Nonlinear Trajectory Optimization for Multiple Robot Arms

Philip Huang[1] and Megan Lee[1]

***Abstract*—The use of multiple robot arms is increasingly crucial in industrial automation for tasks requiring coordination and collaboration, such as assembly, rearrangement, and inspection. However, planning collision-free, dynamically feasible trajectories for multiple arms sharing a workspace remains challenging. This paper presents a nonlinear trajectory optimization framework for multiple robot arms that generates smooth, time-optimal motions while satisfying kinodynamic constraints. We compare two optimization approaches: Direct Collocation (DIRCOL) with the IPOPT solver and Augmented Lagrangian Trajectory Optimization (ALTRO) with an iLQR solver. Experimental results on both 2D kinematic chains and 3D dual 6-DOF industrial robots demonstrate our framework's ability to efficiently generate collision-free, dynamically feasible trajectories. The key advantages of our approach include time optimization, smooth motion profiles for practical robot execution, and the ability to initialize with kinematically feasible but dynamically infeasible trajectories.**

I. Introduction

Multi-robot arm motion planning (M-RAMP) has recently seen an increased research interest, especially with the rise of bimanual manipulation systems. Fig. 1 shows several examples of multi-arm motion planning and coordination for tasks such as rearrangment and assembly. This trend is partly driven by the crucial role M-RAMP plays in multi-arm manipulation. The simultaneous use of multiple robot arms facilitates the automation of complex tasks infeasible with a single arm, such as collaborative robotic assembly, while also improving the efficiency of tasks typically performed by a single arm, such as pick-and-place operations. Among recent approaches to M-RAMP, robot arm trajectories are often computed with a sampling-based algorithm derived from RRT [1], for example work by [2], or variants of A* search [3] such as in [4] or [5].

Although these sampling-based algorithms have proven effective in finding feasible paths, they often struggle to incorporate dynamic constraints and generate smooth trajectories suitable for robot execution. Additionally, in multi-robot settings, the coordination between arms introduces additional complexity that purely geometric planners do not address optimally. Industrial applications demand not only collision-free paths but also time-efficient trajectories that respect the joint velocity and acceleration limits of the hardware while minimizing jerk for smooth operation.

In this paper, we set up M-RAMP as a constrained optimization problem that will allow for inter-robot correlation to a specified task while considering the kinodynamic constraints. This was developed as an offline planning framework as we explore ways to improve convergance speed. For example, we leverage warm-start techniques to initialize the problem from kinematically feasible but potentially dynamically infeasible trajectories, significantly improving convergence rates.

[1]Authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA

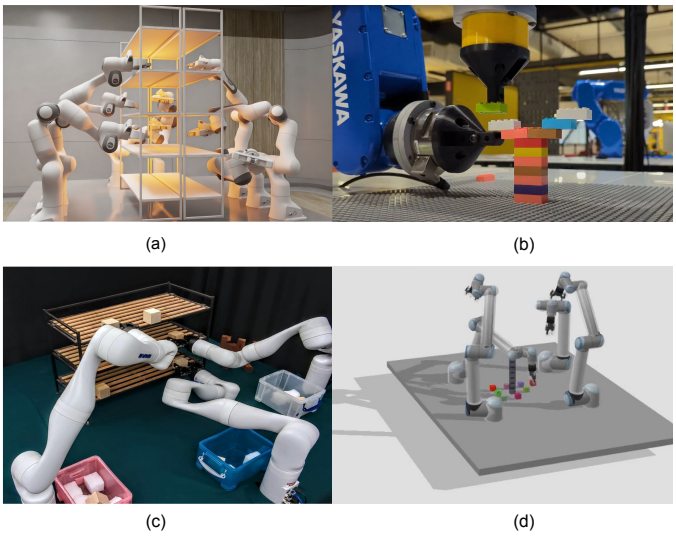

Fig. 1. Examples of multi-arm motion planning and coordiatnion tasks in literature. (a): motion planning tasks with six Panda arms around shelves in [4], (b): a collaborative, dual arm LEGO assembly environment in [8] (c): a real world shelf rearrangement task in [4] (d): a block rearrangement task in [9]

Our contributions in this paper are of three parts. First, we implement a differentiable collision detection framework that efficiently computes distances between robot links represented as geometric primitives, providing smooth gradients for optimization. Second, we utilize direct collocation (DIRCOL) [6] and Augmented Lagrangian Trajectory Optimization (ALTRO) [7]for solving the multi-robot motion planning problem. Finally, we validate our approach through experiments on various test cases, demonstrating its effectiveness in generating time-optimal and dynamically feasible trajectories for multiple robot arms working in coordination.

II. Related Work

A. Multi Robot Arm Motion Planning

Given N robots sharing a workspace $\mathscr{W} \subset \mathbb{R}^3$, let $\mathscr{C}^i$ be the configuration space of each robot i with DoF^i degrees of freedom (DoF). For robot arms with DoF^i joints, a configuration $C^i \in \mathscr{C}^i \subseteq \mathbb{R}^{\text{DoF}^i}$ defines a selection of joint angles. Given a set of start and goal configurations, a pair for each robot i, M-RAMP focues on finding a set of collision-free *trajectories* $\tau := \{\tau^1, \cdots \tau^N\}$, one for each robot from its start to goal. We represent a single-robot trajectory with a sequence of H^i timestamped configurations $\tau^i = \{\tau^i_j\}_{j=1}^{H^i} = \{(C^i_j, t^i_j)\}_{j=1}^{H^i}$, with C^i_j being the configuration of robot i at time t^i_j. In many related

work [4], [10], the objective of M-RAMP is focused on kinematic path planning. Typically, each step (C^i_j, t^i_j) to (C^i_{j+1}, t^i_{j+1}) is over a constant time interval Δt and spatially defined as a linear interpolation between the two configurations. Then, the objective is often minimizing travel time or cumulative motion, while satisfying collision constraints, i.e., no robot-robot or robot-world collisions.

Researchers have proposed various approaches to multi-robot motion planning and applied them to M-RAMP, including directly using single-robot planners, such as RRT-Connect [11] or BIT* [12], in the composite configuration space $\mathscr{C} := \mathscr{C}^1 \times \cdots \times \mathscr{C}^N$. Other approaches more tailored towards M-RAMP include building composite roadmaps from the Cartesian product of individual roadmaps [13], coordinating the speed of individually planned motions [14], [15], and using prioritized planning [16]. Researchers have also studied more long-horizon manipulation tasks with multiple robot arms, such as object pick and place [17]–[19] and object rearrangement and assembly [5], [20]–[23]. A key difference between these work and ours is that we focus on satisfying kinodynamic constraints as well as collision-free constraints, i.e., the robot trajectory should be smooth and kinematically fasible with bounded velocity, acceleration, and jerk. Many of the multi-robot motion plans surveyed here also use post-processing to improve their motion plan obtained.

B. Trajectory Optimization for Robot Arms

Optimization-based trajectory planners have become central to robot arm motion planning, encoding smoothness, dynamics, and collision avoidance into unified cost functions. Existing methods such CHOMP (Covariant Hamiltonian Optimization for Motion Planning) use functional gradients to iteratively refine a trajectory, trading off smoothness against obstacle costs [24]. Building on this, convexification approaches such as TrajOpt employ sequential convex optimization with hinge-loss penalties to enforce collision constraints, achieving faster planning times and improved path quality compared to CHOMP or sampling-based planners for high-DOF manipulators [25]. Similarly, Gaussian process motion planners (GPMP/GPMP2) represent trajectories as sparse continuous-time samples and recast planning as probabilistic inference. GPMP2, for example, interleaves GP interpolation with structure-exploiting optimization and incremental replanning, yielding solutions much faster than earlier planners while maintaining robustnes [26]. These methods have been extensively validated on complex robots (e.g. 7-DOF arms, humanoids, dual-arm PR2), often demonstrating real-time or near real-time performance in cluttered environments. In practice, modern frameworks (e.g. MoveIt!) integrate variants of these optimizers to handle full-body motion and dynamic constraints, enabling smooth, collision-free trajectories in industrial applications.

More recent work extends these ideas to multi-robot coordination and dynamic environments. Coordinating multiple manipulators requires handling inter-robot collisions and synchronization. For example, Salehian et al. [27] introduce a dynamical-systems control law for dual-arm robots to adaptively intercept moving objects in uncertain settings, demonstrating synchronized re-planning on a physical multi-arm system. Distributed methods like consensus ADMM have been proposed to plan trajectories for each agent with collision avoidance; a convex C-ADMM formulation can dramatically speed convergence in multi-robot waypoint navigation, whereas naive non-convex approaches may violate safety constraints [28]. Similar to our work, Zimmermann et al. [29] formulate distance computations between complex shapes as differentiable optimization problems, enabling robot arm and even mobile manipulator path optimization with Newton-method based optimization. Dynamic obstacle scenarios often use fast re-planning or learning: GPMP2's incremental variant (iGPMP2) [30] replans in a fraction of the time of fresh planning, and learning-based planners (e.g. DRL with safety filters) can produce near-optimal manipulator paths in milliseconds. Among these planners that accommodate multiple robots and moving obstacles, our work focuses on solving the dynamically feasible motion planning problem with multiple robot arms.

C. Nonlinear Trajectory Optimization Solver

On the topic of nonlinear trajectory optimization solvers, DIRCOL tends to be more robust and fast [6]. Direct methods discretize both states and controls at a set of collocation points. The dynamics constraints are enforced through collocation, resulting in a large but sparse constraint structure that can be exploited by NLP solvers. They use solvers such as IPOPT [31] or SNOPT [32] and can be initialized with a dynamically infeasible trajectory. The downside of DIRCOL is the convergence time in which indirect methods such as DDP and iLQR have an advantage. However, DDP and iLQR [33] are unable to initialize with infeasible state trajectories. This led us to Augmented Lagrangian Trajectory Optimization (ALTRO) [7] which aims to combine the best of direct and indirect methods like numerical robustness, speed, and infeasible trajectory initialization. However, DIRCOL is still more reliable for time-penalized problems which is one of the aims of this paper.

D. Collision Detection

For collision detection and avoidance, we leverage DCOL (Differentiable Collision Detection for a Set of Convex Primitives) [34], a fully differentiable collision detection framework that handles six types of convex primitives: polytopes, capsules, cylinders, cones, ellipsoids, and padded polygons. The differentiability of DCOL allows us to directly incorporate collision constraints into gradient-based trajectory optimization, allowing our planning framework to efficiently generate collision-free trajectories without heuristic approximations of collision gradients.

III. Problem Formulation

Given the initial and goal states of all of the arms, we formulate M-RAMP as a constrained trajectory optimization problem that is dynamically feasible and collision free for

a single task. The trajectory optimization problem is the following:

$$\min_{x_{0:N},u_{0:N-1}} \quad \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k,u_k) \tag{1}$$

$$\text{subject to} \quad x_{k+1} = f(x_k,u_k) \tag{2}$$

$$x_o = x_{init} \tag{3}$$

$$x_g = x_{goal} \tag{4}$$

$$x_{min} \leq x_k \leq x_{max} \tag{5}$$

$$u_{min} \leq u_k \leq u_{max} \tag{6}$$

$$\text{prox}(P_1^i, P_2^j) \geq \text{threshold}, \quad \forall i,j \in \{1,..,N_L\} \tag{7}$$

where k is the time step index, ℓ_N is the terminal cost, ℓ_k is the stage cost, x_k are the states, u_k are the control inputs, and $f(x_k,u_k)$ are the discrete dynamics. To eleaborate on the constraints of the problem, the first three are equality constraints where (2) ensures that the solution is dynamically feasible, while (3) and (4) ensures that the trajectory respects the given initial and goal states. For the inequality constraints, (5) sets the bounds on the state vector, (6) sets the control bounds, and (7) represents the minimum distance allowed between robot links to avoid collisions.

For each time step, we use a triple integrator model where:

$$\text{where} \quad x_k = \begin{bmatrix} q_k^1 \\ \dot{q}_k^1 \\ \ddot{q}_k^1 \\ q_k^2 \\ \dot{q}_k^2 \\ \ddot{q}_k^2 \end{bmatrix}, \quad u_k = \begin{bmatrix} \dddot{q}_k^1 \\ \dddot{q}_k^2 \end{bmatrix} \tag{8}$$

The state vector includes joint positions, velocities, and accelerations for robot 1 and 2. The control vector includes the jerk. This representation allows us to generate smooth trajectories with continuous acceleration profiles, which is desirable for robot motion execution. Joint limits for each position, velocity, and acceleration are all included in the state inequality (5), where as the jerk limit is included in (6).

IV. Method

To solve the trajectory optimization problem, we model the robot links as a geometric primitive using the DCOL [34] library. Then, we implement and compare DIRCOL which uses the IPOPT solver and ALTRO with an iLQR solver. Both were implemented in the Julia programming language. Below, we will eloaborate the deail for geometry modeling of manipulator and the math for trajectory opitmization below.

A. Modeling for Collision

A key requirement for trajectory optimization is how to model collisions for robot arms. In typical robot motion planning, a high-fidelity mesh model is used for collision checking using the Gilbert–Johnson–Keerthi(GJK) distance algorithm. This is typically provided in modern motion planning library via tools such as FCL or bullet. However, DCOL do not provide the capability for mesh-mesh collision checking. Thus,

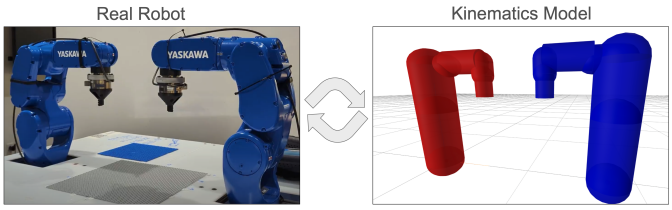

Fig. 2. Real robot of two GP4 robot vs our kinematic model with shape primitives

we use simplified primitives (cylinders, capsules, spheres) to represent the collision geometry of each link l_i of the robot arm. While this involves a hand conversion, it is flexible and allow us to utilize the differentiable collision capabilities of DCOL.

In our experiment, we use two Yaskawa GP4 as our setup. Each robot arm is a 6-dof industrial arm. We model each link as either a capsule or cylinder. Fig. 2 shows an example view of the geometric model rendered in Meshcat versus the real robot images.

To compute the collision constraint, we compute the pose of each link l_i with forward kinematic. This is performed with the power of exponential formula. Mathematically, we define the following notation:

- $\theta = [\theta_1, \ldots, \theta_6]^T$: Vector of joint angles.
- $\mathcal{S}_i \in \mathbb{R}^6$: Spatial twist of joint i in base frame.
- $\hat{\mathcal{S}}_i \in \mathbb{R}^{4\times4}$: Matrix representation (in $\mathfrak{se}(3)$) of $\mathcal{S}_i$.
- $e^{\hat{\mathcal{S}}_i\theta_i} \in SE(3)$: Exponential map of the twist.
- $M_i \in SE(3)$: Home configuration (pose at zero joint angles) of link i.
- $T_i(\theta) \in SE(3)$: Pose of link i as a function of joint angles.
- $p_i \in \mathbb{R}^3$: Position of link i's origin in base frame.
- $p_j \in \mathbb{R}^3$: Position of joint j's axis in base frame.
- $\omega_j \in \mathbb{R}^3$: Unit rotation axis of joint j, expressed in base frame.

We use the product of exponential forward kinematic equations to compute the pose of each link $T_i(\theta)$

$$T_i(\theta) = e^{\hat{\mathcal{S}}_1\theta_1} e^{\hat{\mathcal{S}}_2\theta_2} \cdots e^{\hat{\mathcal{S}}_i\theta_i} M_i \tag{9}$$

In order to fully make use of the power of DCOL in our trajectory optimization algorithm, we also need to make use of the forward kinematic gradient to backpropagate the constraint gradient to our state x. Hence, we compute the following gradient of T_i w.r.t. joint θ_j $(j \leq i)$:

$$\frac{\partial p_i}{\partial \theta_j} = \omega_j \times (p_i - p_j) \tag{10}$$

The gradient of orientation of link i R_i w.r.t. joint θ_i $(j \leq i)$ is

$$\frac{\partial R_i}{\partial \theta_j} = [\omega_j]_\times R_i \tag{11}$$

$$\text{where } [\omega_j]_\times = \begin{bmatrix} 0 & -\omega_{j,z} & \omega_{j,y} \\ \omega_{j,z} & 0 & -\omega_{j,x} \\ -\omega_{j,y} & \omega_{j,x} & 0 \end{bmatrix} \tag{12}$$

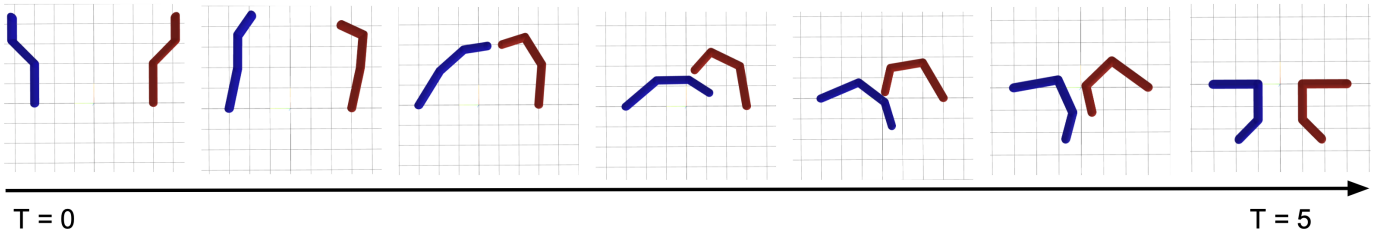

Fig. 3. An optimized trajectory of the 2D, 3-link robot arm.

This computes the gradient of each link i with respect to the revolute joints j.

DCOL determines collision information between all pairs of primitives between robots as well as primitives within the same robot that can self collide. It achieves this by formulating an optimization problem to solve for the smallest uniform scaling of two object primitives that leads to an intersection. This is achieved by formulating an optimization problem where the following are the primal variables:

$$\min_{p,\alpha} \quad \alpha \tag{13}$$

$$\text{subject to} \quad \alpha \geq 0, \tag{14}$$

$$p \in P_1(\alpha), \tag{15}$$

$$p \in P_2(\alpha) \tag{16}$$

where α is the minimum scaling, and p is the intersection point that are in both scaled primitive $P_1(\alpha)$ and $P_2(\alpha)$. This is a conic problem and solution of the optimization can be differentiable with respect to its parameters. Finally, we impose the constraint $alpha \geq 1$ for all pairs of robot link that can collide and use the chain rule to compute the gradient of this constraint with respect to the joint state.

B. Simple ALTRO

ALTRO combines the speed of iterative LQR (iLQR) with the constraint handling capabilities of an augmented Lagrangian approach, making it particularly well-suited for our constrained trajectory optimization problems. ALTRO tends to be more numerical robust and allows us to initialize with dynamically infeasible trajectories. This method comprises two main stages: 1) an Augmented Lagrangian iLQR (AL-iLQR) that quickly solves to handle the general state and input constraints 2) direct projection for high precision constraint satisfaction.

We implement a simpler version of ALTRO where we still use AL-iLQR to incorporate constraints into the cost. The Augmented Lagrangian are as follows:

$$\begin{aligned}\mathscr{L}_A =& \ell_N(x_N) + (\lambda_N + \frac{1}{2} I_{\mu_N} c_N)^T c_N \\ &+ \sum_{k=0}^{N-1} [\ell_k + (\lambda_k + \frac{1}{2} I_{\mu_k} c_k)^T c_k]\end{aligned} \tag{17}$$

where λ_k are the Lagrange multipliers, μ_k are the penalty weights, and $c_k(x_k, u_k)$ represents the constraint violations. I_{μ_k} is a diagonal active-set matrix that selectively applies penalties only to active constraints.

Our approach incorporates time optimization by including a time scaling variable h_k in the control vector $u_k = [h_k; \ddot{q}_k^1; \ddot{q}_k^2]$, where j_k^i represents the jerk inputs for robot i During the backward pass, the optimal control policy is computed as:

$$\delta u_k^* = -(Q_{uu} + \rho I)^{-1}(Q_{ux}\delta x_k + Q_u) \tag{18}$$

which yields the feedback policy K_k and feedforward term d_k. After each iLQR iteration, dual variables are updated:

$$\lambda_{k,i}^+ = \begin{cases} \lambda_{k,i} + \mu_{k,i} c_{k,i} & i \in \mathscr{E} \\ \max(0, \lambda_{k,i} + \mu_{k,i} c_{k,i}) & i \in \mathscr{I} \end{cases} \tag{19}$$

where $\mathscr{E}$ are equality constraints and $\mathscr{I}$ is inequality constraints.

Unlike the complete ALTRO implementation described in the paper [7], we omitted the full square-root backward pass formulation and the sophisticated objective-weighted Newton projection method. Instead, we directly handle collision constraints using differentiable proximity functions from DCOL, defined as $c_{collision}(x_k) = \text{threshold} - \text{prox}(P_1^i, P_2^j)$, which is computationally efficient for our dual-robot planning scenario.

C. DIRCOL

Due to the inherent sensitivity to initial time steps in the shooting methods used in ALTRO, DIRCOL is more reliable when it comes to time-penalizing problems. Since we want to explore a smooth yet fast trajectory, we implement DIRCOL using IPOPT and added time as a control variable. The cost function is modified to include time h_k:

$$\begin{aligned}J = \sum_{k=1}^{N-1} u_k[1] \Big(& \frac{1}{2}(x_k - x_{\text{ref},k})^T Q (x_k - x_{\text{ref},k}) \\ & + \frac{1}{2}(u_k[2:] - u_{\text{ref},k}[2:])^T R_{2:,2:}(u_k[2:] - u_{\text{ref},k}[2:]) \Big) \\ & + \frac{1}{2}(x_N - x_{\text{ref},N})^T Q_f (x_N - x_{\text{ref},N})\end{aligned} \tag{20}$$

The control vector is now $u_k = [h_k; \ddot{q}_k^1; \ddot{q}_k^2]$ where $h_k = u_k[1]$ represents the time cost. The dynamics of the system are also modified to incorporate the time scaling variable h_k. The time scaling acts as a multiplier for all state derivatives:

$$\dot{x} = h_k \cdot f(x, u[2:]) \tag{21}$$

TABLE I
SOLVER PERFORMANCE FOR MINIMUM TIME TRAJECTORY OPTIMIZATION PROBLEMS

Method	Start	Total	Success	Rate(%)	Runtime (Avg±Std) (s)	Trajctory Duration (Avg±Std) (s)	Length (Avg±Std) (rad)
DIRCOL	Warm	271	271	100.0	0.857 ± 2.319	3.57 ± 3.49	4.56 ± 2.47
	Cold	271	271	100.0	0.681 ± 1.718	3.37 ± 3.08	4.31 ± 1.33
ALTRO	Warm	271	243	89.7	4.443 ± 5.804	4.83 ± 1.72	4.29 ± 1.35
	Cold	271	243	89.7	4.458 ± 5.837	4.83 ± 1.72	4.29 ± 1.35

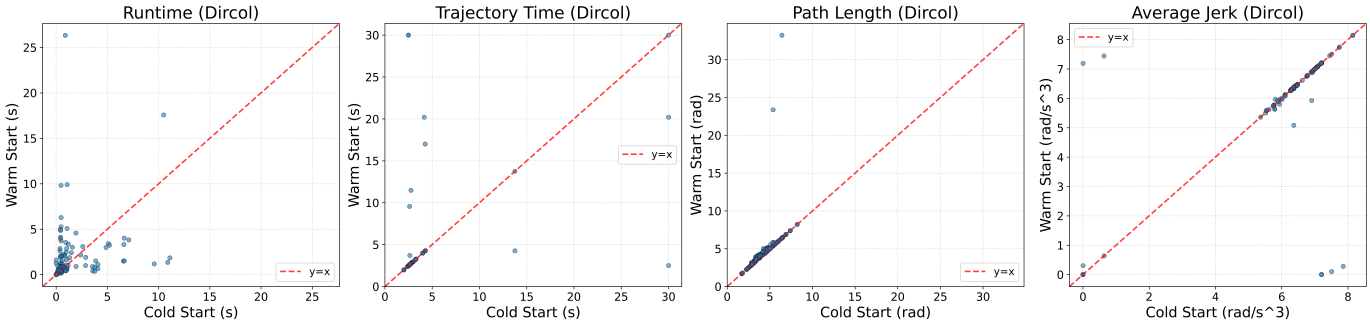

Fig. 4. Scatter plot comparison of warm starting versus cold starting of all instances for DIRCOL trajectory optimization. Runtime is the time for the DIRCOL optimizer to converge. Trajectory time is the minimized execution time of the final solution. Path length is the final path length of the final solution, measured in average L2 distance over all 6 joints. Lastly, the average jerk is the mean of all state input across all time steps.

This formulation allows the optimizer to directly control the evolution rate of the system. For our dual-robot system with triple integrator dynamics, this becomes:

$$\begin{bmatrix} \dot{q}^1 \\ \dot{q}^2 \\ \ddot{q}^1 \\ \ddot{q}^2 \\ \dddot{q}^1 \\ \dddot{q}^2 \end{bmatrix} = h_k \cdot \begin{bmatrix} \dot{q}^1 \\ \dot{q}^2 \\ \ddot{q}^1 \\ \ddot{q}^2 \\ u[2:(2+N_j)] \\ u[(2+N_j+1):end] \end{bmatrix} \tag{22}$$

Where values of $h_k < 1$ accelerate the trajectory execution and values of $h_k > 1$ decelerate it. The bounds are set to 0.3 seconds as a minimum so that the solution doesn't try to go to an infeasible time, and 5.0 seconds as a maximum.

V. RESULTS

A. 2D Toy Example

To evaluate our method, we first setup a simple toy problem with two 2D, 3-link kinematic chain. The forward kinematic for this example is much simpler to debug and implement in a planar environment, and we use it to quickly evaluate our differentiable collision and trajectory optimization pipeline.

Fig. 3 shows an example trajectory of 2D toy problem. The two chained links has to swap their orientation from facing up to bottom. We use the simple ALTRO as our optimizer and use naive intpolation between the start and goal pose as the initial guess. Although the initial guess would lead to a collision, the trajectory optimizer is able to successfully find a collision-free trajectory, where the red robot would retract its third link, allowing the blue link to pass first to its goal pose. This validates the effectiveness of trajectory optimization with differentiable collision for two simple kinematic chains.

B. 3D Dual Robot Arm

We then evaluate our ALTRO and DIRCOL method on 272 instance of dual arm manipulation problem. The robot kinematic and collision geometry is setup following Sec. IV-A. Fig. 5 shows two examples of the optimized trajectory. Qualitatively, the trajectory of the robot arm often moves very close to each other, opting for a shorter path while narrowly avoiding collision.

Quantitatively, Table I shows the performance of the minimum time trajectory optimization for both DIRCOL and ALTRO. First, we note that DIRCOL has a 100% success rate, meaning the optimizer successfully converged. DIRCOL may also finds a more aggressive (faster) solution, with a shorter trajectory duration (measured in time) on average, while the path length is similar. On the other hand, ALTRO was unable to find a solution within 3000 iterations and failed for some instances. We also noticed the runtime difference, where DIRCOL seems to be converging much faster. However, the convergence time also depends on other things, such as the number of knot points. In fact, we used a much smaller N for DIRCOL, and this considerably improves the runtime.

Another important evaluation we tried is warm starting the trajectory with a collision-free path generated by xECBS [4], a multi-robot arm path planner. The initial trajectory may not be dynamically feasible but is synchronized and collision-free. In Table I as well as empirically, we noticed minimal difference between the quality of the converged trajectory. In fact, a warm start to the trajectory can sometimes hurt the performance.

To further dissect whether the result is skewed by some hard instances, we plot four scattered plots to compare the performance of warm versus cold starting across all 272 instances in Fig. 4. For most instances, the quality of the final

trajectory is usually the same (falling on the dotted red line $y = x$) with few outliers, but the variance with run-time is more significant. This suggests that the IPOPT optimizer is not making good use of the initial guess.

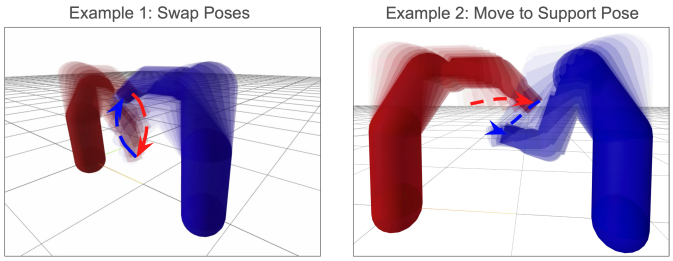

Fig. 5. Two optimized trajectory of the dual GP4 robot arm. The intermediate poses are displayed in shadow, and the trajectory of the robot end-effector are shown in dashed line.

VI. Conclusion and Future Work

In this work, we presented a nonlinear trajectory optimization framework for planning collision-free, dynamically feasible, and time-optimal motions for multiple robot arms sharing a workspace. Our approach leverages differentiable collision detection using geometric primitives, enabling the incorporation of collision constraints directly into the optimization problem with smooth gradients. We implemented and compared two prominent nonlinear optimization methods, Direct Collocation (DIRCOL) using IPOPT and a simplified Augmented Lagrangian Trajectory Optimization (ALTRO) using an iLQR solver.

Our experimental results on both 2D kinematic chains and realistic 3D dual 6-DOF industrial robot arms demonstrate the framework's capability to generate smooth trajectories that respect kinodynamic limits while avoiding inter-robot and robot-world collisions. The inclusion of a time optimization variable allows the framework to find time-efficient solutions, crucial for industrial applications. We found that DIRCOL exhibited robust convergence properties, achieving a 100% success rate on our benchmark of 272 dual-arm planning instances, and generally produced faster trajectories compared to our simple ALTRO implementation. While warm-starting the optimizer with kinematically feasible paths did not significantly improve convergence time or final trajectory quality for DIRCOL in this benchmark, the framework remains capable of starting from dynamically infeasible initial guesses, a key advantage for practical use.

While the presented work provides a good starting point for applying nonlinear trajectory optimization to complex multi-robot corrdinaton tasks, many avenues for future work still exists. First, the scalability of the approach to a larger number of robots could be explored. As the number of robots increases, the dimensionality of the state space and the number of collision pairs grow combinatorially. Naively scaling the current framework by treating all robots as a single agent may not scale well. Finding collision-free trajectory also becomes more difficult with growing number of robots in a confined space. Second, a more thorough implementation and evaluation of our algorithm, including the ALTRO direct projection phase and the full effect of warm starting, may be helpful. Therotically, optimization-based method should benefit from a good initial guess, but our experimental result suggests that is not guaranteed. Lastly, in order to enable online execution of our optimizer on real robot, investing whether a short-horizon trajectory optimization can be used within a model-predictive controller is another interesting direction. This requires the solver to converge to a dynamically feasible and collision-free path, and adjust to online disturbances quickly within a time limit, which can be nontrivial.

Overall, we believe this work provides a foundation for applying nonlinear trajectory optimization to complex multi-robot coordination tasks, producing high-quality trajectories suitable for direct execution on robotic hardware. Our implementation is available at https://github.com/meganmlee/Multi-Robot-TrajOpt.

References

[1] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[2] V. N. Hartmann and M. Toussaint, "Towards computing low-makespan solutions for multi-arm multi-task planning problems," *arXiv [cs.RO]*, May 2023.

[3] N. J. Nilsson, *Principles of artificial intelligence*. Springer Science & Business Media, 1982.

[4] Y. Shaoul, I. Mishani, M. Likhachev, and J. Li, "Accelerating search-based planning for multi-robot manipulation by leveraging online-generated experiences," in *Int. Conf. Autom. Plan. Sched. (ICAPS)*, vol. 34, 2024, pp. 523–531.

[5] J. Chen, J. Li, Y. Huang, C. Garrett, D. Sun, C. Fan, A. Hofmann, C. Mueller, S. Koenig, and B. C. Williams, "Cooperative task and motion planning for multi-arm assembly systems," *arXiv [cs.RO]*, Mar. 2022.

[6] K. Well and K. Ebert, "Trajectory optimization techniques and software implementation," *IFAC Proceedings Volumes*, vol. 25, no. 22, pp. 77–87, 1992, 12th IFAC Symposium on Automatic Control in Aerospace 1992, Ottobrun, Germany, 7-11 September.

[7] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 7674–7679.

[8] P. Huang, R. Liu, S. Aggarwal, C. Liu, and J. Li, "Apex-mr: Multi-robot asynchronous planning and execution for cooperative assembly," *arXiv preprint arXiv:2503.15836*, 2025.

[9] V. N. Hartmann, T. Heinle, and S. Coros, "A benchmark for optimal multi-modal multi-robot multi-goal path planning with given robot assignment," *arXiv preprint arXiv:2503.03509*, 2025.

[10] Y. Shaoul, I. Mishani, M. Likhachev, and J. Li, "Accelerating search-based planning for multi-robot manipulation by leveraging online-generated experiences," in *International Conference on Automated Planning and Scheduling*, 2024.

[11] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2. IEEE, 2000, pp. 995–1001.

[12] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (BIT*): Informed asymptotically optimal anytime search," *Int. J. Rob. Res.*, vol. 39, no. 5, pp. 543–567, Apr. 2020.

[13] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap composition for multi-arm systems path planning," in *IEEE/RSJ Int. Conf. Robot. Syst. (IROS)*, Oct. 2009, pp. 2471–2476.

[14] Z. Bien and J. Lee, "A minimum-time trajectory planning method for two robots," *IEEE Trans. Rob. Autom.*, vol. 8, no. 3, pp. 414–418, Jun. 1992.

[15] S. Zhang and F. Pecora, "Online and scalable motion coordination for multiple robot manipulators in shared workspaces," *IEEE Trans. Autom. Sci. Eng.*, vol. PP, no. 99, pp. 1–20, 2024.

[16] M. Erdmann and T. Lozano-Pérez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, pp. 477–521, Nov. 1987.

[17] K. Harada, T. Tsuji, and J.-P. Laumond, "A manipulation motion planner for dual-arm industrial manipulators," in *IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, May 2014, pp. 928–934.

[18] K. Gao and J. Yu, "Toward efficient task planning for dual-arm tabletop object rearrangement," *Rep. U. S.*, 2022.

[19] K. Gao, Z. Ye, D. Zhang, B. Huang, and J. Yu, "Toward holistic planning and control optimization for dual-arm rearrangement," *arXiv [cs.RO]*, Apr. 2024.

[20] R. Shome and K. E. Bekris, "Synchronized multi-arm rearrangement guided by mode graphs with capacity constraints," in *Int. Workshop Algorithmic Found. Robot. (WAFR)*, vol. 14. Springer, 2021, pp. 420–436.

[21] V. N. Hartmann, A. Orthey, D. Driess, O. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *arXiv [cs.RO]*, Jun. 2021.

[22] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin, "Fast, high-quality two-arm rearrangement in synchronous, monotone tabletop setups," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 888–901, Jul. 2021.

[23] T. Pan, A. M. Wells, R. Shome, and L. E. Kavraki, "A general task and motion planning framework for multiple manipulators," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Sep. 2021, pp. 3168–3174.

[24] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009.

[25] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research*, vol. 33, no. 9, 2014.

[26] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *International Journal of Robotics Research*, vol. 37, no. 11, 2018.

[27] S. S. Mirrazavi Salehian, N. Figueroa, and A. Billard, "Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty," in *Proceedings of Robotics: Science and Systems (RSS)*, 2016.

[28] J. Chen, "Multi-robot trajectory generation via consensus admm: Convex vs. non-convex," *arXiv preprint arXiv:2410.01728*, 2024.

[29] S. Zimmermann, M. Busenhart, S. Huber, R. Poranne, and S. Coros, "Differentiable collision avoidance using collision primitives," *arXiv preprint arXiv:2204.09352*, 2022.

[30] J. Liu, H. J. Yap, and A. S. M. Khairuddin, "Path planning for the robotic manipulator in dynamic environments based on a deep reinforcement learning method," *Journal of Intelligent & Robotic Systems*, vol. 111, no. 1-3, p. 3, 2025.

[31] F. E. Curtis, O. Schenk, and A. Wächter, "An interior-point algorithm for large-scale nonlinear optimization with inexact step computations," *SIAM Journal on Scientific Computing*, vol. 32, no. 6, pp. 3447–3475, 2010.

[32] P. E. Gill, W. Murray, and M. A. Saunders, "Snopt: An sqp algorithm for large-scale constrained optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, 2002.

[33] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *International Conference on Informatics in Control, Automation and Robotics*, 2004.

[34] K. Tracy, T. A. Howell, and Z. Manchester, "Differentiable collision detection for a set of convex primitives," 2023. [Online]. Available: https://arxiv.org/abs/2207.00669