Hierarchical Motion Planning for Cooperative Mobile Manipulation: A Preliminary Report

Student Names and Numbers

I. INTRODUCTION

This project addresses the motion planning problem for cooperative mobile manipulation, where two mobile manipulators must transport a heavy, elongated payload (e.g., pipe or beam) in a warehouse-like environment with obstacles. The task requires coordinated motion while satisfying closed-chain constraints imposed by rigid grasping and ensuring collision-free trajectories.

We adopt a **hierarchical motion planning framework** combining global path planning with local Model Predictive Control (MPC). This approach decomposes the complex high-dimensional planning problem into manageable sub-problems: the global planner operates in object space to find feasible paths, while the local MPC controller optimizes robot configurations in real-time while tracking the reference trajectory.

Why this approach? Hierarchical frameworks have proven effective for multi-robot systems [1], [2], reducing computational complexity compared to fully centralized approaches. MPC as a local planner provides predictive capabilities, naturally handles constraints (joint limits, closed-chain, obstacles), and enables smooth coordinated motion for transporting heavy payloads.

To ensure robust development, we follow an **incremental complexity approach** explained in section V-A.

II. KINEMATIC MODEL

We use the **Albert** mobile manipulator, which consists of a Clearpath Boxer mobile base and a 7-DOF Franka Emika Panda arm. The choice may be revised based on URDF availability and suitability for the task.

A. Single Mobile Manipulator

1) Mobile Base Kinematics: The Clearpath Boxer is a differential-drive mobile base with configuration $\mathbf{q}_{b,i} = [x_i, y_i, \theta_i]^T \in \mathbb{R}^3$. The nonholonomic kinematic constraints are:

$$\dot{x}_i = v_i \cos(\theta_i), \quad \dot{y}_i = v_i \sin(\theta_i), \quad \dot{\theta}_i = \omega_i \tag{1}$$

where v_i is the linear velocity and ω_i is the angular velocity. To relate these velocities to the left and right wheel speeds, they can be computed as:

$$v_i = \frac{R}{2}\left(\dot{\phi}_{R,i} + \dot{\phi}_{L,i}\right), \qquad \omega_i = \frac{R}{2L}\left(\dot{\phi}_{R,i} - \dot{\phi}_{L,i}\right),$$

where R is the wheel radius, $2L$ is the track width, and $\dot{\phi}_{R,i}, \dot{\phi}_{L,i}$ are the angular velocities of the right and left wheels, respectively.

The contribution of the mobile base to the end-effector velocity is:

$$\mathbf{J}_{b,i}(\mathbf{q}_i) = \begin{bmatrix} \cos(\theta_i) & 0 & -d_y \sin(\theta_i) - d_x \cos(\theta_i) \\ \sin(\theta_i) & 0 & d_y \cos(\theta_i) - d_x \sin(\theta_i) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{6\times 3} \tag{2}$$

where $[d_x, d_y]^T$ represents the offset from the base center to the manipulator mounting point in the base frame.

2) Manipulator Kinematics: The Franka Panda arm has configuration $\mathbf{q}_{a,i} \in \mathbb{R}^7$. Its forward kinematics relative to the base frame is:

$$\mathbf{X}_{b,i}^{e,i} = f_{k,arm}(\mathbf{q}_{a,i}) \tag{3}$$

with Jacobian $\mathbf{J}_{a,i}(\mathbf{q}_{a,i}) \in \mathbb{R}^{6\times 7}$ relating joint velocities to end-effector twist in the base frame.

3) Combined System: The complete configuration space is $\mathcal{C}_i = \mathbb{R}^2 \times \mathbb{S}^1 \times (\mathbb{S}^1)^7$. The combined configuration is:

$$\mathbf{q}_i = [\mathbf{q}_{b,i}^T, \mathbf{q}_{a,i}^T]^T \in \mathbb{R}^{10} \tag{4}$$

The forward kinematics mapping to world frame is:

$$\mathbf{x}_{e,i}^w = \mathbf{X}_w^{b,i}(\mathbf{q}_{b,i}) \circ \mathbf{X}_{b,i}^{e,i}(\mathbf{q}_{a,i}) = f_k(\mathbf{q}_i) \tag{5}$$

where $\mathbf{x}_{e,i}^w \in SE(3)$ is the end-effector pose in world frame.

The differential kinematics is:

$$\dot{\mathbf{x}}_{e,i} = \mathbf{J}_i(\mathbf{q}_i)\dot{\mathbf{q}}_i \tag{6}$$

where the complete geometric Jacobian is:

$$\mathbf{J}_i(\mathbf{q}_i) = [\mathbf{J}_{b,i}(\mathbf{q}_i) \quad \mathbf{J}_{a,i}(\mathbf{q}_{a,i})] \in \mathbb{R}^{6\times 10} \tag{7}$$

Note that due to the nonholonomic constraint of the differential-drive base, the instantaneous motion is restricted, but the system remains controllable and can reach any configuration through appropriate maneuvers. The system has 10 DOF configuration space but only 9 instantaneous velocity DOF (2 base velocities + 7 arm velocities), while still being redundant for the 6-DOF end-effector task.

B. Dual Mobile Manipulator System

The complete system configuration is:

$$\mathbf{c} = [\mathbf{q}_1^T, \mathbf{q}_2^T, \mathbf{t}_{obj}^T]^T \in \mathbb{R}^{26} \tag{8}$$

where $\mathbf{t}_{obj} = [\mathbf{p}_{obj}^T, \boldsymbol{\alpha}_{obj}^T]^T \in \mathbb{R}^6$ is the minimal representation of the object pose.

1) Closed-Chain Constraint: When both robots rigidly grasp the object at fixed grasp points, the position-level constraint is:

$$f_{C^3}(\mathbf{c}) = \mathbf{E} - \mathbf{G} = \mathbf{0}_{12\times 1} \tag{9}$$

where:

$$\mathbf{E} = \begin{bmatrix} \mathbf{t}_{e,1}^{w} \\ \mathbf{t}_{e,2}^{w} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{t}_{g,1}^{w} \\ \mathbf{t}_{g,2}^{w} \end{bmatrix} \tag{10}$$

The grasping poses are computed via:

$$\mathbf{t}_{g,i}^{w} = g(\mathbf{t}_{obj}^{w}, \mathbf{t}_{g,i}^{obj}), \quad i = 1, 2 \tag{11}$$

where $\mathbf{t}_{g,i}^{obj}$ are known constant grasp poses relative to the object frame.

At velocity level, the closed-chain constraint requires:

$$\mathbf{J}_1(\mathbf{q}_1)\dot{\mathbf{q}}_1 = \mathbf{J}_{obj,1}\dot{\mathbf{t}}_{obj} \tag{12}$$

$$\mathbf{J}_2(\mathbf{q}_2)\dot{\mathbf{q}}_2 = \mathbf{J}_{obj,2}\dot{\mathbf{t}}_{obj} \tag{13}$$

where $\mathbf{J}_{obj,i} \in \mathbb{R}^{6\times 6}$ is the grasp mapping matrix:

$$\mathbf{J}_{obj,i} = \begin{bmatrix} \mathbf{I}_{3\times 3} & -[\mathbf{r}_i]_\times \\ \mathbf{0}_{3\times 3} & \mathbf{I}_{3\times 3} \end{bmatrix} \tag{14}$$

with $[\mathbf{r}_i]_\times$ being the skew-symmetric matrix of vector $\mathbf{r}_i$ from object center to grasp point i.

The nonholonomic constraints of both differential-drive bases add complexity to the motion planning problem, as not all object velocities $\dot{\mathbf{t}}_{obj}$ can be instantaneously realized. This motivates the hierarchical approach where the global planner finds feasible paths considering these constraints, and the local MPC controller handles the real-time trajectory tracking while respecting the differential-drive kinematics.

III. CHOSEN PLANNER

A. Global Planner: Custom RRT in Object Space

We implement a custom RRT-based planner [3] operating in object space rather than using off-the-shelf solutions. This choice is motivated by:

Why RRT? RRT efficiently handles high-dimensional configuration spaces, which is crucial when dealing with the complex constraints of cooperative mobile manipulation. The probabilistic completeness and ability to explore non-convex spaces make it well-suited for warehouse environments with multiple obstacles.

The planner operates in 6D object space ($SE(3)$ minimal representation) and performs the following validity checks for each sampled pose $\mathbf{x}_{obj}$:

- Object-obstacle collision detection
- Inverse kinematics feasibility for both robots
- Basic workspace reachability verification
- (Later phases) Redundancy metric threshold checking

B. Local Planner: Model Predictive Control

MPC optimizes robot joint trajectories over a prediction horizon N while tracking the global path.

Why MPC? Compared to single-step QP controllers, MPC:

- Anticipates future constraints and reference trajectory
- Produces smoother, more coordinated motion
- Naturally handles multiple objectives (tracking, smoothness, effort)
- Can potentially handle dynamic obstacles through predictive avoidance
- Provides clear comparison metric (horizon length analysis)

IV. PLANNED SIMULATION ENVIRONMENT

A. Simulator

PyBullet with custom OpenAI Gym environment. PyBullet provides:

- Realistic rigid body dynamics
- Efficient collision detection
- Robot URDF loading capability
- Reasonable computational requirements
- Python integration for rapid prototyping

Initial consideration of NVIDIA Isaac Sim was abandoned due to high computational requirements not all team members can satisfy.

B. Environment Setup

Warehouse-like scenario with:

- Task: grasp a 3-meter pipe, lift it from a pallet, and transport it to its storage location on a shelf.
- Static obstacles: storage racks, walls, boxes (modeled as convex shapes for computational efficiency in collision checking)
- Payload: rigid heavy elongated object (pipe/beam, 3-5m length)

If time permits, dynamic obstacles (e.g., moving humans or forklifts) will be added.

V. PLANNED SCENARIOS AND METRICS

A. Evaluation Scenarios

Following incremental complexity:

1) **Phase 1**: Single robot, point-to-point motion at different heights (validate basic motion using both the mobile base and arm).
2) **Phase 2**: Single robot holding an object, object-space planning, narrow passages (test hierarchical approach)
3) **Phase 3**: Two robots, cooperative transport, moderate clutter (validate closed-chain handling)
4) **Phase 4**: Two robots, dense obstacle field, long payload (stress test complete system)

B. Performance Metrics

- **Success rate**: Percentage of completed tasks without collisions or constraint violations
- **Planning time**: Global planner computation time (RRT convergence)
- **Execution time**: Total task completion time
- **Tracking error**: $\|\mathbf{x}_{actual} - \mathbf{x}_{desired}\|$ for object pose
- **Constraint violations**: Joint limits, collisions, closed-chain errors

C. Baseline Comparisons

- Single-step QP controller vs. MPC with varying horizons
- Joint-space planning vs. object-space hierarchical planning
- Custom RRT vs. standard RRT-Connect (if time allows)
- Static vs. dynamic obstacle scenarios

VI. OPEN QUESTIONS

We seek feedback on the following aspects:

1) Is the incremental development approach (single robot $\rightarrow$ object-space hierarchical $\rightarrow$ cooperative dual-robot) appropriate for demonstrating the framework's capabilities within the project timeline?
2) Are the chosen metrics sufficient for evaluating the planner's performance, or should we include additional measures (e.g., energy consumption, load distribution between robots)?
3) For the MPC implementation: should we prioritize linear MPC with linearized dynamics (faster, simpler) or nonlinear MPC (more accurate) given the project scope?
4) Is modeling obstacles as convex shapes acceptable for computational efficiency, or should we implement more complex collision geometries from the start?
5) For the custom RRT implementation: are there specific features or modifications (e.g., informed sampling, kinodynamic constraints) that would be valuable to demonstrate?
6) Should dynamic obstacle handling be considered a core feature or an optional extension?

REFERENCES

[1] L. Han et al., "A Hierarchical Motion Planning Framework for Reactive Cooperative Operation of Dual-Mobile Manipulators," *LNNS*, vol. 1376, pp. 829-840, 2025.

[2] H. Zhang et al., "A Novel Semi-Coupled Hierarchical Motion Planning Framework for Cooperative Transportation of Multiple Mobile Manipulators," *arXiv:2208.08054*, 2025.

[3] S. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," Research Report 9811, 1998.