Final report for Nonprehensile Object Transportation on a Mobile Manipulator

Binling He*, Jiawei You†, Xinlong Li‡

Jaunary 20, 2023

Abstract

The goal of this project is to control a manipulator with an mobile base to transport a cup of water, which is a nonprehensile object transportation task, through Model Predictive Control (MPC) and reinforcement learning. Nonprehensile object transportation is a challenging task. In this project, we want to train the manipulator to transport a cup of water using nonprehensile manipulation. We use a MPC method for nonprehensile object transportation with a mobile manipulator, with a focus on transporting a cup of water without spilling it. Also, we used a goal-conditioned reinforcement learning (GC-RL) based method to train the robot arm to learn the kinematic. Our method optimizes the joint-space trajectory online based on task-space objectives and constraints, and uses a fluid dynamics model to prevent water sloshing. We show that our method can handle various scenarios, such as moving to a desired location. Compared to traditional RL method, our RL method can converge faster. Finally, we test our method in simulations and hardware experiments.

1 Introduction

Nonprehensile Object Transportation is a non-grasping manipulation task, where a robot balances an object on its end-effector (EE) without grasping it, and transports it to a desired location while avoiding obstacles. This is a challenging task, because it requires considering the dynamics, friction, gravity, collision, and other factors of the object, as well as the motion constraints and control policies of the robot. Nonprehensile Object Transportation is a hot research topic in robotics, with many papers and projects related to it.

[1] proposed a trajectory-planning based method for nonprehensile manipulation. [2] proposes a model predictive non-sliding manipulation (MPNSM) control approach to safely transport an object on a tray-like end-effector of a robotic manipulator. The MPNSM controller uses a linear complementarity problem (LCP) formulation to enforce the non-sliding constraint and a mixed-integer quadratic program (MIQP) formulation to handle the robotic system constraints. [3] tackles the problem of nonprehensile object transportation through a legged manipulator. It solves the quadratic optimization problem to realize the sought transportation task while maintaining the contact forces between the tray and the object and between the legs and the ground within their respective friction cones, also considering limits on the input torques. Bernheisel and Lynch [4] introduced a variant of the Waiter's Problem that involves the stable transportation of object assemblies while accounting for inertial and gravity forces. Among this, trajectory planning plays a crucial role and usually necessitates prior knowledge of the robot's starting and ending locations, as well as their respective postures. Moreover, specific techniques for solving inverse kinematics (IK) and planning are also required In [1], a spherical pendulum model is employed to counteract the sloshing effect caused by a serial robot's movement. Some research focus on the model of pouring liquid tasks, for instance, in [5] the accurate pouring task has been executed using a pourer container in the shape of a cube and a target container in the shape of a cylinder. [6]

*Student ID: 2022231012, Email: hebl2022@shanghaitech.edu.cn
†Student ID: 2022233291, Email: youjw2022@shanghaitech.edu.cn
‡Student ID: 2022231019, Email: lixl12022@shanghaitech.edu.cn

addressed the need for efficient and stable object transportation using robotic arms. The authors propose the GOMP-FIT algorithm, which focuses on optimizing the grasp and motion planning process while considering the constraints and dynamics of the system. The paper emphasizes the importance of incorporating physics-based interaction models in motion planning to ensure better stability during transportation. The GOMP-FIT algorithm considers forces applied during grasping and balancing, as well as the weight and dynamics of the object. By doing so, it aims to improve the efficiency and stability of object transportation. Bernheisel and Lynch [4] introduced a variant of the Waiter's Problem that involves the stable transportation of object assemblies while accounting for inertial and gravity forces. Among this, trajectory planning plays a crucial role and usually necessitates prior knowledge of the robot's starting and ending locations, as well as their respective postures. Moreover, specific techniques for solving inverse kinematics (IK) and planning are also required In [1], a spherical pendulum model is employed to counteract the sloshing effect caused by a serial robot's movement. Some research focus on the model of pouring liquid tasks, for instance, in [5] the accurate pouring task has been executed using a pourer container in the shape of a cube and a target container in the shape of a cylinder.

Reinforcement learning (RL) is a subfield of machine learning that deals with how an agent interacts with an environment to learn how to make optimal decisions. RL has been applied in various domains, including robotics, where it has shown a lot of promise. In the field of robotics, RL algorithms can be used to train robots to perform complex tasks that are difficult to program using traditional methods. The use of RL in robotics is particularly relevant when the robot operates in environments that are unpredictable or dynamic, making it difficult to pre-program all possible scenarios. In such cases, RL algorithms can enable the robot to learn from its own experience and adapt to new situations, making it more versatile and efficient. [7] proposed a RL method applied to robot manipulators with a workspace invaded by unpredictable obstacles. [8] figured out that the prior experience is very import for the robot to learning a reinforcement learning algorithm.

In this project, we propose a method for solving the waiter's problem with a mobile manipulator, with the additional requirement of transporting a cup of water without spilling it. Unlike previous approaches that use sensor feedback or offline planning, our method uses a model predictive controller (MPC) that optimizes the joint-space trajectory online based on task-space objectives and constraints. We also incorporate a fluid dynamics model to account for the water motion inside the cup and prevent sloshing. Meanwhile, a reinforcement learning (RL) based method is used to learn the kinematic of the robot arm. We demonstrate the effectiveness of our method in simulations and hardware experiments, where the mobile manipulator can transport a cup of water quickly and smoothly, without spilling or dropping it.

2 System Description

2.1 Hardware platform

In this project, we utilized the Kinova Jaco2 and Clearpath Jackal as our hardware platforms. Additionally, we employed the OptiTrack motion capture system for localizing the positions of the robot arm's end-effector, the object to be transported, and any obstacles. The Kinova Jaco2 robot is a versatile robotic arm designed for various applications in research, industry, and assistive technology. Developed by Kinova Robotics, the Jaco2 robot offers advanced capabilities and precision control. The Jaco2 robot features 6 degrees of freedom (DOF), allowing it to move in a highly articulated manner. It is equipped with precise and responsive actuators that enable smooth and accurate movements, making it suitable for delicate tasks requiring fine manipulation.

One notable feature of the Jaco2 is its lightweight and compact design. This makes it highly portable and easy to integrate into different environments, whether it's a laboratory, manufacturing facility, or even a home setting. The robot arm is equipped with a range of sensors, including force/torque sensors and position encoders, which enable it to perceive and interact with its surroundings. This allows for safe and collaborative operation, as the robot can detect and respond to external forces, ensuring human safety during interactions. The Jaco2 robot can be controlled using various programming interfaces, such as ROS (Robot Operating System), which provides a flexible and powerful framework for developing robotic applications. It also supports a wide range of programming languages, making it accessible to developers with different backgrounds and preferences.

In our setup, we added a tray at the end of the robotic arm and grasped it using the fingers at the end of the arm. Due to the limited strength of the robotic arm's fingers, the mass of this board needs to be as small as possible. At the same time, a cup is placed on top of the board. Our goal is to transport the cup to a designated location without spilling the water, as shown in Fig 2. Meanwhile, in order to capture the motion of the bottle, we use a motion capture.

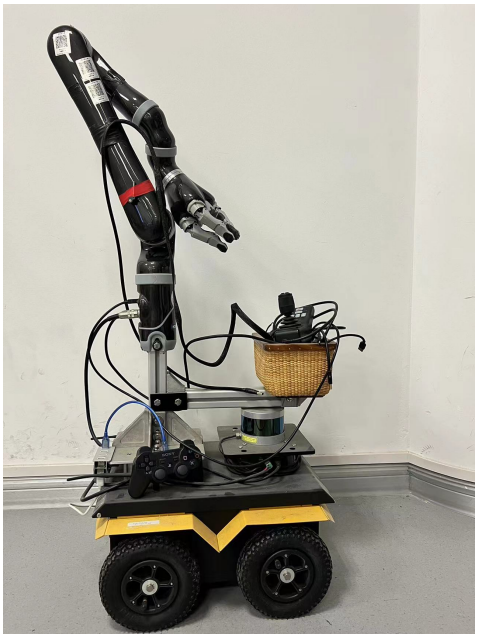

Figure 1: Hardware platform: Kinova Jaco2 and Clearpath Jackal

Figure 2: Hardware platform: Kinova Jaco2 and Clearpath Jackal with a tray

2.2 Problem Formulation

Our task can be separated into two parts. First, after the robot gets a target destination, the Cleanpath Jackal robot will go to a place where the distance between the target point and the end effector is smaller than the length of the Kinova arm. This means that the car needs to get to a place where the target point is in the reachable set of the end effector.

2.2.1 Robot Model

The type of robot we are targeting with our approach consists of a mobile base with a manipulator with 9 degrees of freedom (DOF). The base can turn in place but has a non-holonomic constraint and cannot drive sideways. A kinematic model is used in the MPC. The arm joint positions describe the state of the arm $x_{arm} \in \mathbb{R}^{\dim(\mathbf{6})}$, and the full base pose $\boldsymbol{x}_{\text{base}} = [x, y, \varphi_{\text{yaw}}]$ is used in the system model. The arm is controlled with joint angles $\boldsymbol{u}_{\text{arm}} = [\varphi_1, \ldots, \varphi_n]$. The base's wheel speeds are calculated from the desired base twist, the forward velocity, and the turning rate $\boldsymbol{u}_{\text{base}} = [v, \dot{\varphi}_{\text{base}}]$.

We consider a velocity-controlled mobile manipulator with state $\boldsymbol{x} = [\boldsymbol{q}^T, \boldsymbol{v}^T, \dot{\boldsymbol{v}}^T]^T$, where $\boldsymbol{q}$ is the generalized position, which includes the planar pose of the mobile base and the arm's joint angles, and $\boldsymbol{v}$ is the generalized velocity. We include acceleration in the state and take the input $\boldsymbol{u}$ to be jerk, which ensures a continuous acceleration profile [2]. The input is double-integrated to obtain the velocity commands sent to the actual robot. We require only a kinematic model, which we represent generically as

$$\dot{\boldsymbol{x}} = \boldsymbol{a}(\boldsymbol{x}) + \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u},$$

with $\boldsymbol{a}(\boldsymbol{x}) \in \mathbb{R}^{\dim(\boldsymbol{x})}$ and $\boldsymbol{B}(\boldsymbol{x}) \in \mathbb{R}^{\dim(\boldsymbol{x}) \times \dim(\boldsymbol{u})}$.

2.2.2 Kinematic Model of the Robotic Arm

The robotic arm mounted on the aforementioned mobile base is a 6-DOF arm. The end-effector pose of the robotic arm is denoted by $\mathbf{x}^a := \left[\mathbf{p}^{a\top}, \theta^{a\top}\right]^\top$, where $\mathbf{p}^a \in \mathbb{R}^3$ is the end-effector position in the

robotic arm base frame represented in the Cartesian coordinates and θ^a is the end-effector orientation. Here, we use the $SO(3)$ group to avoid representation singularities and error definition discontinuities. To do so, we define the mapping function $f : SO(3) \to \mathcal{F} \subset \mathbb{R}^9$, such that, for a rotation matrix $R \in SO(3)$,

where $[R]_i, i \in \{1, 2, 3\}$, is the i-th column vector of the rotation matrix R. Thus, the orientation vector θ^a is $\theta^a = f\left(R_E^b\right) \in \mathcal{F}$, and R_E^b is the rotation matrix of the end-effector in the base frame of the robotic arm.

Using such a representation, the kinematic model of the robotic arm can be described using the analytical Jacobian $\mathbf{J}_a$ of the forward kinematics transformation matrix $\mathbf{T} \in SE(3)$ derived using the DH-parameters of the robotic arm as

$$\mathbf{x}_{k+1}^a = \mathbf{f}_{ra}\left(\mathbf{x}_k^a, \mathbf{q}_k, \dot{\mathbf{q}}_k\right) = \mathbf{x}_k^a + \tau \mathbf{J}_a\left(\mathbf{q}_k\right) \dot{\mathbf{q}}_k \tag{1}$$

where $\mathbf{x}^a = \left[\mathbf{p}^{a\top}, \theta^{a\top}\right]^\top \in X_{ra} \subset \mathbb{R}^{12}$ is the state vector defined using θ^a from (1), $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^\top \in Q \subset \mathbb{R}^6$ is the joint angles vector, $\dot{\mathbf{q}} \in \Omega \subset \mathbb{R}^6$ is the joint velocities vector, and, the analytical Jacobian $\mathbf{J}_a$ is given by $\mathbf{J}_a := \partial \mathbf{T} / \partial \mathbf{q}$.

The constraint sets for the end-effector X_{ra}, joint angles Q, and joint velocities Ω are defined by

$$\begin{aligned} X_{ra} &:= [\underline{x}^a, \bar{x}^a] \times \left[\underline{y}^a, \bar{y}^a\right] \times [\underline{z}^a, \bar{z}^a] \times \mathcal{F}, \\ Q &:= \left\{\mathbf{q} \in \mathbb{R}^6 \mid \underline{q}_i \leq q_i \leq \bar{q}_i, \forall i \in \{1, \ldots, 6\}\right\} \\ \Omega &:= \left\{\dot{\mathbf{q}} \in \mathbb{R}^6 \mid \|\dot{\mathbf{q}}\|_\infty \leq \dot{q}_{\max}\right\} \end{aligned} \tag{2}$$

where $\underline{q}_i$ and $\bar{q}_i$ denote the lower and upper limits of the joint angles, respectively.

In order to be able to keep track of the joint angles and consider joint constraints, we extend system (1) to

$$\begin{bmatrix} \mathbf{x}_{k+1}^a \\ \mathbf{q}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_k^a \\ \mathbf{q}_k \end{bmatrix} + \tau \begin{bmatrix} \mathbf{J}_a(\mathbf{q}) \\ I_{6\times 6} \end{bmatrix} \dot{\mathbf{q}}_k \tag{3}$$

where $\left[\mathbf{x}^{a\top}, \mathbf{q}^\top\right]^\top \in \bar{X}_{ra} \subset \mathbb{R}^{19}$ is the concatenated state vector, and $\bar{X}_{ra}$ is the state constraint set for the new augmented model and is defined as $\bar{X}_{ra} := X_{ra} \times Q$.

3 Model Predictive Control Method

A model predictive control (MPC) module generates control inputs for the robot to follow an end-effector trajectory while respecting several constraints. In the following sections, we describe the algorithm, the system model used, the cost-function, and our soft constraints mechanism.

3.0.1 NONLINEAR MODEL PREDICTIVE CONTROL

In this section, we formulate an NMPC scheme for the end-effector pose stabilization of the mobile manipulator. To this end, we define

$$\begin{aligned} \mathcal{U}_N &:= \left(\begin{bmatrix} \mathbf{u}_k \\ \dot{\mathbf{q}}_k \end{bmatrix}, \begin{bmatrix} \mathbf{u}_{k+1} \\ \dot{\mathbf{q}}_{k+1} \end{bmatrix}, \ldots, \begin{bmatrix} \mathbf{u}_{k+N-1} \\ \dot{\mathbf{q}}_{k+N-1} \end{bmatrix}\right) \text{ and} \\ \mathcal{X}_N &:= (\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+N}) \end{aligned} \tag{4}$$

as the sequences of controls and states over the prediction horizon $N \in \mathbb{N}$, respectively. As standard in NMPC, these sequences are used to form the quadratic cost function

$$\begin{aligned} J\left(\mathcal{U}_N, \mathcal{X}_N\right) = & \underbrace{\left\|\mathcal{E}_N^{\mathbf{p}}\right\|_{\mathcal{S}^{\mathbf{p}}}^2 + \left\|\mathcal{E}_N^{\theta}\right\|_{\mathcal{S}^{\theta}}^2}_{J_f} \\ & + \sum_{i=k}^{k+N-1} \left\|\mathcal{E}_k^{\mathbf{p}}\right\|_{\mathcal{Q}^{\mathbf{p}}}^2 + \left\|\mathcal{E}_k^{\theta}\right\|_{\mathcal{Q}^{\theta}}^2 + \left\| \begin{array}{c} \mathbf{u}_i \\ \dot{\mathbf{q}}_i \end{array} \right\|_{\mathcal{R}}^2 \end{aligned} \tag{5}$$

where $\mathcal{S}^{\mathbf{p}} \in \mathbb{R}^{3\times3} \succ 0, \mathcal{S}^{\theta} \in \mathbb{R}^{9\times9} \succ 0, \mathcal{Q}^{\mathbf{p}} \in \mathbb{R}^{3\times3} \succ 0, \mathcal{Q}^{\theta} \in \mathbb{R}^{9\times9} \succ 0$ and $\mathcal{R} \in \mathbb{R}^{10\times10} \succ 0$ are the weighting matrices of the quadratic cost function, and J_f is the terminal cost of the cost function. $\mathcal{E}^{\mathbf{p}} \in \mathbb{R}^3$ is the translational error of the end-effector pose defined as $\mathcal{E}^{\mathbf{p}} := \mathbf{p} - \mathbf{p}^{\mathbf{r}}$, where $\mathbf{p}^r$ is the reference position, and $\mathcal{E}^{\theta} \in \mathbb{R}^9$ is the orientation error of the end-effector pose defined as

$$\mathcal{E}^{\theta} := \begin{bmatrix} [I_{3\times3}]_1 \\ [I_{3\times3}]_2 \\ [I_{3\times3}]_3 \end{bmatrix} - \begin{bmatrix} \left[\left(R_E^I\right)^\top R_r\right]_1 \\ \left[\left(R_E^I\right)^\top R_r\right]_2 \\ \left[\left(R_E^I\right)^\top R_r\right]_3 \end{bmatrix} \tag{6}$$

where R_r is the reference orientation, and R_E^I is the orientation of the end-effector in the inertial frame calculated as $R_E^I = R_b^I R_E^b \cdot R_b^I$ is determined using a localization algorithm of the mobile robot and R_E^b is the rotation matrix from the mobile robot to the end-effector and is calculated from the forward kinematics of the robotic arm, i.e. $\mathbf{T}$.

Using the cost function in Eq. (13), the NMPC optimal control problem can be formulated as:

$$(\mathcal{U}_N^*, \mathcal{X}_N^*) = \underset{\mathcal{U}_N \in U, \mathcal{X}_N \in X}{\arg\min} \; J\left(\mathcal{U}_N, \mathcal{X}_N\right) \tag{7a}$$

$$\text{subject to} \quad \begin{bmatrix} \mathbf{x}_{k+1} & \mathbf{q}_{k+1} \end{bmatrix}^\top - \mathbf{f}_{mm}\left(\mathbf{x}_k, \mathbf{q}_k, \mathbf{u}_k, \dot{\mathbf{q}}_k\right) = 0, \tag{7b}$$

$$\mathcal{X}_N \in X \subseteq \mathbb{R}^{19}, \tag{7c}$$

$$\mathcal{U}_N \in U \subseteq \mathbb{R}^{10}, \tag{7d}$$

$$\left|\det\left(\mathbf{J}_a \mathbf{J}_a^T\right)\right| > \epsilon \tag{7e}$$

where ϵ is a threshold for avoiding singular configurations of the robotic arm.

OCP (7) is converted to a nonlinear programming problem (NLP) using the direct multiple-shooting method. Here, both the control sequence $\mathcal{U}_N$ as well as the state sequence $\mathcal{X}_N$ are considered as decision variables in the optimization problem. Moreover, the system model is considered as an optimization constraint as formulated by Eq. (7c). Multipleshooting discretization technique provides a more computationally efficient solution to OCP when compared with other discretization techniques, e.g. single-shooting, see for more details. Finally, state and control constrains are considered by means of Eq. (7d) and (7e). Note that the inequality constraint (7a) is added to avoid kinematic singularities of the robotic arm through operation. This is accomplished through ensuring that the pseudo-inverse of the robot arm Jacobian matrix is always invertible and, thus, singular configurations are avoided.

The feedback control law can now be stated as

$$\begin{bmatrix} \mathbf{u}_k^* & \dot{\mathbf{q}}_k^* \end{bmatrix}^\top := \mathcal{U}_N^*(0) \tag{8}$$

i.e. the feedback control is the first element in the optimal control sequence $\mathcal{U}_N^*$. Moreover, the resulting feedback system can be stated as

$$\begin{bmatrix} \mathbf{x}_{k+1} & \mathbf{q}_{k+1} \end{bmatrix}^\top = \mathbf{f}_{mm}\left(\mathbf{x}_k, \mathbf{q}_k, \mathbf{u}_k^*, \dot{\mathbf{q}}_k^*\right). \tag{9}$$

3.1 Goal-Conditioned Reinforcement Learning Method

We used a goal-conditioned reinforcement learning (GC-RL) method to learn the kinematics of the kinova robot arm. The GC-RL method set the target as an additional information to the state of the agent. The original state of the arm is a 12 dimension vector contains the angle and velocity of the 6 joints. However, in GC-RL, we add the position of the target into the state, including the coordinate of x, y, z. Therefore, the state dimension of the agent is 15. Also, for the waiter problem, we need to transport the water without sliding it, the position of the cup is in the state. The full state is a 16 dimension vector. The advantage of GC-RL is that it can transform to other goal easily, which means it is data efficient. As we all know, one of the problem to apply RL to robotic is that there is limited training data. For example, in this robot arm reaching task, the traditional method need to generate target randomly, pushing the training data into replay buffer and then train the model. However, it is not efficient and needs huge episode to converge. For the GC-RL, we can use the reward reshape method to easily transform one goal from another.

For the RL algorithm, we choose the soft actor critic (SAC). SAC is a reinforcement learning algorithm that combines the principles of deep Q-learning and policy gradient methods. It is designed to solve continuous control tasks, where the action space is continuous and requires finding optimal policies. SAC is known for its ability to handle both exploration and exploitation efficiently.

The key idea behind SAC is to learn a stochastic policy that maximizes the expected cumulative reward while simultaneously learning an estimate of the state-action value function. SAC achieves this by using an entropy regularization term that encourages exploration and prevents premature convergence to suboptimal policies. By explicitly maximizing policy entropy, SAC can strike a balance between exploration and exploitation, leading to more robust and diverse policy search. One of the advantages of SAC is that it can handle environments with high-dimensional state spaces and continuous action spaces effectively. It achieves this by employing a deep neural network as a function approximator to represent the policy and value function. The neural network is trained using a combination of policy gradient and Q-learning techniques, enabling SAC to learn directly from raw sensory inputs. Another notable feature of SAC is the use of twin Q-networks and target networks. Twin Q-networks help reduce overestimation biases in estimating the state-action value function, improving the stability and performance of the algorithm. Target networks, on the other hand, are used to compute target values during the training process, providing more stable and accurate value estimates. To build the RL environment for the kinova robot arm, we register a openai Gym environment kinova-gym. It achieves an environment with feedback and updates by inheriting the standard OpenAI Gym class.

The action for the arm is the radian of the joint, it is a gym.space.box object with the range between -1 and 1. Then, the actions of the joints will be mapped to the degrees as $'0-360', '0-180', '90-270', '0-360', '0-360', '0-360'$. The second joint has a range limits of 0 to 180 degrees because when the angle is bigger than 180 degree, it may touch the ground or the mobile base.

For the state of the robot, it is a gym.spaces.box object. We can subscribe the the topic "$/gazebo/link-states$" to get the state information. This topic can return a object contains the name list of the joints, the position information and the velocities of the joints. The list is a 14 items long list contains all the arm and fingers. The details are as follow:

$['ground-plane :: link', 'j2n6s300 :: root', 'j2n6s300 :: j2n6s300-link-1', 'j2n6s300 :: j2n6s300-link-2', 'j2n6s300 :: j2n6s300-link-3', 'j2n6s300 :: j2n6s300-link-4', 'j2n6s300 :: j2n6s300-link-5', 'j2n6s300 :: j2n6s300-link-6', 'j2n6s300 :: j2n6s300-linkfinger-1', 'j2n6s300 :: j2n6s300-link-finger-tip-1', 'j2n6s300 :: j2n6s300-link-finger-2', 'j2n6s300 :: j2n6s300-link-finger-tip-2', 'j2n6s300 :: j2n6s300-link-finger-3', 'j2n6s300 :: j2n6s300-link-finger-tip-3']$

For the reward function of this task, we have two types of reward function: continuous and sparsed. For the continuous setting, we calculate the negative value of the L2 norm of the position of the end effector. For the sparsed setting, we set a threshold of the minimum distance of the end effector and the target point. Then we will calculate the negative value of the L2 norm of the distance then do a item based comparation for the x, y ,z dimension. Also, as we want to achieve a nonprehensile control, we add the tilt angle of the bottle to the reward function, it is calculated by the motion capture.

For the SAC algorithm, we achieve it by ourselves. For the critic network, we use a 3 layers neural network with RELU as the active function. Meanwhile, we use the Gaussion policy to approximate the bolzman function. Therefore, in order to let the two function close enough, we need to minimize the KL divergence. To update the critic network, we minimize the TD error. To update the policy network, we use the policy gradient equation.

3.2 Challenge

Due to performance limitations of our platform compared to the one used in the reference paper, we acknowledge that we may not be able to achieve all the objectives outlined in the paper. However, we are committed to tackling challenging tasks within our capabilities. For instance, we aim to develop a platform that can successfully transport a bottle of water to a predefined location while ensuring no water spillage. To achieve this, key components such as real-time control, accurate perception of the bottle's state, and precise End-Effector(EE) translation and attitude adjustments are crucial. We are aware that these challenges require considerable effort, but we are confident that through dedication and hard work, we will be able to address them.

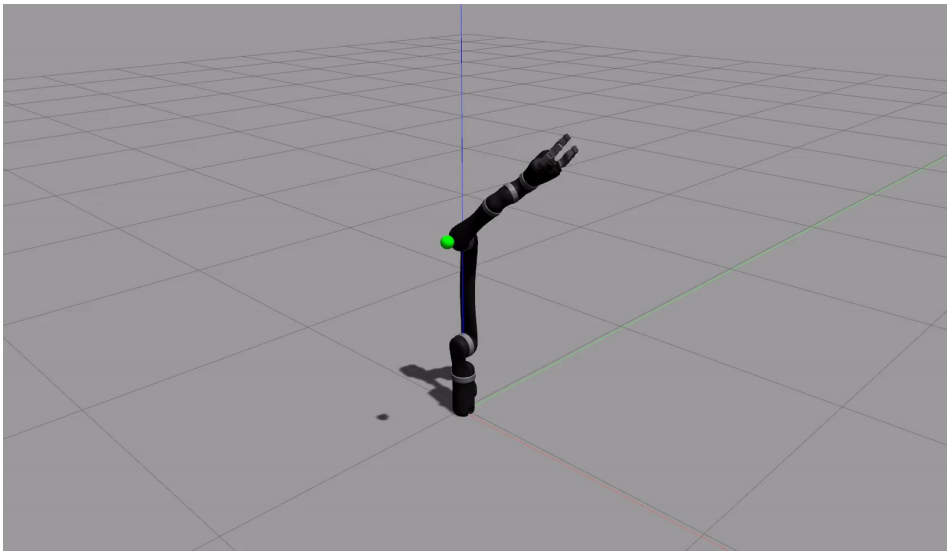

Figure 3: Training Kinova With Reinforcement learning in gazebo

4 System Evaluation

We test our algorithm for both the MPC and GC-RL.

4.1 Water's problem Experiment

We plan to use a real velocity-controlled mobile manipulator balancing up to seven objects; balancing an assembly of stacked objects; and avoiding static and dynamic obstacles, including a thrown volleyball. Our goal is that the EE achieves speeds and accelerations up to 2.0 m/s and 7.9 m/s2, respectively. However, due to the out-of-control project progress management, although we have made a lot of efforts, we still do not have the expected functions.

Once we have constructed the forward kinematics for the j2n6s300, obtaining the angles of the six joints will yield the corresponding Cartesian position. With this model, we can evaluate whether each trajectory planned by MoveIt! will result in spillage. If it does, we discard the current path planning and initiate a re-planning process. Our method requires the forward kinematic model of the Kinova Jaco2 j2n6s300 robotic arm for predicting the end-effector position iteratively to obtain the optimal solution. Therefore, an accurate model is crucial. The computation of the end-effector position of the robotic arm can be achieved using the Denavit-Hartenberg (DH) parameters and the forward kinematics equation.

Parameter	Description	Length (m)
D1	Base to shoulder	0.2755
D2	Upper arm length (shoulder to elbow)	0.4100
D3	Forearm length (elbow to wrist)	0.2073
D4	First wrist length (center of actuator 4 to center of actuator 5)	0.0741
D5	Second wrist length (center of actuator 5 to center of actuator 6)	0.0741
D6	Wrist to center of the hand	0.1600
e2	Joint 3-4 lateral offset	0.0098

Parameter	Description	Value
aa	Half of the angle of curvature of each wrist segment (60°), measured in radians.	$\frac{30.0\pi}{180.0}$
sa	Sine of half the angle of curvature of wrist segment.	$sin(aa)$
s2a	Sine of angle of curvature of wrist segment.	$sin(2aa)$
d4b	Length of straight-line segment from elbow to end of first sub-segment of first wrist segment.	$D3 + (\frac{sa}{s2a})D4$
d5b	Length of straight-line segment consisting of second sub-segment of first wrist segment and first sub-segment of second wrist segment.	$(\frac{sa}{s2a})D4 + (\frac{sa}{s2a})D5$
d6b	Length of straight-line segment consisting of second sub-segment of second wrist segment and distance from wrist to the center of the hand.	$(\frac{sa}{s2a})D5 + D6$

T_i^{i-1}	α_{i-1}	a_{i-1}	d_i	θ_i
T_1^0	$\frac{\pi}{2}$	0	D1	q_1
T_2^1	π	D2	0	q_2
T_3^2	$\frac{\pi}{2}$	0	-e2	q_3
T_4^3	$2aa$	0	-d4b	q_4
T_5^4	$2aa$	0	-d5b	q_5
T_6^5	π	0	-d6b	q_6

$q_1(\text{physical})$	$-q_1(robot)$
$q_2(\text{physical})$	$q_2(robot) - 90°$
$q_3(\text{physical})$	$q_3(robot) + 90°$
$q_4(\text{physical})$	$q_4(robot)$
$q_5(\text{physical})$	$q_5(robot) - 180°$
$q_6(\text{physical})$	$q_6(robot) + 90°$

Figure 4: DH parameters

The transformation matrices associated to each link could be obtained following the DH convention. That transformation matrix from frame i to frame $i-1$ is given by

$$T_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\alpha_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As soon as the DH parameters are known, the transformation matrices associated to each link could be calculated by applying the equation, then the forward kinematics can be calculated through the equation depicted as follows.

$$T_e^b(q) = T_1^b(q_1)\, T_2^1(q_2) \ldots T_n^{n-1}(q_n) \tag{10}$$

where n = 6 for the robotic manipulator. The experimental results are showcased in the video, and the implementation details can be found in the README.

4.2 GC-RL Experiment

For the RL experiment, we didn't test it on the hardware (in the real environment). We only test our RL algorithm in the gazebo simulation. We compared our method with the general RL method. We trained both algorithm for 1000 episode and max horizon is 128. The result is shown in Fig and table. As we can see, our method realize a faster convergence rate and higher success rate,

Table 1: The success rate of the task

	GC-RL	**General RL**
Average Success Rate in 20 second	0.98	0.86

5 Conclusions

In this project, we want to reappearance an MPC-based approach for balancing objects with a velocity-controlled mobile manipulator and demonstrated its performance in simulated and real experiments in a static scenario. Based on several reference materials, we choose the [9] as our main reference. In real experiments part, we use Kinova JACO2 and control it by using MoveIt ROS package.

Also, we tried a goal-conditioned reinforcement learning method to learning the kinematic and the waiter problem. The experiment result shown our method can achieve the goal well.

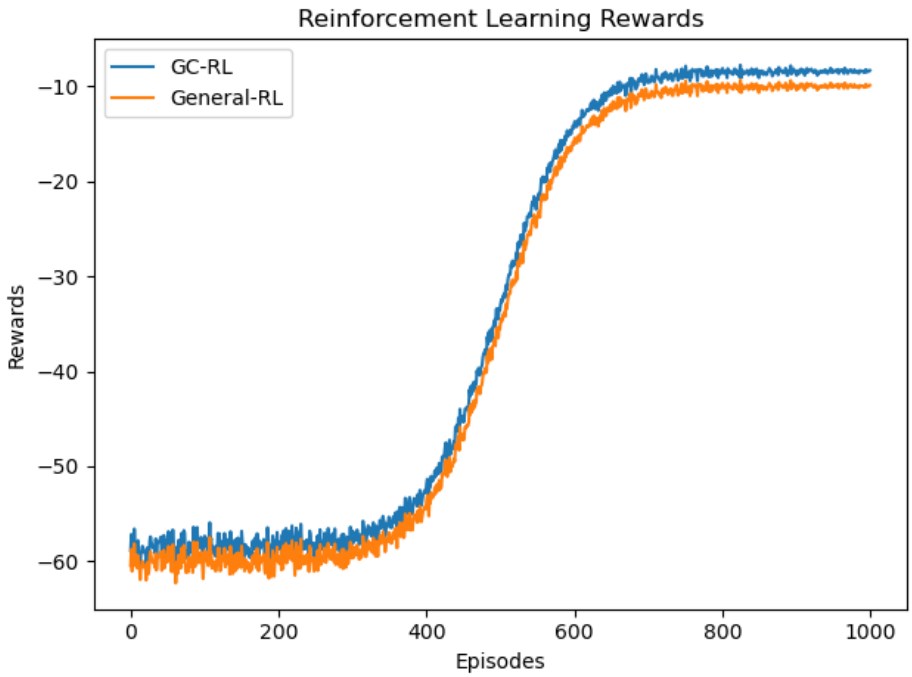

Figure 5: Reward of different RL method

References

[1] P. Acharya, K.-D. Nguyen, H. M. La, D. Liu, and I.-M. Chen, "Nonprehensile manipulation:a trajectory-planning perspective," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 1, pp. 527–538, 2021.

[2] M. Selvaggio, A. Garg, F. Ruggiero, G. Oriolo, and B. Siciliano, "Non-prehensile object transportation via model predictive non-sliding manipulation control," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 5, pp. 2231–2244, 2023.

[3] V. Morlando, M. Selvaggio, and F. Ruggiero, "Nonprehensile object transportation with a legged manipulator," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 6628–6634, 2022.

[4] J. Bernheisel and K. Lynch, "Stable transport of assemblies: pushing stacked parts," *IEEE Transactions on Automation Science and Engineering*, vol. 1, no. 2, pp. 163–168, 2004.

[5] M. Kennedy, K. Queen, D. Thakur, K. Daniilidis, and V. Kumar, "Precise dispensing of liquids using visual feedback," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1260–1266, 2017.

[6] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, "Gomp-fit: Grasp-optimized motion planning for fast inertial transport," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 5255–5261, 2022.

[7] B. Sangiovanni, A. Rendiniello, G. P. Incremona, A. Ferrara, and M. Piastra, "Deep reinforcement learning for collision avoidance of robotic manipulators," in *2018 European Control Conference (ECC)*, pp. 2063–2068, 2018.

[8] T. Matsubara, S.-H. Hyon, and J. Morimoto, "Learning parametric dynamic movement primitives from multiple demonstrations," *Neural Networks*, vol. 24, no. 5, pp. 493–500, 2011.

[9] A. Heins and A. P. Schoellig, "Keep it upright: Model predictive control for nonprehensile object transportation with obstacle avoidance on a mobile manipulator," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7986–7993, 2023.