-horizon tasks, solving only 5 out of 20 tasks with trajectory length $T = 192$. In contrast to the untrained model, the average number of evaluations required by the trained model to find the first feasible solution grows rather slowly with the trajectory length.

IV. Conclusion

In this work, we proposed a framework that combines data-driven tree search via PVMCTS and efficient non-convex optimization via ADMM to find dynamically feasible contact forces and locations to realize a given object motion. We show that the capability of learning from data allows our framework to achieve great scalability for long-horizon motions even when the dataset only contains data collected from shorter motions. We recognize that two most limiting aspects of our approach are 1) the object motion must be provided; 2) perfect knowledge about the environment is required, which we leave for future work.

References

[1] Bernardo Aceituno-Cabezas and Alberto Rodriguez. A global quasi-dynamic model for contact-trajectory optimization. In *Robotics: Science and Systems (RSS)*, 2020.

[2] Stephen Boyd et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1): 1–122, 2011.

[3] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[4] David Silver et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[5] Manuel Wüthrich et al. Trifinger: An open-source robot for learning dexterity. *arXiv preprint arXiv:2008.03596*, 2020.