Using Track-IK for Guiding MPC

January 22, 2024

1 Using Track-IK for Guiding MPC

We employ multi-processing using Track-IK to guide MPC out of local minima and simultaneously optimize its operability. This method continuously generates IK solutions $q_{\text{des},t} = \text{Track\_IK}(q_t, X_g)$ based on current joint states q_t and the target Cartesian pose $X_g(p_g, R_g)$. Based on this result, we design a joint-space goal cost dist_{joint}:

$$\text{dist}_{joint} = \|q_t - q_{\text{des},t}\|_2 \tag{1}$$

In cases where Track-IK is infeasible or the target pose $X_g(p_g, R_g)$ is outside the robot's workspace, we design a Cartesian-space goal cost dist_{cart}:

Given the current robot end-effector pose $X_{ee}(p_{ee}, R_{ee})$:

$$\text{dist}_{cart} = w_1 \|\mathbf{p}_{ee} - \mathbf{p}_g\|_2 + w_2 \cos^{-1}\left(2\langle\hat{\mathbf{q}}_{ee}, \hat{\mathbf{q}}_g\rangle^2 - 1\right) \tag{2}$$

Where $\mathbf{p}_{ee}, \mathbf{p}_g$ are the position vectors and $\hat{\mathbf{q}}_{ee}, \hat{\mathbf{q}}_g$ are the corresponding quaternions representing the rotation matrices R_{ee}, R_g.

While the Euclidean norms $\|.\|_2$ in dist_{joint} and dist_{cart} guide the robot globally towards the target pose, they exhibit low sensitivity near the goal, which can result in large errors upon approach.

we integrate it with sparse reward tricks to minimize end-pose error:

$$\text{SparseReward}\ (\text{dist}_{\text{joint|cart}}) = 1 - \exp\left(-\,\text{dist}^2\,/\,(2\sigma^2)\right) \tag{3}$$

Where dist can be dist_{joint} or dist_{cart}, and σ is the convergence radius.

When dist is within the radius σ, SparseReward rapidly decreases costs, driving sampled trajectories to quickly converge in the target region, whereas outside the radius, it has no guiding effect, allowing the robot to automatically slow down near the target location and reduce reach errors.

2 Robot Dynamic Constraints

Using the approach from the Paper[STORM], Robot dynamics constraints $C_{safe\_constraints}$ are designed to protect the safe operation of the robotic arm. This requires considering: Joint Position Bound Limits C_{bound_limit}, Joint velocity constraint

C_{stop_vel} and self-collision avoidance $C_{self\_collision}$ which uses JointNERF to accelerate collision query.

$$C_{safe\_constraints} = C_{bound\_limit} + C_{stop\_vel} + C_{self\_collision} \tag{4}$$

3 Differ Policy [Greedy/Sensitive/Judge]

The Greedy policy aims to reach the target position quickly with lower sensitivity to collisions, and assigns a higher weight to the goal-related cost functions:

$$C_{greedy}(x_t, u_t) = w_{11} dist_{joint|cart} + w_{21} SparseReward(dist_{joint|cart}) + w_{31} Coll_p + C_{safe-constraints}$$

The Sensitive policy is inclined to avoid collisions, exhibiting higher sensitivity to obstacles, and assigning a larger weight to collision-related cost functions:

$$C_{sensitive}(x_t, u_t) = w_{12} dist_{joint|cart} + w_{32} Coll_{ppv\theta} + C_{safe-constraints}$$

the reason $C_{sensitive}$ does not integrate SparseReward is that when the obstacle moves within the target area, the sampled trajectories do not converge in that region, thereby enhancing its exploration capability for obstacle avoidance. This design also indirectly affirms the robust exploratory performance of Parallel MPPI.

The Judge policy focuses solely on the trajectories' distance to the target and their collision status. Thus, the robot dynamics constraints are omitted, and the associated weights are set between those of the Greedy and Sensitive policies:

$$C_{judge}(x_t, u_t) = w_{13} dist_{joint|cart} + w_{23} SparseReward(dist_{joint|cart}) + w_{33} Coll_{ppv\theta}$$

Remark: There is a misunderstanding that parallel MPPI is meaningless, and the same result can be achieved by continuously tuning the weights of single-layer MPPI. However, theoretical analysis suggests that the exploration capability of Parallel MPPI is significantly higher than that of the single-layer MPPI, without sacrificing the convergence of the algorithm. Experimental results and robot motion characteristics indicate that parallel MPPI can adaptively adjust its policy weights based on working conditions: switching to the Sensitive policy to avoid collisions when obstacles are approaching and leaning towards the Greedy policy to accelerate towards the target when obstacles are far away. This motion characteristic, combining the advantages of each strategy, minimizes collisions while striving to complete the task as quickly as possible.