Technical Details

In this section, we will elaborate on some of the technical details that is only briefly outlined in the paper.

Details for Motion Planning in Second Stage

For the second part of motion planning, we have innovatively proposed a reactive control-based planning approach. In this section, we will elaborate on the specific details of this approach.

The first-order differential kinematics for a serial-link manipulator can be described as

$$\mathbf{v}(t) = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}(t), \tag{1}$$

where $\mathbf{v}(t) = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z) \in \mathbb{R}^6$ denotes the spatial velocity of the end-effector. $\mathbf{J}(\mathbf{q})$ denotes the Jacobian matrix at state $\mathbf{q} \in \mathbb{R}^9$. For the mobile manipulator used in this work, it's degree of freedom is 9.

With the given task $\mathcal{T} = [O, \mathbf{p}_{\text{init}}, \mathbf{p}_{\text{end}}]$, we plan a collision-free trajectory of the object from $\mathbf{p}_{\text{init}}$ and $\mathbf{p}_{\text{end}}$ which can be denoted as $\mathbf{p}(t)$. We assume that the grasp points during SCM is fixed with the object, and T^i_{obj} is used the define the relative transform matrix between the object frame and grasp frame of $\mathcal{R}^i$. The trajectory of the end-effector for $\mathcal{R}^i$ can be denoted as

$$\mathbf{p}^i(t) = T^i_{\text{obj}}\mathbf{p}(t), \tag{2}$$

where $\mathbf{p}^i(t)$ is the trajectory for the end-effector of $\mathcal{R}^i$.

Then, we consider generate a trajectory in configuration space for $\mathcal{R}^i$ that follows $\mathbf{p}^i(t)$ and perform collision avoidance. Firstly, we consider an obstacle point in 3D space $\mathbf{p}_o \in \mathbb{R}^3$. For a point $\mathbf{p}^i_r$ on the robot $\mathcal{R}^i$, we calculate the distance between this two points

$$d_{ro} = ||\mathbf{p}_o - \mathbf{p}^i_r||. \tag{3}$$

A unit vector $\mathbf{n}_{ro}$ from $\mathbf{p}^i_r$ to $\mathbf{p}_o$ is

$$\mathbf{n}_{ro} = \frac{\mathbf{p}_o - \mathbf{p}^i_r}{d_{ro}} = -\mathbf{n}_{or}. \tag{4}$$

The time derivative for d_{ro} is

$$\dot{d}_{ro} = \mathbf{n}^T_{ro}(\dot{\mathbf{p}}_o - \dot{\mathbf{p}}^i_r). \tag{5}$$

For any point on $\mathcal{R}^i$, like $\mathbf{p}^i_r$, we can obtain the time derivative using

$$\dot{\mathbf{p}}^i_r = \mathbf{J}^i_{\mathbf{p}^i_r}(\mathbf{q}^i)\dot{\mathbf{q}}^i(t), \tag{6}$$

where $\mathbf{J}_{\mathbf{p}^i_r}(\mathbf{q}^i) \in \mathbb{R}^{3\times 9}$ can be seen as the Jacobian matrix for point $\mathbf{p}^i_r$.

By replacing Eq.6 into Eq. 5, we can get

$$\dot{d}_{ro} = \mathbf{n}^T_{ro}(\dot{\mathbf{p}}_o - \mathbf{J}^i_{\mathbf{p}^i_r}(\mathbf{q}^i)\dot{\mathbf{q}}^i(t)). \tag{7}$$

The velocity damper prevents robot to collide with other objects by restricting the velocity based on its distance from obstacles. A basic form of velocity damper can be formed as

$$v \leq \zeta\frac{d - d_s}{d_i - d_s}, \tag{8}$$

where v is the change rate of distance between obstacles, d is the distance between the robot and obstacles. ζ, d_s, and d_i are three hyper-parameters. ζ is a positive parameter that decide the intensity of damper. d_i defines the influence range of damper, and d_s is the stop distance of this damper.

By applying velocity damper into 6, we can get

$$\mathbf{n}^T_{ro}(\dot{\mathbf{p}}_o - \mathbf{J}^i_{\mathbf{p}^i_r}(\mathbf{q}^i)\dot{\mathbf{q}}^i(t)) \leq \zeta\frac{d_{ro} - d_s}{d_i - d_s}. \tag{9}$$

Then, we can get

$$-\mathbf{n}^T_{ro}\mathbf{J}^i_{\mathbf{p}^i_r}(\mathbf{q}^i)\dot{\mathbf{q}}^i(t) \leq \zeta\frac{d_{ro} - d_s}{d_i - d_s} - \mathbf{n}^T_{ro}\dot{\mathbf{p}}_o. \tag{10}$$

Eq. 10 indicates that we get a constraint in the configuration space for $\mathcal{R}^i$. Once the generated trajectories for all robots' configuration satisfy these constrains, we can guarantee these trajectories are collision free.

We consider a simplified model of the manipulator. Multiple balls are used to represent the structure of it, which can be found in Fig. 2. For any ball b^i_m on $\mathcal{R}^i$, we can find a point in the environment or on other robots with minimal distance $d_{b^i_m o}$. If this point is a point in the environment, $d_{b^i_m o}$ is defined as

$$d_{b^i_m o} = d_{c^i_m o} - r^i_m, \tag{11}$$

where c^i_m is the center of b^i_m, r^i_m is the radius of b^i_m. If this point is a point on other robots, like b^j_n, $d_{b^i_m o}$ is defined as

$$d_{b^i_m o} = d_{c^i_m c^j_n} - r^i_m - r^j_n. \tag{12}$$

For the first scenario, Eq. 10 provides a constraint for $\mathcal{R}^i$. For the second scenario, Eq. 10 becomes

$$-\mathbf{n}^T_{b^i_m o}\mathbf{J}^i_{b^i_m}(\mathbf{q})\dot{\mathbf{q}}^i(t) + \mathbf{n}^T_{b^i_m o}\dot{\mathbf{p}}_o \leq \zeta\frac{d_{b^i_m o} - d_s}{d_i - d_s}, \tag{13}$$

where

$$\dot{\mathbf{p}}_o = \dot{b}^j_n = \mathbf{J}^j_{b^j_n}(\mathbf{q}^j)\dot{\mathbf{q}}^j(t). \tag{14}$$

By replacing Eq. 14 into Eq. 14, we can get

$$-\mathbf{n}^T_{b^i_m o}\mathbf{J}^i_{b^i_m}(\mathbf{q})\dot{\mathbf{q}}^i(t) + \mathbf{n}^T_{b^i_m o}\mathbf{J}^j_{b^j_n}(\mathbf{q}^j)\dot{\mathbf{q}}^j(t) \leq \zeta\frac{d_{b^i_m o} - d_s}{d_i - d_s}. \tag{15}$$

Eq. 15 forms a constraint for two robots' configuration space that performs collision avoidance.

Considering each robot should follow the trajectories of their end-effectors, we can obtain

$$\mathbf{v}^i(t) = \mathbf{J}^i(\mathbf{q})\dot{\mathbf{q}}^i(t), \tag{16}$$

where $\mathbf{v}^i(t)$ is the velocity of the end-effector of $\mathcal{R}^i$. This can be seen as an equality constraint in configuration space to guarantee the relative pose between the robot and the object.

We define the poses for all robots in configuration space can be denoted as

$$\mathbf{q} = [\mathbf{q}^{1^T}, \mathbf{q}^{2^T}, \cdots, \mathbf{q}^{N^T}]^T. \tag{17}$$

By checking the nearest point for each balls of each robot, we can use Eq. 10 and Eq. 15 to get all constraints for collision avoidance, which can be simplified as

$$\mathbf{G}\dot{\mathbf{q}} \leq \mathbf{h}. \tag{18}$$

Eq.18 form a constraint for all robots in configuration space to avoid collision.

Similarly, we can combine constraints of end-effectors in Eq. 16 to get

$$\mathbf{A}\dot{\mathbf{q}} = \mathbf{b}. \tag{19}$$

We form this trajectory planning problem as an optimization problem

$$\begin{aligned} &\min\ \dot{\mathbf{q}}^T \dot{\mathbf{q}} \\ &\text{s.t.} \begin{cases} \mathbf{G}\dot{\mathbf{q}} \leq \mathbf{h} \\ \mathbf{A}\dot{\mathbf{q}} = \mathbf{b}. \end{cases} \end{aligned} \tag{20}$$

Eq. 20 is formulated as a Quadratic Programming (QP) problem, which we solve using the cvxopt solver. First, a discrete motion trajectory of the object is obtained, from which the corresponding end-effector trajectories for the robots can be derived. Then, using this equation, we can plan each robot's trajectory in the configuration space.

Prompt Design for Grasp Points Selection

We provide an example of prompt design for the grasp point selection task. Our prompt is mainly composed of five parts. The first part introduces the robot configuration used, such as each robot being equipped with a 6-DOF robot arm and an AGV. The second part describes the grasping task to be executed, specifying the number of grasp points to be generated and providing an example of the output prompt. In this section, we also include a requirement for CoT. The third part introduces the metrics for analyzing the stability of collaborative grasp poses. In the fourth part, we provide an example of the prompt. Finally, the fifth part contains the specific instructions for the current task.

Besides, we have also included the full content of all the prompts in the open-source code.

```
 You are leading a team of intelligent robots that can assign
$(num_robot) robots to do multiple manipulation tasks in a daily
environment. Each robot in your team has THREE parts: an agv
base with 4 Mecanum wheels, a 6-dof robot arm, and a parallel
jaw gripper. Now, this team is working in the environment. For
example, they need to move a table from one room to another.
They can do multiple tasks. Currently, they are working on
$(task_description).

 Your task is to find suitable GRASP points for these robots. You
will be provided with $(num_grasp) candidate grasp points labeled
from 0-$(num_grasp-1) in this image. You should choose $(num_robot)
grasp points in these $(num_grasp) points. Think step by step.
You should begin your thinking and analysis procedure using two
steps. 1. Analysing what kind of object is in the image, begin
with ⟨I FIND⟩. In this part, you should first find different parts
of the object. Then, analyze each point in what part. Describe the
positions of each point as detailed as possible! 2. Thinking about
what kind of poses are suitable for this grasping task, begin with
⟨I THINK⟩.

 Then, output the grasp points you choose. You should consider the
following aspects when you decide where your team should grasp. 1.
The center of mass of an object. During the selection of grasping
points, it is necessary to ensure that the object's center of mass
falls within the area formed by the grasping points. For example,
when two robots perform a grasp, the center of mass should ideally
lie on the line connecting the grasping points. 2. The distance
between each robot when performing grasping. To avoid potential
collision when grasping, these robots should not be close to each
other. For example, two robots can not grasp the same edge when
moving a rectangle. 3. The direction for the grasping. Since the
robot's gripper is a parallel two-finger gripper, you tend to
grasp horizontal surfaces rather than vertical surfaces. This
is because vertical surfaces are more prone to slipping between
the fingers due to the influence of gravity. 4. The stability
of the grasping. A flat surface is better than a curved surface.
A rough surface is better than a smooth surface. When grasping
a rectangle, grasping the center of each edge is better than
grasping the edge of each side. Because, when grasping edges,
the gripper is more prone to slipping off the object. The most
important criterion is 1, followed by 2, then 3, and finally 4.
...

 ⟨EXAMPLE⟩... ⟨/EXAMPLE⟩
 ⟨TASK⟩... ⟨/TASK⟩
```

Experiment Results

Case Study

We also demonstrate a task where two robots collaboratively organize a table and chairs. As shown in Fig. 1, two robots positioned arbitrarily in the environment can autonomously detect the location of chairs and then move them next to the table through collaborative transportation. This example illustrates that our proposed method generalizes well to different numbers of robots.

Discussion

In this work, we propose the first generalist framework, CollaBot, for simultaneous collaborative manipulation. We achieve a success rate of 52% among all the selected tasks. However, there are some limitations for CollaBot. Firstly, the success evaluation is mainly conducted with a limited type of robots. A dataset with more objects can be created to validate a more accurate success rate of the proposed method. Secondly, we propose a hierarchical framework to address the SCM problem, in which the failure of a single module may lead to the breakdown of the overall functionality. In future work, feedback mechanisms can be introduced to enable recovery from such failures. Additionally, CollaBot can be used as an expert policy for data collection, paving the way for end-to-end simultaneous collaborative manipulation.

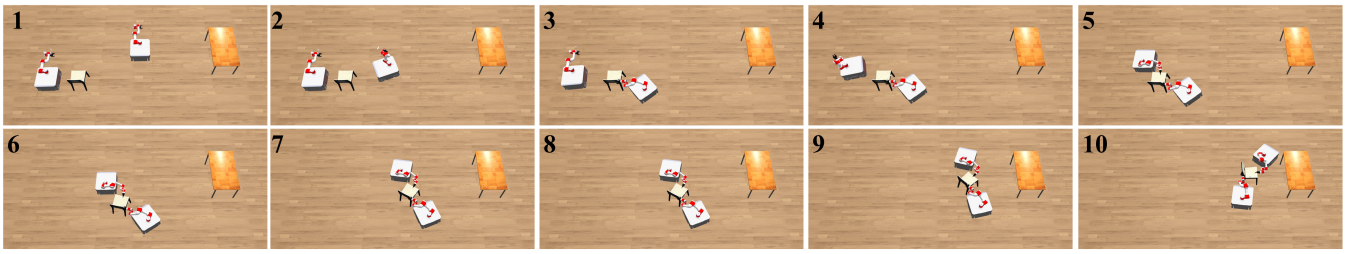

Figure 1: An example of using 2 robots to arrange a chair and a table. 1-5: Collaborative grasping is achieved. 6-10: SCM is performed.