2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

TOCALib: Optimal control library with interpolation for bimanual manipulation and obstacles avoidance

Yulia Danik[1], Dmitry Makarov[2], Aleksandra Arkhipova[3], Sergei Davidenko[4] and Aleksandr Panov[5]

***Abstract*— The paper presents a new approach for constructing a library of optimal trajectories for two robotic manipulators, Two-Arm Optimal Control and Avoidance Library (TOCALib)[1]. The optimization takes into account kinodynamic and other constraints within the FROST framework. The novelty of the method lies in the consideration of collisions using the DCOL method, which allows obtaining symbolic expressions for assessing the presence of collisions and using them in gradient-based optimization control methods. The proposed approach is applicable for complex bimanual manipulations that require precision. In this paper we tested TOCALib on Mobile Aloha robot, as an example. The approach can be extended to other bimanual robots, as well as to gait control of bipedal robots. It can also be used to construct training data for machine learning tasks for manipulation.**

I. INTRODUCTION

The field of robotics is rapidly evolving, with a growing emphasis on the development and deployment of bimanual robots, such as the TIAGo [1] and anthropomorphic robots like Baxter [2], Figure 01, Astribot S1 etc. These robots, designed with two arms to mimic human coordination, have opened up new possibilities to perform complex tasks that require simultaneous control of multiple manipulators. Bimanual tasks are becoming increasingly important as industries seek more sophisticated automation solutions that can handle tasks requiring precise coordination and flexibility, such as assembly, packaging, and collaborative work with humans. Collision avoidance in robot manipulators is essential for safe and efficient operation. To accomplish this task, it is necessary to plan the collision-free paths and control trajectories of the manipulators from the current positions to the desired positions that are comfortable for safe object grasping.

The recent emergence of relatively low-cost mobile bimanual robots has attracted considerable interest from researchers in this field. One of these robots is a Mobile Aloha, which was developed recently. It is a mobile platform with two independently controlled wheels equipped with four manipulators (two rare for teleoperation and two front for manipulation). Its manipulators feature 16 DOF in total, with 8 DOF for each arm [3].

Usually Aloha is controlled using behavior cloning based on the Action Chunking with Transformers approach [3]. However, it is still relevant to control Mobile Aloha using alternative methods capable of solving tasks without human intervention and finding optimal or unconventional solutions. These include "classical" optimal control methods and reinforcement learning (RL) approaches. Optimization methods (trajectory optimization, nonlinear programming, etc) are notoriously sensitive to initial guesses. A good warm start (e.g., using the previous timestep's solution, or a geometrically feasible trajectory from a sampling-based planner) can drastically reduce solve time and improve convergence. Poor initialization often leads to local minima, divergence, or infeasible trajectories — a well-known challenge in kinodynamic planning frameworks. For the implementation of offline RL training or for the practical application of certain online RL methods (e.g. SERL [4]), high-quality task solution examples are also required. Poor initialization of trajectories or policy parameters can lead to slow learning or collapse.

Thus, for both classical real-time optimal control manipulation and RL-based manipulation, it is desirable to have good initial approximations for the motion of the manipulators. To achieve this, we propose a precomputed motion library with interpolation mechanism. The library contains optimal trajectories and controls for the manipulators, taking into account self-collisions and collisions with static and dynamic objects. Each trajectory is parameterised by the initial and final state of the manipulator.

This paper proposes a method for constructing a precomputed Two-Arm Optimal Control and Avoidance Library (TOCALib) for manipulators, which provides the following advantages:

- Optimal trajectories and controls taking into account detailed kinodynamic model of the robot.
- Fast computation of optimal control thanks to the use of the symbolic representation of the problem.
- The ability to account for dynamic obstacles (robot arms, moving objects, etc.).
- Solution for arbitrary endpoints through the interpolation mechanism (trilinear interpolation and splines) and the use of library entries.
- Reduction of computation and search time for trajectories between library grid nodes.

Kinodynamic planning addresses the problem of generating robot motions that simultaneously satisfy kinematic

[1]Yulia Danik is with MIRAI, Moscow, Russia danik.y@miriai.org
[2]Dmitry Makarov is with MIRAI, Moscow, Russia makarov.d@miriai.org
[3] Aleksandra Arkhipova is with DGIST, Daegu, South Korea aleksandra_01@dgist.ac.kr
[4] Sergei Davidenko, SberRobotics, PhD CESS Moscow, Russia s.davidenko@applied-ai.ru
[5]Aleksandr Panov is with MIRAI and AXXX, Moscow, Russia panov.a@miriai.org
[1]https://sites.google.com/view/tocalib

constraints (e.g., obstacle avoidance, joint limits) and dynamic constraints (e.g., bounds on velocity, acceleration, forces). Unlike purely geometric planning, kinodynamic methods output trajectories that are both collision-free and dynamically feasible, making them particularly suitable for autonomous driving, UAV flight, robotic manipulation, and legged locomotion. The developed TOCALib software platform can be used to build training datasets for RL agents, as well as a baseline solution to evaluate their performance. We use the Mobile Aloha as an example of TOCALib application.

II. Related works

A. Trajectory Planning for Manipulators

Recent kinodynamic planning methods illustrate different trade-offs between physical realism and computational cost. Silico method [5] is a differentiable simulation framework that unifies collision detection and contact dynamics into a differentiable single-level optimization, though the method remains solver-heavy. KDF, Kinodynamic Motion Planning via Geometric Sampling-based Algorithms and Funnel Control [6], is a framework that integrates sampling-based planning with funnel-based feedback control. A geometric planner first generates a safe path in an extended free space, and a funnel controller then guarantees robust tracking under nonlinear and uncertain dynamics. Notably, neither module requires explicit system dynamics. Search-based Kinodynamic Motion Planning for Quadruped Robots [7] first generates smooth, time-optimal, dynamically feasible paths using kinodynamic search, with collision costs as soft constraints for safety. These paths are then refined with the Timed Elastic Band (TEB) method tailored to quadruped locomotion. MICP approach, kinodynamic motion planning framework for multi-legged robot jumping based on the mixed-integer convex program [8], combines configuration space discretization with the construction of a feasible wrench polytope (FWP), encoding kinematic constraints, actuator limits, friction cones, and gait sequencing into a single mixed-integer convex program. It provides optimal but computationally expensive solutions and scales poorly to complex manipulators.

Our TOCALib takes into account kinodynamic constraints together with differentiable collision detection and employs warm-start interpolation between stored solutions. This enables fast access to kinodynamically feasible motions for arbitrary endpoints, drastically reducing online computation and making the approach well-suited for complex bimanual tasks as well as for generating training data in learning-based methods.

B. Trajectory Planning for Manipulators

We proposed TOCALib, an original approach based on creating a precomputed motion library for manipulators using the FROST package in MATLAB, solving nonlinear programming (NLP) tasks for various end-effector positions with IPOPT and considering a set of symbolic constraints (dynamics, kinematics, collisions). For collision avoidance, the Differentiable Collisions library in Julia is separately integrated, allowing distance evaluation between primitives and obtaining the first derivative of this distance. TOCALib approach is positioned as a means of obtaining a high-quality dataset with manipulator trajectories. Table I presents a comparison of this approach with most popular alternative methods based on different criteria. To sum up, the proposed approach provides high computational efficiency and physical accuracy through symbolic optimization and consideration of the full dynamic model. These properties make it especially suitable for complex industrial tasks and the generation of high-quality data for reinforcement learning (RL).

C. Precomputed motion libraries

There are no existing works in the literature dedicated to creating motion libraries for manipulators; however, this idea has shown promising results for bipedal robots. For instance, in [3, 9, 10], a library of gaits with various longitudinal and lateral speeds was created for the Cassie robot and full-body humanoid robot [11] using the C-FROST framework [12]. If a specific solution was absent in the library, it was obtained using trilinear interpolation or spline interpolation. FROST and its extension, C-FROST, are frameworks that solve control problems for hybrid systems (i.e., systems with both continuous and discrete subsystems) using the hybrid zero dynamics method [13]. To accelerate optimization, gradients of kinodynamic and other constraints are computed symbolically. We used FROST for trajectory optimization.

D. Collision Detection

The most widely used collision detection algorithms based on Gilbert-Johnson-Keerthi (GJK)[14, 15] and Minkowski Portal Refinement (MPR)[16, 17] methods, which use support mapping to determine collisions between convex objects. Enhancements to MPR and zero-crossing handling functions are proposed in [18]. These methods are implemented in the Flexible Collision Library (FCL) [19] and physics engines such as Bullet [20], MuJoCo [21] etc. However, they are not differentiable due to logical control flow. Differentiable collision detection method based on randomized smoothing [22], that perturbs object configurations to approximate gradients. It has been integrated into HPP-FCL (Humanoid Path Planner – Flexible Collision Library) and Pinocchio, though the gradients remain approximate and potentially noisy.

Differentiable collision detection has also been explored through Signed Distance Function (SDF)-based methods and DiffCo. SDF methods [23] represent shapes as continuous distance fields, giving exact collision gradients but are costly for complex geometries. DiffCo (Autodifferentiable proxy collision detection)[24] learns neural implicit functions to represent collision boundaries directly in configuration space, enabling fast differentiable collision queries for manipulators in cluttered scenes, but at the cost of approximation accuracy.

In contrast, methods like Differentiable Collision (DCOL) [25] or Silico [5] use analytical convex geometry operations and integrate collision detection into a single optimization

TABLE I

COMPARISON OF APPROACHES TO CREATING A PRECOMPUTED MOTION LIBRARY FOR MANIPULATORS

Criterion	**TOCALib** (gitlab.com/yuliadanik/tocalib)	**MoveIt** (https://moveit.ai) / **Tesseract** [a]	**RoboDK** (https://robodk.com/)	**RMP Flow**[b]**/ TrajOpt**[c]**/ CHOMP**[d]	**OMPL** (https://ompl.kavrakilab.org/)	**Curobo** (https://github.com/nvlabs/curobo)
Approach Characteristics	Symbolic optimization; IPOPT solver for precise trajectories with dynamics and collision considerations; integration with Julia for distance gradient computation	Kinematic planning (OMPL: RRT, PRM); collision avoidance via TrajOpt and DART; tight integration with ROS	Offline programming for industrial tasks	Numerical optimization (SQP, Adam); FCL and Bullet for reactive collision avoidance; efficient in real-time	Heuristic path construction (PRM, RRT), user-defined constraint support	CUDA-accelerated robotics motion library with GPU parallel trajectory optimization for real-time manipulators optimization; Isaac ecosystem integration
Planning Methods	Nonlinear programming (NLP) with IPOPT and symbolic derivatives	Sampling-based planning with collision constraints (RRT, PRM)	Calculation of static trajectories without optimization	Numerical trajectory optimization (SQP, Adam)	Sampling-based planning (RRT, PRM)	Batched inverse kinematics, geometric planning, and trajectory optimization (minimizing jerk/acceleration) on GPU
Collision Algorithm	Differentiable Collisions (Julia), symbolic constraints	FCL (Flexible Collision Library)	Simple collision checking for offline tasks	Bullet / FCL (reactive collision avoidance)	FCL for basic collision checking	Signed-distance collision cost using sphere-based robot queries; supports primitive/cuboid and mesh worlds, plus voxel/ESDF worlds (nvblox)
Dynamic Consideration	Full (forces, torques, and acceleration constraints)	None (kinematics only)	None (only basic motion constraints)	Yes (acceleration and velocity consideration)	None, focused on static planning	Partial (velocity and acceleration limits; primarily kinematic with time parameterization and smoothness costs)
Constraint Handling	Symbolic constraints for collisions and dynamics	Collisions and motion boundaries	Range of motion constraints	Collisions, speeds, and accelerations	Basic collision prevention constraints	Collision constraints, joint limits, velocity/acceleration bounds (numerical formulation)
Trajectory Accuracy	High, symbolic derivatives and dynamic constraints	Medium, limited by kinematic parameters	High for industrial tasks with fixed trajectories	High, adaptive trajectories for dynamic conditions	Low, primitive heuristics	High in real-time applications due to dense optimization and parallel refinement
Adaptability to Environmental Changes	Limited, designed for precomputed trajectories	Limited, suitable for static tasks	Low, designed for predefined motions	High, adaptive real-time response	Low, focused on tasks without dynamics	High, supports rapid replanning (on GPU) and reactive collision avoidance in dynamic scenes
Symbolic Constraint Support	Yes, supported in Julia, useful for gradient optimization	No, numerical methods only	No, designed for static tasks	No, numerical methods and adaptive correction	No, heuristic constraints only	No, fully numerical GPU-based optimization
Use in RL	Suitable for generating high-quality RL training data	Moderate, used as initial dataset	Low, designed for fixed tasks	High, suitable for adaptive RL	Low, limited RL support	High, fast batched planning useful for large-scale RL data generation
Computational Resource Intensity	High, due to IPOPT and symbolic optimization, but justified for complex tasks	Moderate, optimized for ROS, suitable for research	Low, simple maintenance for static tasks	High, especially in real-time tasks	Low, focused on simple calculations	High GPU utilization; optimized for massively parallel computation

[a] https://github.com/tesseract-ocr/tesseract

[b] https://docs.omniverse.nvidia.com/isaacsim/latest/concepts/motion_generation/rmpflow.html

[c] https://github.com/tesseract-robotics/trajopt?ysclid=m3hqz7burk86695758

[d] https://personalrobotics.cs.washington.edu/publications/zucker2013chomp.pdf

problem, yielding analytic and exact gradients. This distinction makes differentiable formulations far more suitable for trajectory optimization, reinforcement learning, and other applications.

In this paper for collision detection we rely on the DCOL method, which is implemented in the Julia DifferentiableCollisions library [25]. It determines collisions between convex primitives, including polyhedra, capsules, cylinders, cones, ellipsoids, and padded convex polygons. These shapes can approximate the geometry of the robot and any surrounding objects. DifferentiableCollisions computes the scaling factor of convex bodies at which a collision occurs. If the scaling factor is greater than one, no collision is present. The library also computes the derivative of this factor with respect to the geometric parameters of convex bodies, such as position, orientation, and size, making it suitable for gradient-based optimization methods. Thus, it has greater computational efficiency than alternative methods.

III. PROBLEM STATEMENT

We consider manipulator movement with collision avoidance as a nonlinear programming problem (NLP). To describe the dynamics of the whole robot, the next second-order Euler-Lagrange equation is used.

$$M(q)\,\ddot{q} + C(q,\ \dot{q})\,\dot{q} + G(q) = Bu, \tag{1}$$

where $q \in \mathbb{R}^n$ are the generalized coordinates, $M(q)$ is the mass matrix, $C(q,\ \dot{q})\,\dot{q}$ are vectors containing the centrifugal and Coriolis forces, $G(q)$ are the gravitational forces, and $B \in \mathbb{R}^{n\times m}$ is the actuation matrix, $u \in U \subset \mathbb{R}^m$ is

the control (torques) actuating the system, U is the set of admissible control inputs. Let's rewrite (1) as

$$\dot{x} = f(x) + g(x)u$$

where $f(x) = \begin{bmatrix} \dot{q} \\ -M(q)^{-1}(C(q,\dot{q})\dot{q} + G(q)) \end{bmatrix}, g(x) = \begin{bmatrix} 0 \\ M(q)^{-1}B \end{bmatrix} u$, $x = \left[q^T \dot{q}^T\right]^T \in \mathbb{R}^{2n}$. The cost function:

$$\min_{u_i} \sum_{i=1}^{N} Cost(x_i, u_i, t_i), \tag{2}$$

N is the number of nodes in the optimization problem, $t \in [t_{min}, t_{max}]$. There are the following constraints:

- *C1*. The closed system dynamics equation, $\dot{x} = f(x) + g(x)u(x)$,
- *C2*. The physical feasibility condition and fixed-joints constraints (holonomic), including the torque limits, joints angles and velocities limits, etc.,
- *C3*. Collision constraints,
- *C4*. Target constraints (end-effector final point or path).

In our work we use the sum of squares of controls as a cost function, that is $Cost = u^T u$. To construct the motion library, it is necessary to solve the problems 2 for the given sets of initial and final states, corresponding to the initial and final positions of the robot manipulators.

IV. MOTION LIBRARY DESIGN

A. NLP Solver Framework

The FROST framework was selected for solving the NLP, as it allows for the rapid formulation and solution of control problems for robotic systems. Additionally, it enables solving control problems for hybrid systems (e.g., walking robots) in the future. FROST is implemented in MATLAB and uses the direct collocation method for numerically solving the nonlinear dynamics *C1*. It supports multiple solvers. In the current work, IPOPT was used [26]. To run IPOPT from FROST, all NLP constraints must be translated into C language files and compiled using the mex compiler embedded in MATLAB. The resulting solution is not globally optimal and includes elements of stochastic search.

B. Constraints

The constraints related to the robot's dynamics and kinematics (i.e., constraints *C1* and *C2*) are automatically generated in FROST from the URDF file in symbolic form. The constraints *C3* and *C4* are user-defined based on the specific requirements of the task. FROST does not have a convenient collision detection and avoidance mechanism that is why additional third-party packages need to be used.

C. Collision Detection Tool

To handle collisions, the DifferentiableCollisions (DCOL) library implemented in Julia was used. It also provides the first derivative of the collision parameters with respect to object position and orientation, making it suitable for integration into gradient-based control and learning methods.

For convex primitive shapes, collision constraints are generated in symbolic form using the DCOL, written in Julia. For DCOL, a wrapper is generated in C, which is compiled into a MEX file, accessible for calling from FROST (MATLAB, IPOPT).

D. Interpolation

Trilinear interpolation is employed as an alternative to solving complex optimal control problems. Optimal trajectories stored in the library are involved in the interpolation and help to determine an approximate solution to the problems not included in the library. For an arbitrary end-effector position within the studied range 8 trajectories that correspond to the nearest points of the grid are used to get the approximation. Trilinear interpolation is a method of multivariate interpolation on a 3-dimensional regular grid. It approximates the value of an intermediate point (x, y, z) within the local axial rectangular prism linearly, using data on the lattice points. An example of interpolation errors is shown in Fig. 1. We have also tested the cubic spline data interpolation to further improve the quality of the obtained results.

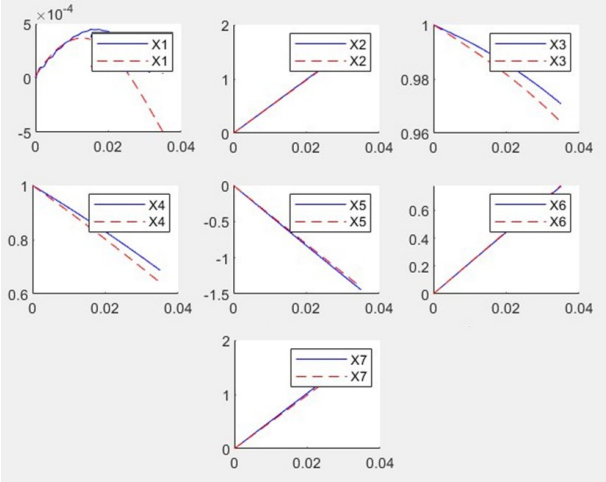


Fig. 1. Interpolation errors for all joints of the Aloha manipulator when moving the manipulator from a given initial point to a final point located 40 cm away. The red dashed curve represents the interpolation results, while the solid blue line shows the exact solution obtained through optimization.

V. EXPERIMENTS

Before performing the experiments, firstly, we calibrated several parameters of the physical robot and transferred them into the simulation model (Table II).

A total of 10 experiments were conducted on the Aloha (described below). Note that they used robot built-in controllers that embodied the trajectories calculated in TOCALib. We have experimentally verified that the error between the real and simulated trajectories is quite small. The average error of the joint angles is 0.0258 rad, the average error of the end-effector position in meters is 0.0170.

Experiments with torque control on a real robot require clarification of the mathematical model by identifying various parameters (rigidity and damping coefficients, friction,

TABLE II

ARM JOINT EFFORTS, VELOCITIES, AND BOUNDS

Name	Effort (Nm)	Velocity (rad/s)	Bounds (rad)
joint1	9	3.7699	[-2.11...3.1294]
joint2	9	3.7699	[0.044...3.65]
joint3	9	3.7699	[0.0364...3.229]
joint4	3	12.5664	[-1.37...1.36]
joint5	3	12.5664	[-1.534...1.541]
joint6	3	12.5664	[-2.1...2.087]
joint7	3	12.5664	[0.02...14.35]
joint8	3	12.5664	[0.02...14.35]

backlash, zones of insensitivity, etc.). The identification is a separate research and may be the subject of future work.

For the numerical experiments with TOCALib, the NLP problems (2) were solved, considering the full model of Mobile Aloha with 39 DOF, i.e., $q \in \mathbb{R}^{39}$. The cost function was defined as the sum of the squared controls, and the entire time interval was divided into 31 segments (i.e. $N = 31$).

A. Enhanced URDF Loading Mechanism

The original URDF model of ALOHA was initially written as a single, monolithic file. The authors introduced a parameterization of the URDF model fields by using XACRO files and implemented a modular system[2]. As a result, the model became more adaptable to changes in manipulator configurations, and code duplication was eliminated, significantly enhancing its flexibility and maintainability.

B. Self-collision Constraints

Each Aloha robot manipulator consists of 8 links, each represented by a separate capsule (highlighted in transparent blue), resulting in a total of 16 capsules for the two manipulators. The total number of 141 self-collision constrains were used, including the self-collisions of each manipulator, the collision of one manipulator with the other and, finally, the collision of manipulators with the base. A flexible mechanism was implemented for adding collision constraints between specific capsules. In the optimization process, a holonomic constraint $\alpha(q)_i > 1,\ i = 1, 2, ..., 141$ was applied, where α is the scaling parameter for the capsules. For gradient optimization, the Jacobian for each $\alpha(q)_i$ is computed as follows

$$\left[\frac{\partial \alpha_i}{\partial q}\right]_{1\times 39} = \left[\frac{\partial \alpha_i}{\partial \mathbf{params}}\right]_{1\times 14} \left[\frac{\partial \mathbf{params}}{\partial q}\right]_{14\times 39}, \quad (3)$$

where **params** is the parameter vector for two examined capsules which contains 3D position vectors and 4D quaternions defining orientation (a vector of length 7 for each capsule).

C. Collision with Static and Dynamic Objects

For experiments, we considered the approximation of collision objects using spheres, polytopes and cylinders. We can also automatically create random cluttered scenes from mesh files.

[2]https://sites.google.com/view/tocalib?usp=sharing

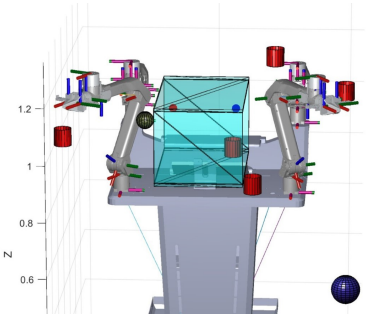
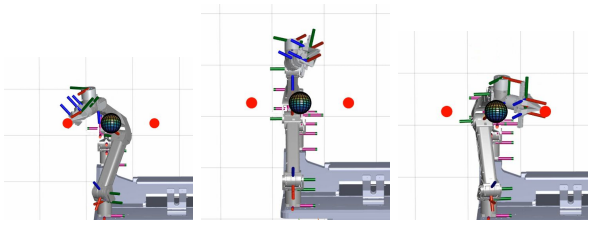


Fig. 2. A possible optimization task for TOCALib



(a) Initial position (b) Intermediate position (c) Final position

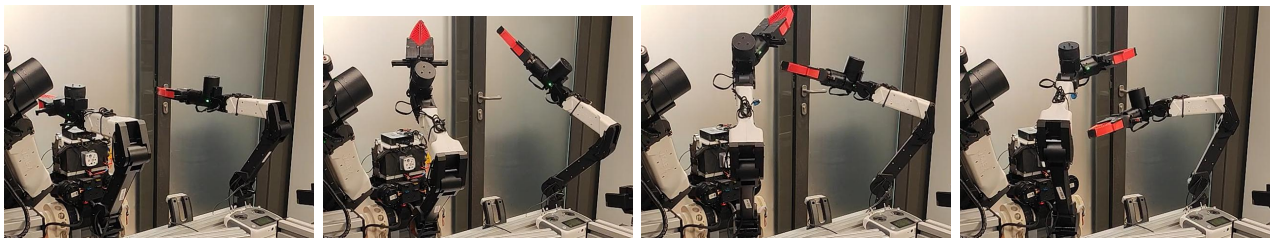
Fig. 3. Collision-free trajectories of a manipulator

Each sphere is described by a radius, a position and an orientation vector. A polytope is described by a set of halfspace constraints $Aw \leq b$, where $w \in \mathbb{R}^3$, $A \in \mathbb{R}^{m\times 3}$ and $b \in \mathbb{R}^m$ represent the m halfspace comprising the polytope (in our experiments we used four parallelepipeds described by $m = 6$ halfspaces). A polytope also has position and orientation parameters. Thus, one additional constraint was added for collision of robot links with a sphere object $\alpha(q)_i > 1,\ i = 1, 2, ..., 16$ (16 pairs, 8 for each manipulator) and another for collision with a shelf constructed from 4 polytopes (64 pairs for collision check).

D. Tasks

The numerical experiments involved moving manipulators along a number of target points with self-collision avoidance and collision-free moving in the presence of static and moving objects (Fig. 2) and also for random cluttered scenes from mesh objects. A sphere (Fig. 3) and shelves (Fig. 5-6) were used as static objects and a moving sphere (Fig. 7-8) as a dynamic object. The optimization tasks incorporated motion equations, self-collision constraints, and target end-effector positions. The provided numerical experiments showed that the proposed method provides trajectories without self-collision (Fig. 4). The corresponding RMSE (Root Mean Square Error) estimation error is significantly small. In Fig. 3 the scenario with a static sphere avoidance while following the way-points (red dots) is presented. We have also conducted the experiments with a moving sphere collision avoidance (Fig. 7). These trajectories have been successfully implemented on a real Aloha robot (Fig. 8).

The experiment with a more complicated environments showed that the proposed algorithm found a collision-free trajectory and the corresponding control. We compared TOCALib with CHOMP. In to TOCALib, CHOMP algorithm

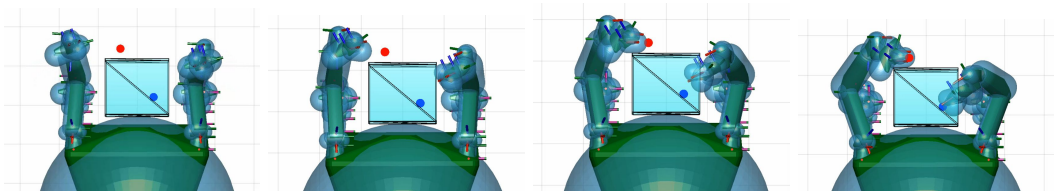


(a) Initial position (b) Intermediate position (c) Intermediate position (d) Final position

Fig. 4. Self collision avoidance

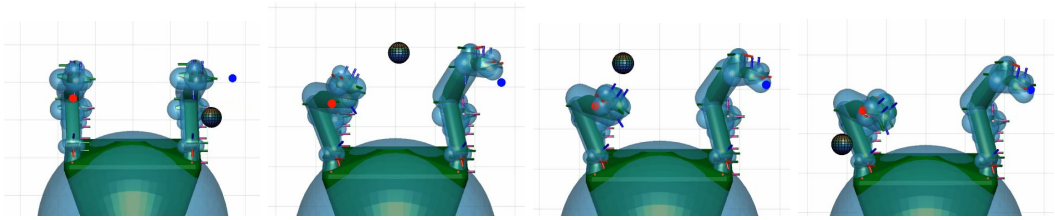
(a) Initial position (b) Intermediate position (c) Final position

Fig. 5. Collision avoidance in a presence of a shelf



(a) Initial position (b) Intermediate position (c) Intermediate position (d) Final position

Fig. 6. One trajectory from a library with shelf avoidance

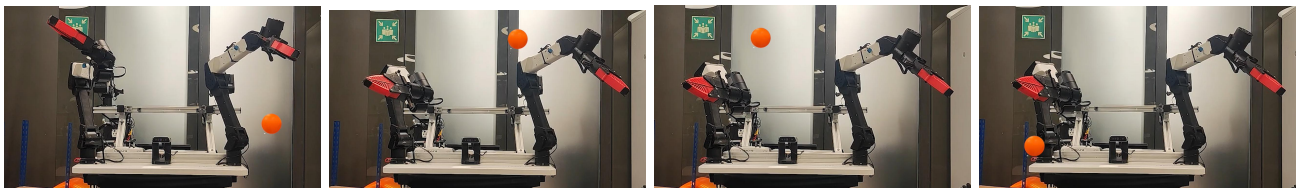


(a) Initial position (b) Intermediate position (c) Intermediate position (d) Final position

Fig. 7. Collision avoidance of a moving sphere in simulation

provides only the way-points. To calculate the trajectories, it approximates obstacles with spheres (Fig. 11) and finds a smooth collision-free trajectory. Though the direct comparison of TOCALib and CHOMP is not possible, as CHOMP does not account for the dynamics of the robot, the qualitative comparison of the obtained trajectories was made.

Two manipulator libraries were constructed using TOCALib. The first one contains 36 goal positions within a shelf ($x \in [0.5, 0.55, 0.6]$, $y \in [0.55, 0.1, 0.15]$, $z \in [0.95, 1.05, 1.15]$) (Fig. 9). One of the obtained trajectories was tested on a real Aloha robot (Fig. 5). Our method succeeded (Fig. 9) in 90% of cases (in 10% there was no solution because of the collision in the final state), while CHOMP managed to find the solution for 44% of cases, not including cases where it failed to give any result and the cases where the obtained solution was infeasible (false positive) because the integrity of the robot was compromised (Fig. 11). Moreover, we had to turn off self-collision checking in CHOMP because the MATLAB implementation offers only two presets for skipped self-collisions—"parent" or "adjacent"—and does not allow specifying custom link pairs, while Aloha robot requires omitting additional self-collision pairs. Therefore, when self-collision checking is enabled, CHOMP fails to produce a feasible solution. The second library ($x \in [0.15, 0.33, 0.43]$, $y \in [0, 0.08, 0.15]$, $z \in [1.03, 1.1, 1.26]$) was constructed for an environment



(a) Initial position (b) Intermediate position (c) Intermediate position (d) Final position

Fig. 8. Collision avoidance of a moving sphere for the robot

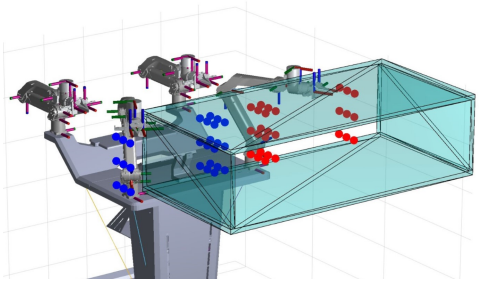


Fig. 9. The distribution of goal points in a library (red for the left arm, blue - for the right arm)

TABLE III

RESULTS OF THE EXPERIMENTS WITH LIBRARY

Method	Success	Success rate	Average time, sec
TOCALib	129 of 166	78%	483,55
CHOMP	85 of 166	51%	465,21

with a shelf placed between manipulators and 27 target positions (Fig. 10): some of them must be reached only by one manipulator, and others must be reached by both manipulators consistently. Thus, the library includes 325 combinations of final positions for both manipulators (cases where both manipulators have the same goal positions were not considered). From 325 trajectories calculated using TOCALib (Fig. 6) and CHOMP 159 were infeasible (a task for which no solution was obtained using TOCALib or CHOMP was considered unacceptable). The table III shows the result only for feasible cases. The CHOMP operation time was limited to 600 seconds, which corresponds to the maximum operating time limit of our method (see the section Simulation and runtime parameters). The algorithm for solving the problem using CHOMP is provided here[3]. TOCALib computation time can be decreased using the parallelization mechanizm implemented in CFROST, an open-source C++ interface for FROST. As can be seen from the table, our approach solves the problem in a significantly larger number of cases, while spending approximately the same time as CHOMP. Moreover, in 22% cases the video of CHOMP trajectories shows unexpected behavior (Fig. 11), thus it real success rate is likely even lower (we did not exclude such solutions from the list of successful ones for CHOMP).

E. Interpolation

We have also conducted experiments on interpolating between precomputed trajectories of the second library

[3]https://arxiv.org/html/2504.07708v1

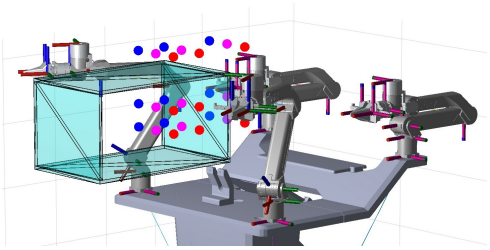


Fig. 10. The distribution of goal points in a library (red for the left arm, blue - for the right arm, and magenda - for both arms)

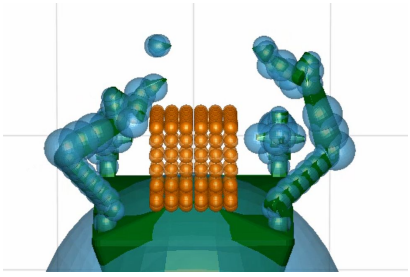


Fig. 11. The trajectory obtained by CHOMP with shelf collision avoidance (one connection of the arm is broken)

for new end-effector terminal positions defined by $x \in [0.386, 0.396]$, $y \in [0.101, 0.111]$, $z \in [1.111, 1.125]$. The average final state errors for the left and right manipulators are $0,0498$m and $0,0380$m, respectively. The RMSE is $0,163$ and $0,115$. The use of interpolated trajectory as an initial guess reduces optimization time: 1. ***Trilinear interpolation of trajectories*** provides a 10–12% runtime gain. 2. ***Trilinear interpolation of both trajectories and optimizer parameters*** yields 17–20% runtime improvement and higher accuracy. 3. ***Cubic spline interpolation*** achieves the highest accuracy (30–32%), although at increased computational complexity. For cubic spline it was necessary to compute additional trajectories so that the library grid contained not less than 4 values. The accuracy of interpolation strongly depends on grid resolution, the existence of a feasible solutions in all grid nodes. A key observation is that the complexity of the environment increases the number of required interpolation points to maintain accuracy. Trilinear interpolation performs best in the vicinity of grid nodes, whereas spline interpolation maintains higher accuracy across the entire interval.

F. Simulation and runtime parameters

During the simulation the end-effector target position error is set to 10^{-3} and the permissible deviation from the dynamic

equation in (1) is 10^{-4}. On average, trajectory computation takes approximately 2 minutes. The maximum computation time for one trajectory is set to 10 minutes. Computations were carried out on Intel Core i7 processor with 16 GB RAM.

VI. CONCLUSIONS

The proposed Two-Arm Optimal Control and Avoidance Library framework successfully combines precomputed motion strategies with collision avoidance techniques to address the challenges of bimanual manipulation. The integration of optimization in FROST with DifferentiableCollisions method for collision detection ensures the generation of high-quality motion trajectories in complex environments that are both accurate and safe. The collision constraints are symbolically represented in the optimization problem together with kinodynamic constraints and enable the integration of efficient collision avoidance into the trajectory planning process. TOCALib offers several advantages, such as support for a wide range of robots, fast optimization using gradient-based methods, and flexible collision-checking control. Approximate solutions can also be obtained without full optimization by using interpolation which gives adaptability. TOCALib works both for static and dynamic environments. In our comprehensive evaluation, we demonstrated the significant advantages of TOCALib over CHOMP for trajectory planning in bimanual manipulation tasks.

However, TOCALib has limitations. The computational time and limited number of allowed approximation primitives restrict TOCALib applicability in real-time scenarios that require rapid and accurate adaptation to environmental changes. It also needs convex decomposition of the scene and the robot. Nevertheless, our approach provides the interpolation tool, which helps to overcome this problem by utilizing precomputed optimal feasible trajectories with close goal positions. Moreover, our method provides a powerful solution for creating high-quality datasets for reinforcement learning and lays the foundation for generating diverse scenarios necessary for training RL agents.

The directions of future research include the application of the method for other types of robots, implementation of parallel calculations.

ACKNOWLEDGMENT

The study was supported by the Ministry of Economic Development of the Russian Federation (agreement No. 139-15-2025-013, dated June 20, 2025, IGK 000000C313925P4B0002)

REFERENCES

[1] Jordi Pages, Luca Marchionni, and Francesco Ferro. "Tiago: the modular robot that adapts to different research needs". In: *International Workshop on Robot Modularity, IROS*. Vol. 290. 2016.

[2] Tom Sorell. "Cobots, "collaboration" and the replacement of human skill". In: *Ethics and Information Technology* 24.44 (2022).

[3] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. *Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation*. 2024. arXiv: 2401.02117 [cs.RO].

[4] Jianlan Luo et al. *SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning*. 2024. arXiv: 2401.16013.

[5] Simon Le Cleac'h et al. "Single-level differentiable contact simulation". In: *IEEE Robotics and Automation Letters* 8.7 (2023), pp. 4012–4019.

[6] Christos K. Verginis, Dimos V. Dimarogonas, and Lydia E. Kavraki. *KDF: Kinodynamic Motion Planning via Geometric Sampling-based Algorithms and Funnel Control*. 2021. arXiv: 2104.11917 [cs.RO]. URL: https://arxiv.org/abs/2104.11917.

[7] Pei Wang et al. "Search-based kinodynamic motion planning for omnidirectional quadruped robots". In: *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2021, pp. 823–829.

[8] Yanran Ding, Chuanzheng Li, and Hae-Won Park. "Kinodynamic motion planning for multi-legged robot jumping via mixed-integer convex program". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 3998–4005.

[9] Jenna Reher and Aaron D. Ames. *Inverse Dynamics Control of Compliant Hybrid Zero Dynamic Walking*. 2020. arXiv: 2010.09047.

[10] Jenna Reher and Aaron D. Ames. *Control Lyapunov Functions for Compliant Hybrid Zero Dynamic Walking*. 2021. arXiv: 2107.04241.

[11] E. Chaikovskaya et al. "Benchmarking the Full-Order Model Optimization Based Imitation in the Humanoid Robot Reinforcement Learning Walk". In: *2023 21st International Conference on Advanced Robotics (ICAR)*. IEEE. 2023, pp. 206–211.

[12] Ayonga Hereid et al. *Rapid trajectory optimization using C-FROST with illustration on a Cassie-series dynamic walking biped*. 2021. arXiv: 2107.04241.

[13] J.W. Grizzle, G. Abba, and F. Plestan. "Asymptotically stable walking for biped robots: analysis via systems with impulse effects". In: *IEEE Transactions on Automatic Control* 46 (2001), pp. 51–64.

[14] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. "A fast procedure for computing the distance between complex objects in three-dimensional space". In: *IEEE Journal of Robotics and Automation* 4.2 (1988), pp. 193–203.

[15] S. Cameron. "Enhancing GJK: Computing minimum and penetration distances between convex polyhedra". In: *Proceedings of International Conference on Robotics and Automation*. 1997, pp. 3112–3117.

[16] Michael J. and Newth O. *Minkowski Portal Refinement and Speculative Contacts in Box2D*. 2013.

[17] X. Wang, J. Zhang, and W. Zhang. "The distance between convex sets with Minkowski sum structure: application to collision detection". In: *Computational Optimization and Applications* 77 (2020), pp. 465–490.

[18] A. Neumayr and M. Otter. "Collision handling with variable-step integrators". In: *Proceedings of the 8th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools*. 2017, pp. 9–18.

[19] J. Pan, S. Chitta, and D. Manocha. "FCL: A general purpose library for collision and proximity queries". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 3859–3866.

[20] E. Coumans. *Bullet Physics Simulation*. 2015.

[21] E. Todorov, T. Erez, and Y. Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033.

[22] Louis Montaut et al. *Differentiable Collision Detection: a Randomized Smoothing Approach*. 2022. arXiv: 2209.09012 [cs.RO]. URL: https://arxiv.org/abs/2209.09012.

[23] Eric Heiden et al. "Disect: A differentiable simulation engine for autonomous robotic cutting". In: *arXiv preprint arXiv:2105.12244* (2021).

[24] Yuheng Zhi, Nikhil Das, and Michael Yip. "Diffco: Autodifferentiable proxy collision detection with multiclass labels for safety-aware trajectory optimization". In: *IEEE Transactions on Robotics* 38.5 (2022), pp. 2668–2685.

[25] K. Tracy. *DifferentiableCollisions.jl*. [Online]. Available: https://github.com/kevin-tracy/DifferentiableCollisions.jl. 2024.

[26] A. Wächter and L. T. Biegler. "On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical Programming* 106.1 (2006). (preprint), pp. 25–57.