Fast and Robust Trajectory Generation for Cartesian Path-following Problems of Redundant Manipulators

Minsung Yoon[†], Mincheul Kang[†], Daehyung Park[†‡] and Sung-Eui Yoon[†‡]

***Abstract*— It is an important problem that quickly finds a joint trajectory so that an end-effector path of the trajectory precisely follows the Cartesian path defined in SE(3). However, as the length of the considered path or the degree of freedom of the robot increases, it becomes very complicated to find a trajectory that satisfies necessary constraints such as continuity, mechanical limits, singularity, and collision avoidance. Therefore, we present a learning-based trajectory generation framework that can rapidly produce a joint trajectory while satisfying constraints with generalizability in the configuration of the external environment and the path. Our Markov decision process formulation enables our trained policy to generate trajectories with a lower constraint violation rate than the three other trajectory generation baseline methods.**

I. INTRODUCTION

A path-following problem of manipulators is an important issue for real-world tasks in remote control or other domains. Given a fully constrained (i.e., 6-dimensional) path, we aim to find a configuration-space trajectory for kinematically redundant manipulators taking into consideration a variety of trajectory constraints, such as joint continuity, smoothness as well as potential collision in the environment.

Traditionally, inverse kinematics (IK) has been utilized to find a joint configuration given an end-effector pose [1]. However, the IK approach does not consider the constraints that arise when tracing the path, leading to a situation where a valid configuration no longer exists. Therefore, global search approaches have built a discrete layered graph with IK solutions computed along the path to search feasible trajectories [2]. Although these approaches are asymptotically optimal, they are quite slow since the redundant manipulator has infinite IK solutions even at one end-effector pose. As trajectory optimization has been widely adopted for generating a feasible trajectory, previous method [3] append constraints on the end-effector poses to follow the path. However, an optimized result is highly sensitive to an initial guess, i.e., initial trajectory, since these constraints cause many local minima and the local nature of the optimization method.

To balance a generation time and the trajectory quality, we amortize the online computation to satisfy the constraints by training our neural network policy offline using reinforcement learning (RL) with a variety of path-following problems and demonstrations. Our method generates the trajectory sequentially by selecting the extension direction in configuration space based on the path and environment information starting from an initial configuration. To learn such behavior, we formulate the path-following problem as a finite-horizon Markov decision process (MDP) by defining a unified reward function composed of the task, imitation, and constraint-relevant rewards. In addition, for generalizability over diverse path-following problems (e.g., paths, start configurations, and environments), we generate diverse training environments.

We experimented with a 7 DoF Fetch robot and compared our work with the conventional IK method and a supervised learning-based method as another learning-based method. As a result, it showed faster generation time and a lower constraint violation rate with the improved null-space continuity.

[†]M. Yoon, M. Kang, D. Park, and S. Yoon are with the School of Computing, Korea Advanced Institute of Science and Technology, South Korea;{minsung.yoon, mincheul.kang, daehyung}@kaist.ac.kr, and sungeui@kaist.edu.
[‡]D. Park and S. Yoon are co-corresponding authors.

II. RL-BASED TRAJECTORY GENERATION

A. Notations

Researchers often represent the path in Cartesian space as a sequence of poses $X = [x_0, x_1, ..., x_{N-1}] \in \mathcal{X}$ evenly spaced in time, where N is the number of poses in the path, $\mathcal{X}$ is the space of paths, and each pose is a pair of position ($\in \mathbb{R}^3$) and orientation ($\in SO(3)$). Likewise, the joint trajectory is a sequence of joint configurations $\xi = [q_0, q_1, ..., q_{N-1}] \in \Xi$, where $q \in \mathbb{R}^d$ is a configuration of a d DoF manipulator and Ξ is the Hilbert space of joint trajectories. The d is greater than 6 in the case of redundant manipulators in $SE(3)$.

B. Formulation of MDP

We first formulate a path-conditioned MDP as $\mathcal{M}_X = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}_\mathcal{Z}, \mathcal{T}, \mathcal{Q}_0, \gamma \rangle_{X \sim \mathcal{P}(X)}$, where X is a target path sampled from a distribution of paths $\mathcal{P}(X)$, $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R}_\mathcal{Z} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a time-varying reward function, where $\mathcal{Z} = \{z \in \mathbb{N}_0 | 0 \leq z \leq N-1\}$ is a set of time steps, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a deterministic transition function, $\mathcal{Q}_0$ is a set of start configurations q_0, and $\gamma \in [0, 1)$ is a discount factor. We sample the target paths X within a restricted operation range with the arm's length in the task space. In the case of $\mathcal{Q}_0$, we sample IK solutions at the first pose x_0 in X. The policy trained on the MDP $\mathcal{M}_X$ synthesizes a high quality trajectory ξ by sequentially expanding the trajectory in a direction that satisfies the constraints. To generalize the policy over the path and obtain a unified policy, we define a multi-path RL objective function:

$$\underset{\pi}{\text{maximize}} \ \mathbb{E}_{X \sim \mathcal{P}(X)} \left[\mathbb{E}_{\substack{(s_i, a_i) \sim \rho_\pi \\ q_0 \sim \mathcal{Q}_0}} \left[\sum_{i=0}^{N-1} \gamma^i \cdot \mathcal{R}_i(s_i, a_i) \right] \right], \quad (1)$$

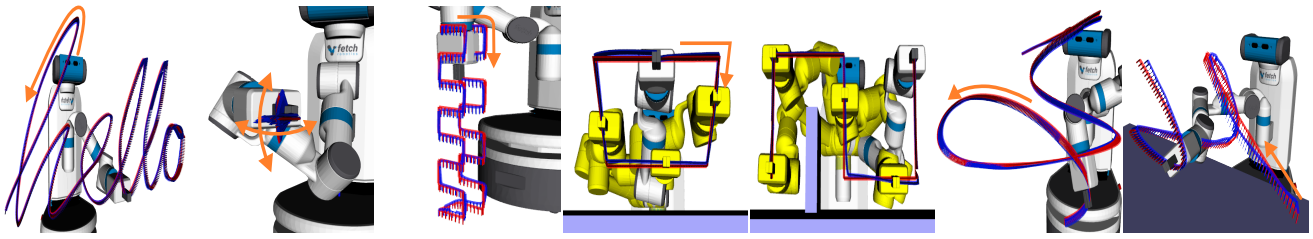

Fig. 1. Visualization of five specific and two exemplar random target paths (red lines) used in evaluations. Orange arrows indicates the progress direction of the path. Blue lines are the end-effector paths calculated from *RL-TG*'s joint trajectories via forward kinematics. In Square and S problems, the original color of the robot represents the initial configurations, and the Yellow trails indicate that the generated trajectories satisfy collision-avoidance constraints.

where ρ_π is the trajectory distribution given the deterministic transition function $\mathcal{T}$ and a stochastic policy $\pi(a_i|s_i)$. Then, we find an optimal policy π^* where $\pi^* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ maximizes the objective function.

1) State and Action space: As the state space $\mathcal{S}$, we consider the 3D occupancy grid for recognizing the surrounding environment, the joint values and the poses of links of the robot, and the poses up to 6 steps ahead of the target paths for near-sight behavior. We also define an action as a configuration difference, $a_i = \Delta q_i \in \mathbb{R}^d$, and $q_{i+1} = q_i + \Delta q_i$ given the deterministic $\mathcal{T}$. Therefore, the policy sequentially extends the trajectory for every step to compose the whole trajectory of length N.

2) Reward formulation: A multi-objective reward function is divided into three main terms. First, a task reward is to encourage the agent to follow the target path X. Second, an imitation reward is to make the policy learn the optimized null-space motion depicted in the demonstration. The last reward term is the constraint-related reward function that penalizes collision, joint-limit violation, singularity condition, and early termination states to satisfy constrains.

III. Results

We prepared five specific ('Hello', 'Rotation', 'Zigzag', 'Square' and 'S') and one random target path benchmark set to show generalizability over external environments and paths. As baselines, *Linear* returns a linearly interpolated trajectory in joint space, *Greedy* [3] efficiently uses IK solutions, and *BC-TG* is the same as ours except learning method with supervised learning. We call our method as *RL-TG*.

Fig. 1 shows the trajectories synthesized by *RL-TG* for each exemplar benchmark problem. Fig. 2 shows the comparative analysis of the trajectory generation methods in terms of four quality metrics: a path-following pose error and a trajectory smoothness to measure a trajectory quality, a constraint violation rate considering the collision-free and joint velocity limit violation constraints, and a generation time. *RL-TG*, a learning-based function approximation method, has a slight error with the target path when looking at the end-effector path of the generated trajectory, but shows great advantages in null-space continuity (joint smoothness) and generation time, which leads to a low constraint violation rate. In Fig. 2-(d), the reason why the *Linear*'s constraint violation is zero is that the generated trajectory does not move at all since the start and end poses of the paths are the same.

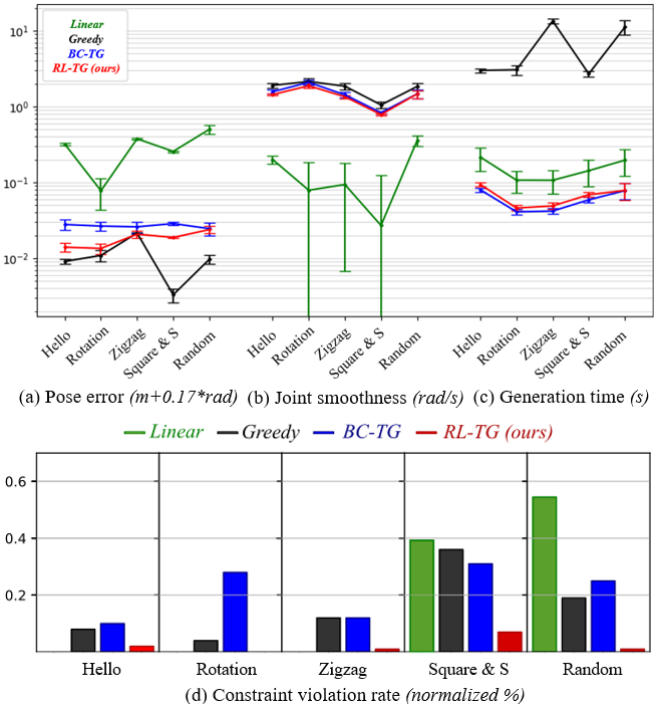

Fig. 2. Comparative analysis of the four trajectory generation methods in five types of simulated environments. The x- and y-axes are the type of benchmark problems and the performance metrics, respectively.

IV. Conclusion

We presented a reinforcement learning-based trajectory generation (*RL-TG*) method that quickly finds a low-cost trajectory of redundant manipulators for path-following problems. We have shown the lower constraint violation rate and fast generation time of *RL-TG* qualitatively and the generated trajectories qualitatively in simulation experiments. One future direction is to post-process the generated trajectories using trajectory optimization techniques to guarantee the generated trajectories' feasibility and to reduce the path-following error.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A4A1032582).

References

[1] Patrick Beeson and Barrett Ames, "TRAC-IK: An open-source library for improved solving of generic inverse kinematics", in *IEEE International Conference on Humanoid Robots*. IEEE, 2015, pp. 928–935.

[2] Rachel Holladay, Oren Salzman, and Siddhartha Srinivasa, "Minimizing task-space frechet error via efficient incremental graph search", *RA-L*, vol. 4, no. 2, pp. 1999–2006, 2019.
[3] Mincheul Kang, Heechan Shin, Donghyuk Kim, and Sung-Eui Yoon, "TORM: Fast and accurate trajectory optimization of redundant manipulator given an end-effector path", in *IROS*. IEEE, 2020, pp. 9417–9424.