Tower of Hanoi - Advanced Robot Learning

Dominic Khang *12012414*
Chung-Hsuan Ryan Lee *52112880*
Dominik Pusztai *12009186*
Pascal Zoehrer *11941484*

***Abstract*—In this project, we developed a system to solve the Tower of Hanoi using a 6-degree-of-freedom robot arm by combining different machine learning methods. First, we recorded basic actions such as picking up and placing disks. These movements were used to create Dynamic Movement Primitives (DMPs), which allow the robot to generalize and repeat similar actions. To recognize the disks, we trained a YOLO object detection model. By using OpenAI's o1 reasoning model we are able to generate the sequence of moves required to solve the puzzle. The complete system was tested in a Gazebo simulation, without the implementation of object detection.**

I. Introduction

The Tower of Hanoi is a classic mathematical puzzle characterized by its simple setup, yet an exponentially complex solution as the number of disks increases. In this project we present a pipeline that combines robotic manipulation with advanced machine learning algorithms to solve the puzzle autonomously using a 6-degree-of-freedom robot arm. Using kinesthetic teaching, we are able to record the fundamental movements required for a pick-and-place motion by moving the robot arm in a gravity compensation mode. The recorded movements are then used for training Dynamic Movement Primitives, enabling the generation of trajectories for similar motions with customized start and end positions.

We use a combination of image based object detection and depth perception to locate the puzzle disks in space and actuate the arm accordingly. To train the object recognition model, we recorded a video of all disks and manually assigned disk labels for a single frame. SAM2 is then able to track and segment the target objects. The resulting annotated dataset was then used for transfer learning by fine-tuning a pretrained YOLO model. By combining the object detector with the depth perception capability of a Intel RealSense camera, we are able to locate disks in space, as well as identify their label based on their color and shape.

We leverage the reasoning capabilities of the OpenAI o1 reasoning model to complete the task planning required for solving the puzzle. The Large Language Model is provided with the set of rules that define the Tower of Hanoi and is asked to generate a sequence of actions to relocate the disks in accordance with these rules.

II. Implementation

A. Dynamic Movement Primitives (DMPs)

To enable the robot to perform generalizable pick-and-place motions, we used Dynamic Movement Primitives (DMPs). We first recorded fundamental actions such as pick, lift, place, and home using kinesthetic teaching while the robot operated in gravity compensation mode. Each of these movements was recorded multiple times with varying object positions to allow experimentation. The recorded trajectories were used to train DMPs, allowing the robot to generate smooth and adaptable motions between arbitrary start and goal positions. Since DMPs operate in task space, we apply inverse kinematics to convert the generated paths into joint trajectories suitable for robot control. We evaluated the generalization of the DMPs by modifying goal positions and analyzing the resulting motions in both simulation and real-world execution. The generated paths successfully replicated the style of the original motions while adjusting to the new targets. Figure 1 shows a DMP generated trajectory for a pick operation with a custom start and end location to the left of the robot. For pick and place operations we use three different recordings. The recordings show an operation to the left, right and straight in front of the arm. Depending on the location of the target we choose the corresponding DMP. This improves the accuracy of the DMP generated trajectory and leads to less errors when calculating the inverse kinematics. Velocity and acceleration profiles showed that DMP-generated trajectories were smoother and more stable than the original recordings.

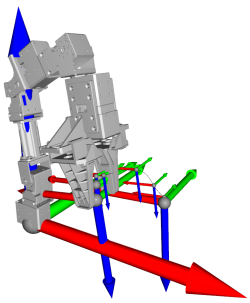

Fig. 1: Generated DMP Trajectory for a Pick Operation

B. Object Detection with YOLO

To enable vision-based automation of the pick-and-place task, we trained a custom YOLOv8 model to detect the disks used in the Tower of Hanoi. The dataset was created by recording a video of the six differently sized disks from multiple angles. A single frame from the video was manually labeled with bounding boxes and class names using the SAM2 tool. SAM2 then automatically propagated these labels to the remaining frames via segmentation, resulting in a full dataset

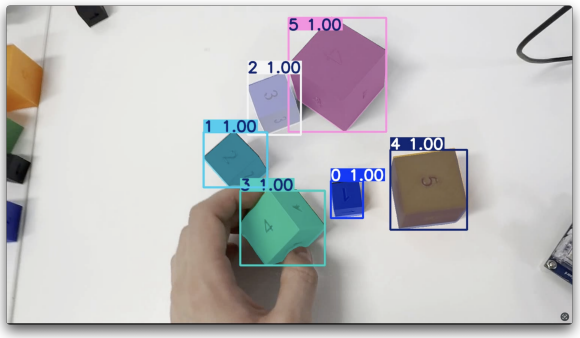

Fig. 2: SAM2 object detection and segmentation

suitable for training. We fine-tuned a pretrained YOLOv8-small model on this dataset. The model achieved a high accuracy, with a mean average precision (mAP) of 99.5% on the validation set. The output of the trained model on a video is shown in figure 2, where the cubes are detected, segmented and labeled correctly. Despite its strong performance, some limitations were observed: visually similar dark-colored disks were occasionally misclassified, and objects with similar color in the background (like a laptop edge) caused confusion. Future improvements include using disks with more distinct colors to improve robustness. To integrate detection with the robot, we used a RealSense camera to acquire images in real time and localize the detected objects in 3D space. While distance measurements from the camera to the disk were accurate up to 1 cm of error, positions relative to the robot base had a higher error (3–5cm). However, in our final solution this detection pipeline was not used, as the complete system was executed in simulation without camera input.

C. Task Planning with a Large Language Model

To generate the correct sequence of moves for solving the Tower of Hanoi, we explored different approaches to using Large Language Models (LLMs). We tested locally hosted models like Gemma3:4b using the Ollama framework, which allowed us to run LLMs offline and evaluate their reasoning abilities. Various enhancements were also explored, including fine-tuning via Group Relative Policy Optimization (GRPO) for better mathematical reasoning, and Retrieval-Augmented Generation (RAG) for supplying the model with external factual knowledge. While these methods showed potential, especially in formal reasoning tasks, they were limited by imperfect accuracy and high hardware requirements, which made them impractical for robust task execution in our setting. For the final project, we chose to use the o1 model from OpenAI via the API, as it delivered consistently correct Tower of Hanoi solutions, even handling illegal states. In addition, it does not require fine-tuning with GRPO or any external factual knowledge about the puzzle by using a RAG system. The model is only provided a simple description of the current state, the goal and a set of rules describing a valid move. Since o1 is a reasoning language model, it is optimized for tasks that require structured thinking and generally outperforms GPTs for solving complex puzzles. We were able to achieve a very high success rate in generating valid and complete move sequences while requiring no local computation, making it the most reliable and efficient solution for our needs.

Example solutions for the Tower of Hanoi puzzle generated with the o1 model can be seen in listing 1 for a normal starting configuration and listing 2 for an illegal starting configuration. The LLM is able to provide the correct solution in both cases, adhering to the puzzle rules. For listing 2 we purposefully chose a difficult illegal starting configuration. Although a move can be saved by executing MD2BA as a first step, the model was able to correctly solve the illegal starting position nevertheless.

```
### Current Problem:
       - Number of Disks: 3
       - Starting State (bottom disk first,
                         top disk last):
           Rod A: [3, 2, 1]
           Rod B: []
           Rod C: []
       - Target State:
           Rod A: []
           Rod B: [3, 2, 1]
           Rod C: []

       Result:
       1. MD1AB
       2. MD2AC
       3. MD1BC
       4. MD3AB
       5. MD1CA
       6. MD2CB
       7. MD1AB
```

Listing 1: Example solution of the Tower of Hanoi puzzle generated using OpenAI's o1 model for a normal, legal configuration

```
### Current Problem:
       - Number of Disks: 3
       - Starting State (bottom disk first,
                         top disk last):
           Rod A: []
           Rod B: [1,2]
           Rod C: [3]
       - Target State:
           Rod A: []
           Rod B: [3, 2, 1]
           Rod C: []

       Result:
       1. MD2BC
       2. MD2CA
       3. MD1BA
       4. MD3CB
       5. MD1AC
       6. MD2AB
       7. MD1CB
```

Listing 2: Example solution of the Tower of Hanoi puzzle generated using OpenAI's o1 model for an illegal starting configuration

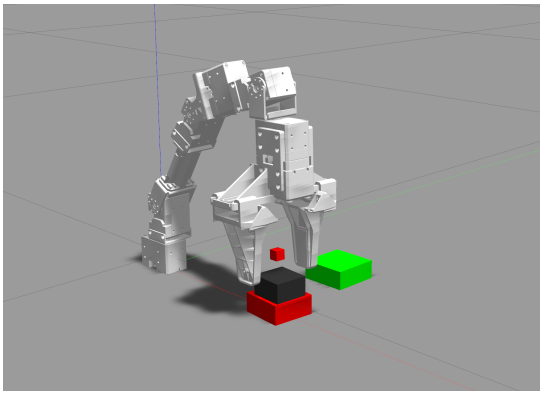

Fig. 3: Pipeline Execution in Gazebo

D. Gazebo Simulation

To integrate all components of our system, we developed a Python script that orchestrates the full pipeline from perception to execution. The process begins by placing one disk on each rod. We then run a script, which uses the object detection model to identify the initial positions of the disks and assign each rod a reference position. Once the rod positions are initialized, we manually arrange the disks into the desired starting configuration of the Tower of Hanoi. The system then scans the new disk positions to determine the current state of the game. This state is converted into a suitable prompt and sent to OpenAI's o1 model, which returns the sequence of moves required to solve the puzzle. For each move, the script updates the internal state by scanning the current configuration, identifies the correct disk to move, and uses the trained DMPs to execute a pick-and-place action. After placing the disk, the system re-scans the scene to confirm the updated state before executing the next move. This loop continues until all moves from the o1 model have been completed and the puzzle is solved. The pipeline without the object detection described in section II-B can be executed in Gazebo, as shown in figure 3. An example run of the full pipeline can be viewed in the video linked in the appendix.

E. Alternative Robot Movement Control

The robot control solution using Dynamic Movement primitives of section II-A has shown to major limitations:

1) The precision of the gripper positioning is often insufficient for reliably grabbing a disk and not disturbing its environment (i.e. other disks in its vicinity) in the process.
2) The computation time required for computing the DMP movement can be excessive, sometimes requiring a wait time of more than 5 minutes for a single DMP run to complete, varying among project team members. This makes it intractable for actively working with this solution during project development.

To mitigate these issues, an alternative robot control solution relying on a single inverse kinematics (IK) computation and simple interpolation has been implemented. This solution performs IK for a desired end-effector (gripper) position with a fixed orientation to obtain target joint positions, which are directly published as target joints for the robot arm. The ROS robot arm controller than directly moves towards these joint positions. Computation times are significantly reduced with this approach, although special care has to be taken in its usage in order to generate next target joint positions which do not prompt the robot to perform an unsuitable movement from its current position. Also, the requested gripper position has to be in the possible action space of the end-effector, as otherwise the IK computation will fail and the robot will not perform any movement at all.
The implementation of the direct movement control is available in the **pick_and_place_direct.py** file under the **execute_direct_motion** function. This file also provides a demonstration when run with an active simulation, note however that the results depend on the disks' being in a suitable position for the robot arm to reach them. For the final demonstrations, the DMP-based movement control has been utilized, as additional measures noted below have improved precision and processing time remained in somewhat reasonable limits for some team members.

III. Results & Discussion

The full pipeline has been shown to execute successfully in simulation, as can be seen in the video linked in the appendix, thereby demonstrating viability of the approach. A number of challenges have arisen over the course of this project and shall be discussed here. The most prominent source of problems concerned the control of the robot's movement for a variety of reasons.

Firstly, the chosen approach using Dynamic Movement Primitives, while powerful in its ability to generalize motions, has shown difficulties to generate precise enough motions for reliably performing stacking. Utilizing a set of different motion recordings, differing in the target disk's position they move to/from, has improved the situation. Another problem with the DMPs were the already mentioned excessive computation times required in some cases, making this a solution a good demonstration but rather impractical. While the direct movement solution of section II-E does significantly shorten computation times, both these solutions are still limited to the actionable space of the robot arm itself, which is due to the robot's size quite small. This limitation has made it especially difficult to achieve stacking using the original disks, which were relatively large in size, leading to only few rod positions at which they could be physically stacked in the robot's end effector action space. The change to flatter disks (i.e. squashed in the vertical dimension) has made it easier to find suitable rod positions reachable for the robot. Additionally, requirements on the arm movement precision have been somewhat relaxed by the change as well.

It is also important to note that the simulation does not use the object detection pipeline and uses the exact positions of the disks instead. Since the object detection introduces

additional error and uncertainty in the gripper/disk interaction, it remains to be seen if stacking can be successfully performed in lab session.

The LLM based reasoning approach has surprisingly worked very well using an advanced model like OpenAI's o1 reasoning model, which illustrates great potential for generalized automation in robotics for this technology. Experiments with locally run LLMs have shown the challenges for practical, offline usage in two ways. Firstly, achieving reliably accurate results is difficult using simpler models, even with various enhancement techniques. Secondly, computational demands and therefore also processing times are high, a problem that becomes worse when utilizing more advanced and complex models for better accuracy. Addressing these challenges is vital for applying robotics in real-time, real-world applications.

IV. Conclusion

The pipeline to solve the Tower of Hanoi puzzle autonomously using a six degree-of-freedom robotic arm has been shown to work in simulation. While our setup does not consistently bring good results, it serves as a demonstration for strengths, challenges and the potential of a variety of machine learning methods for robotic manipulation.
Dynamic Movement Primitives have proven to be a good method for generalizing motions, albeit presenting challenges in achieving precise positioning. Openly available models like YOLO and SAM2 provide means for quick, application-tailored object detection. Utilizing LLMs opens up possibilities for generalizing across reasoning tasks in robotic application. Across all these methods, real-time execution and the associated computational expenses are a major concern, signifying the importance of the development of simultaneously high-performance, compact and power-efficient computing systems specialized for artificial intelligence applications.

V. Appendix

GitHub Repository (with video demo):
github.com/YouProMeNoob/Advanced_Robot_Learning