Robo Waiter: Final Report

Nuran Kasthuriarachchi
nuran@u.nus.edu

Louth Rawshan
e0546303@u.nus.edu

Guilhem Mathieux
e1154518@u.nus.edu

1 Overall goal

The primary objective of this project is to design a robot capable of assisting with everyday tasks. Specifically, our team will concentrate on a common activity in daily routines: opening a bottle and subsequently pouring its contents into a glass.

We aim to create a robotic system with the following capabilities:

1. Grasp the cap and unscrew it
2. Grasp the bottle and move it to a position above the glass
3. Adjust the bottle's position over the glass and pour without spillage.

The code for this project can be found in the public github repository at this link : RoboWaiter

2 Literature review

2.1 Uncapping

In (1), the authors use custom-made grippers and sensors on a UR5 robotic arm in order to remove caps from bottles and jars. Force-sensing resistors enable them to identify when the cap has been successfully removed. In (2), a robot is able to flick off a cap from a bottle by mimicking the motion of a human. In (3), an automated system for removal of test tube caps in a factory is suggested. This scenario is however, quite different from ours as we will have to move our robot arm to arbitrary positions as opposed to fixed test tube locations.

2.2 Pouring

In (4), the authors learn pouring behaviour from human demonstrations and subsequently generalize to further scenarios using self-supervised learning. Using a LSTM based model, they are able to achieve better pouring accuracy than a regular human with a similar pouring speed. In (5), the Navier-Stokes model is used to provide fine-grained information of the velocity distribution inside the liquid body. Using this information, an algorithm is implemented that can compute a collision free trajectory for a robot manipulator to pour liquid from one container to another. In (6), a differentiable environment is created in for the purpose of fluid manipulation. Fluid manipulation includes tasks such as creating ripples in water or making a breeze from air.

2.3 What we are doing?

While there exists research on the individual tasks of uncapping bottles and pouring liquids, our method uniquely attempts to explore the combination of both tasks. We also use differentiable simulators in order to complete this task, which we will discuss in further details in later sections.

3 Initial Process

We commenced our project by using the Mujoco simulator to familiarize ourselves with simulation environments and related fundamental principles (such as inverse kinematics, basic PID control), which was a good starting point owing to its extensive online resources. We created an environment containing a UR5 arm as well as a wine bottle (which contained no fluid inside it).

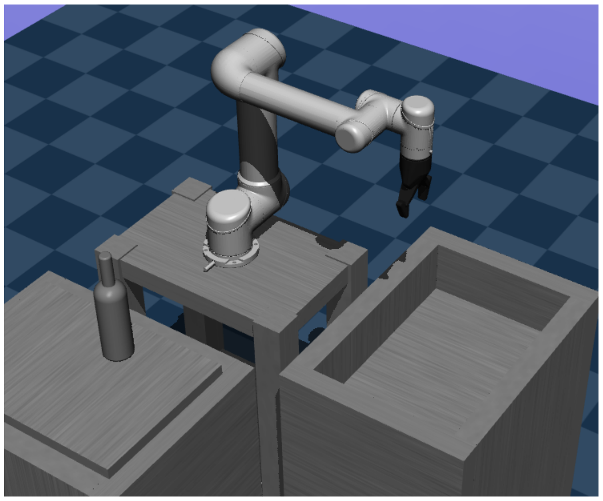

Figure 1: UR5 arm and bottle in the Mujoco simulator

We employed baseline control algorithms to direct the arm to specified locations and manage the gripper's operations.This involved:

1. Utilizing inverse kinematics to deduce joint values configuration for the desired end effector pose.
2. Employing PID controllers to navigate each joint from its present q-value towards the goal q-value.

Utilizing this ability to position the arm at desired locations, we programmed the following procedure:

1. Relocate end-effector within the bottle's grasping range.
2. Secure the bottle ensuring no slippage.
3. Transport the bottle to a new position while maintaining its upright position
4. Tip it over

A demonstration video is available here.

This initial approach helped us understand the task at hand and gave us insights into the potential issues we had to overcome. However, as the coordinates and steps were coded manually, it lacked flexbility and generalizability.

4 Challenges faced

Upon further experimentation in the Mujoco environment, we discovered serious limitations.

1. Using the inverse kinematics method provides the information on what the final state should look like, but not how to get there, i.e., there is no motion planning involved. This results in our manipulator getting twisted in awkward orientations during its path, and thereafter unable to reach the goal pose. In order to overcome this, we had to manually breakdown certain movements into a few sections, i.e., instead of moving from pose p_{start} to pose p_{goal} directly, we had to move from p_{start} to $p_{intermediate}$ instead.

2. Absence of collision avoidance modules caused initial disruptions like knocking over the bottle. Although hard-coding the robot's motion resolved this, an automated solution is preferable.
3. Incorporating a screwable cap on the bottle was non-trivial. Initial mechanical attempts, i.e., threading both the bottle and cap, were error-prone due to extensive contact between thin surfaces, leading to inaccurate simulation.

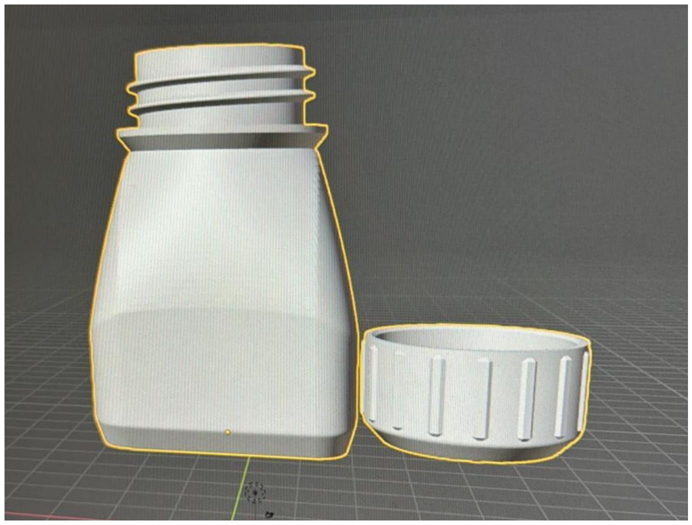

Figure 2: 3D model of a bottle with a screwable cap

5 Solutions

To tackle the above challenges, we decided the on the following workarounds:

1. We decided to use a differentiable simulator instead of Mujoco in order to help with motion planning. The details are explained in the next section.
2. We simulated the cap-bottle relation through a screw joint as opposed to a physical contact using threads.
3. We used the SoftMAC simulator (9) to simulate the liquid in the bottle, due to its realistic fluid simulations and accurate representation of soft-rigid body interactions.

6 Differentiable Simulation and Motion Planning

6.1 Differentiable Simulation

We used the Jade simulator (10) as our main simulation engine. Jade is based on the nimblephysics differentiable simulator, but provides enhanced fidelity of simulation compared to the nimblephysics. It employs a backtracking strategy for collision response, which significantly reduces body penetration during contact. The term "differentiable" comes from the following:

1. The pose of any object in the simulation (for example, the end effector of our robot) is differentiable with respect to the current state (the joint values of the robot)
2. The changes in state between each timestep in the simulation are differentiable with respect to the actions (which in our case is the joint torques of our arm)

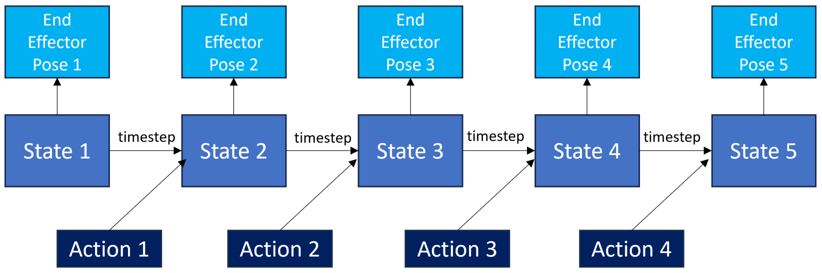

Figure 3: In a differentiable simulator, all the arrows in the above figure are differentiable

6.2 Motion Planning

Hence, we can define a loss function that compares the current pose to some desired pose. Then, we can propagate the gradient of this loss function to the states, and then subsequently to the actions (following the chain rule). Performing gradient descent on these actions should ideally enable us to reach any robot arm pose that we require.

However, such a simple loss function does not work in reality. The robot arm often gets stuck in local minima, especially those involving self-collision where one part of the robot arm collides with another. In order to perform successful motion, we had to include other loss functions (including heuristic loss functions described below). In total, we had the following loss functions:

1. **loss_orientation**: Ensures that the arm has the desired end effector orientation and is usually defined on the final end effector pose. In cases where the orientation has to be kept constant throughout the motion, such as when moving the bottle after grasping where it is essential to avoid tilting, this loss is a summation over states.
2. **loss_move**: Ensures that the end effector has the desired final position.
3. **loss_stop**: Ensures that the arm has final joint velocities close to zero.
4. **loss_avoid_collision**: We store a set of states that previously resulted in collisions. If any of the current states in the path are close to these states, then the pose of the robot arm part that collided is changed.
5. **loss_avoid_floor**: During the path of the robot arm, this prevents the robot arm from crashing to the ground. While this is not strictly necessary because of the loss_avoid_collision, it does improve convergence in most scenarios.
6. **loss_distance_to_center**: Most self-collisions occur when the end-effector approaches the robot base ("the center") during its path planning. This loss ensures that the end effector is discouraged from approaching the base.

The total loss was a weighted sum of these losses.

6.3 Dealing with Gravity

It would be ideal to initialize the actions (on which we will subsequently perform gradient descent) to be zero. However, using this initialization naively to start our motion planning causes the arm to simply fall rapidly and results in difficult to use states and gradients. Instead, we create a function that provides actions that would result in the arm staying mostly still and then use these actions to subsequently fine-tune for motion planning.

7 Grasping and collision simulations

7.1 Grasping of bottle when moving the bottle

We initially attempted to model the grasping of the bottle by the end effector simply as a contact with friction. However, this resulted in much slower simulation and as the training process of motion planning when moving requires several training passes, it became cumbersome to simulate the grasp in this way.

Hence, we instead detected when there was contact between the gripper and the bottle, and then replaced the physical contact with a fixed joint between the gripper and the bottle instead. In this way, we could ignore collisions between the gripper and the bottle and enable efficient training with only a slight compromise on realism (movement of the arm with the friction contact would be mostly similar to simply having the fixed joint).

7.2 Uncapping the bottle: Using a screw joint

As mentioned previously, we decided to use a screw joint to simulate the cap motion. A screw joint imitates the behaviour of the threads: the cap lifts upon rotation. This is a practical alternative to simulation of the contacts between the physical threads of the cap and of the bottle which would be intractable to simulate accurately.

Upon grasping the cap using the end effector, we observed the strong advantage provided by the jade simulator. When using the vanilla nimblephysics engine, the simulation was highly unrealistic with the gripper fingers passing through the cap due to its inability in reducing penetration. On the other hand, in jade, we were able to see a crisp and clean contact, as shown in the figure.

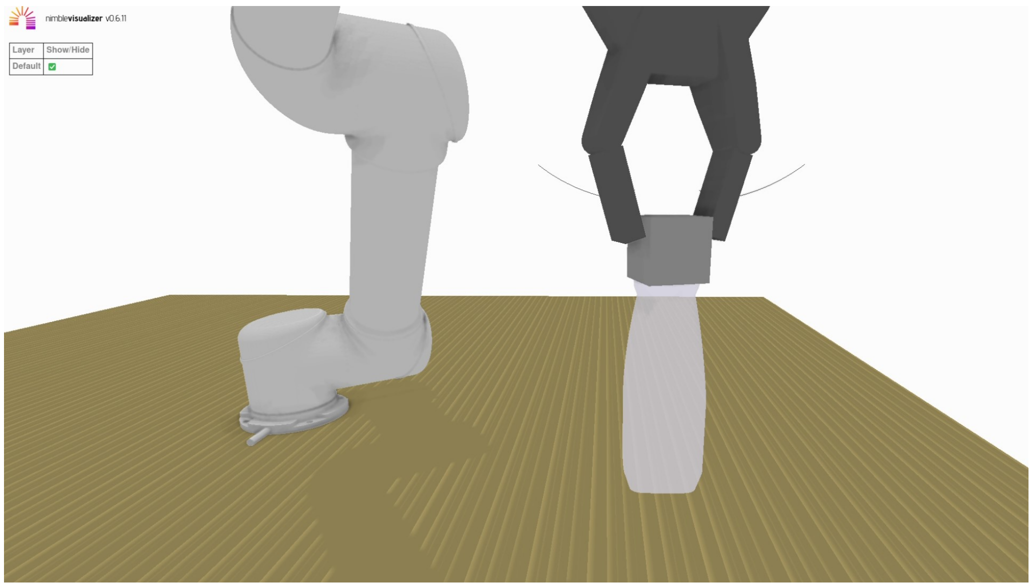

Figure 4: The contact between the cap and gripper is very clean in the Jade simulator

Initially, we attempted to directly optimize the uncapping motion by trying to maximize the screw joint value (increasing the screw joint value would equate to twisting it). However, this resulted in segmentation errors, the cause of which we are unsure. Hence, we opted to instead directly control the motion of the end effector. As shown in our presentation, we were able to accurately simulate the uncapping motion.

8 Simulation of fluids

8.1 SoftMac

To simulate the fluids we looked into using the SoftMAC simulator (9). SoftMac introduces a unified framework for simulating interactions among diverse materials, including soft bodies, articulated rigid bodies, and cloth. The Material Point Method (MPM) used in SoftMAC for soft body simulation, coupled with the forecast-based contact model, addresses complex simulation challenges such as penetration and unnatural rebound. The penetration tracing algorithm proposed in this work enables effective fluid simulation enhancing the realism and applicability of the simulations in various robotic manipulation scenarios.

One key interest of the paper which was very promising to us was the trajectory optimisation for liquid manipulation. The two-way soft-rigid coupling supports gradient calculation which can be used to optimise a control problem.

8.2 Attempts to use SoftMac and results

After discussing with the author Liu Min and testing the code, we got some grasp of the idea behind SoftMAC. We attempted to play around with the trajectory optimization of pouring the contents of a bottle into a cup but found the simulation to be brittle, and for example, removing a small number of particles resulted in completely different results. Hence, we were unable to fully incorporate fluids into our main simulation which included the robot arm.

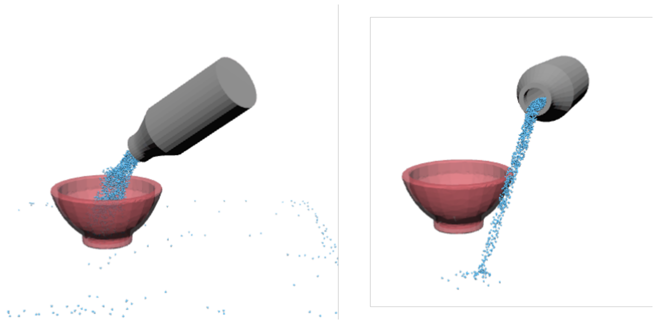

Figure 5: Removing a few particles from the trajectory optimization on the left gives a significantly different result

However, we do recognize that the usefulness of SoftMac in our project and is an aspect that should certainly be developed in a future work in order to further the realism of our simulation.

9 Evaluation

We evaluate our motion planning for horizontal (in order to grasp the bottle) and vertical orientations (in order to grasp the cap) by placing the bottle at 5 very different locations. Our motion planning algorithm is successful in all 5 instances. We also validate the importance of our loss_avoid_collision and loss_avoidcenter by showing the drop in performance upon removing these loss functions.

Loss Type	Bottle Grasping	Cap Grasping
Full Loss Function	5	5
- loss_avoid_collision	4	5
- loss_avoidcenter	4	3

The table above shows the success rate out of 5 attempts. You can view videos of the motion planning algorithm here.

10 Limitations/Future Work

10.1 Simulation Speed Impacted by Collisions

A significant limitation we encountered in our simulations involved the speed of processing. Collisions between the robot arm, the bottle, and other elements within the simulation environment notably slowed down the simulation. As mentioned previously, this caused us to simulate the gripper's grasp on the bottle as a fixed joint rather than a more realistic friction contact. While this simplification helped in managing the computational complexity, it also introduced a limitation in terms of accurately representing the physical dynamics of gripping and manipulation.

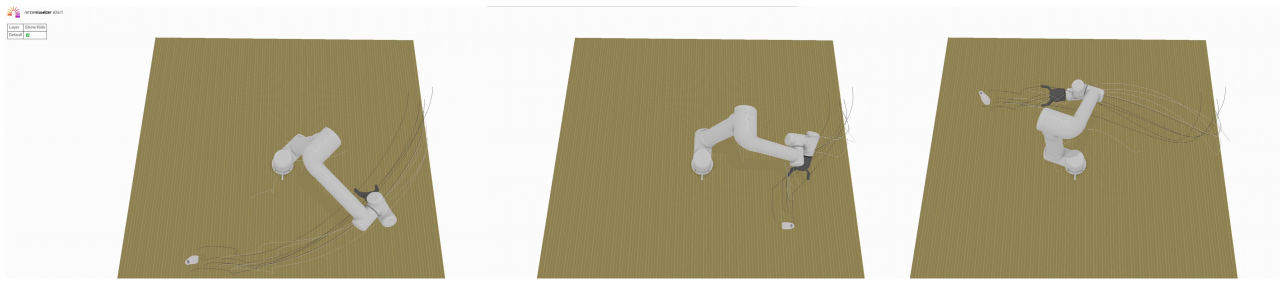

Figure 6: The motion planning algorithm for 3 different bottle locations

10.2 Inability to Fully Remove the Bottle Cap

Our simulation faced difficulties in accurately modeling the complete removal of the bottle cap. We initially simulated the interaction between the cap and the bottle as a screw joint. However, transitioning this screw joint to a floating joint, which would represent the cap being completely unscrewed, proved problematic.

With this limitation, we are only able to model the process of twisting the bottle cap using the end effector, but unable to show the end effector subsequently carrying away the bottle cap.

10.3 Fixed Bottle Assumption

Another limitation of our project is the assumption that the bottle remains fixed during the uncapping process. This was due to our focus on using a single robot arm. The incorporation of a second arm could potentially allow for a more dynamic and realistic simulation, where one arm holds the bottle while the other unscrews the cap, more closely mimicking human-like actions.

10.4 More Efficient Motion Planning System

In our current method, we store a list of states that previously resulted in collision and which we would wish to avoid. However, this method is inefficient, and it would possible be a better idea to develop a method that can predict beforehand whether a certain state will result in a collision or not, and suggest actions based on that. One possible idea is to use a neural network to recommend the next waypoint, as a more intelligent form of motion planning.

11 Conclusion

In this project, we have successfully developed a successful motion planning algorithm for the grasping and manipulation of objects. It was a well-rounded robotics project that confronted us to many key aspects in this domain such as motion planning, collision detection/avoidance and physics simulation.

The project gave us the opportunity to delve into interesting topics such as differential simulation, inverse kinematics, and contact modelling, providing a valuable firsthand experience with these intricate concepts.

Overall, this project served as a practical and captivating extension of the theoretical foundations covered in our lectures. The implementation of these concepts in a tangible project context has not only deepened our comprehension but also broadened our knowledge, equipping us with a more holistic understanding of robotics.

References

[1] B. Lee and M. Jouaneh, "A Robotic System for Securing and Removing a Plastic Bottle Cap," in *2019 IEEE MIT Undergraduate Research Technology Conference (URTC)*, Oct. 2019.

[2] "MIT robot performs Bottle Cap Challenge," *The Robot Report*. [Online]. Available: `https://www.therobotreport.com`. Accessed on: Dec. 5, 2023.

[3] J. W. Jaeger, S. C. Adkins, S. C. Perez-Tamayo, K. E. Werth, G. Hansen, A. J. Nimunkar, and R. G. Radwin, "Automated Device for Uncapping Multiple-Size Bioanalytical Sample Tubes Designed to Reduce Technician Strain and Increase Productivity," *SLAS Technology*, vol. 26, no. 3, pp. 320-326, 2021, Special Collection: Emerging Trends in 3D Cell Culture: High-Throughput Screening, Disease Modeling and Translational Medicine. doi: `https://doi.org/10.1177/2472630320967622`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2472630322011062`

[4] Y. Huang, J. Wilches, and Y. Sun, "Robot gaining accurate pouring skills through self-supervised learning and generalization," *Robotics and Autonomous Systems*, vol. 136, pp. 103692, 2021. doi: `https://doi.org/10.1016/j.robot.2020.103692`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0921889020305327`

[5] Z. Pan, C. Park, and D. Manocha, "Robot Motion Planning for Pouring Liquids," in *International Conference on Automated Planning and Scheduling*, 2016. [Online]. Available: `https://api.semanticscholar.org/CorpusID:18899528`

[6] Z. Xian, B. Zhu, Z. Xu, H.-Y. Tung, A. Torralba, K. Fragkiadaki, and C. Gan, "FluidLab: A Differentiable Environment for Benchmarking Complex Fluid Manipulation," *arXiv preprint arXiv:2303.02346*, 2023.

[7] ROBIOSS, Institut PPRIME UPR CNRS 3346, "2016 - RoBioSS Hand Unscrewing a bottle cap in 'Le Monde' newspaper," [YouTube channel], Dec. 2018. [Online]. Available: `https://www.youtube.com/watch?v=t099beFFljA`

[8] U. Hillenbrand, B. Brunner, C. W. Borst, and G. Hirzinger, "The Robutler: A Vision-Controlled Hand-Arm System for Manipulating Bottles and Glasses," in *ISR 2004 35th International Symposium on Robotics*, 2004.

[9] M. Liu, G. Yang, S. Luo, C. Yu, and L. Shao, "SoftMAC: Differentiable Soft Body Simulation with Forecast-based Contact Model and Two-way Coupling with Articulated Rigid Bodies and Clothes," 2023.

[10] G. Yang, S. Luo, and L. Shao, "Jade: A Differentiable Physics Engine for Articulated Rigid Bodies with Intersection-Free Frictional Contact," *arXiv preprint arXiv:2309.04710*, 2023.

[11] C. Schenck and D. Fox, "Perceiving and Reasoning About Liquids Using Fully Convolutional Networks," *The International Journal of Robotics Research*, vol. XX, no. X, pp. 1–25, 2017. [Online]. Available: `https://www.sagepub.com`.