Grab-o-Matic 3000: The Unstoppable Ball Snatcher Imitation Learning for Robot Manipulation based Ball Catching using Inverse Kinematics

Connor Guzikowski, Eric Vetha

March 27, 2024

Abstract—**We present the Grab-o-Matic 3000, an innovative robotic system designed for ball-catching tasks. The system employs a combination of imitation learning and inverse kinematics to achieve efficient and accurate ball-catching behavior. The imitation learning component involves training a neural network model to imitate expert-like ball-catching actions based on visual observations. In parallel, inverse kinematics calculations are utilized to determine the optimal joint velocities for the robotic arm to intercept the projected ball trajectory in a smooth fashion. The experimental results demonstrate the effectiveness of the proposed approach, showcasing the system's ability to successfully catch balls in a dynamic environment. The integration of imitation learning and inverse kinematics offers a promising framework for enhancing robotic manipulation tasks, with potential applications in various domains requiring dexterous object interaction.**

Index Terms—**Imitation learning, robot manipulation, robotics, deep learning**

I. Introduction

THE scope of our project consisted of us implementing a deep learning model with the purpose of guiding a robotic arm to catch a ball in simulation. To accomplish this, we used the simulator Webots and the provided UR10e arm model.

A. Background and Motivation

Deep learning and robotics have both been increasing in popularity and effectiveness. The two areas share many similarities, leaving a big opportunity for the intersection of the two.

II. Literature Review

In recent years, the integration of machine learning techniques into robotics has garnered significant attention, aiming to enhance robot autonomy and adaptability. In this section, we review relevant literature concerning reinforcement learning (RL) and deep learning techniques applied to robotics.

A. Reinforcement Learning for Inverse Kinematics

Reinforcement learning (RL) has shown promise in solving complex control problems, including inverse kinematics tasks in robotic manipulation. Khatib et al. [1] proposed an RL-based approach for solving the inverse kinematics problem, demonstrating its effectiveness in achieving precise control of robotic manipulators. By formulating IK as a reinforcement learning problem, the authors trained a neural network to map from end-effector positions to joint angles, enabling the robot to learn optimal control policies through trial and error.

B. Deep Learning for Handwriting Recognition

While not directly related to robotics, the application of deep learning in handwriting recognition provides valuable insights into the capabilities of neural networks in learning complex patterns and behaviors. In a notable example, Graves et al. [2] introduced a deep recurrent neural network (RNN) architecture for handwriting recognition, achieving state-of-the-art performance on various benchmark datasets. By leveraging deep learning techniques, the model demonstrated robustness in recognizing diverse handwriting styles and variations, highlighting the potential of neural networks in learning intricate motor skills.

C. Deep Q-Learning for Robot Arm Control

Deep reinforcement learning (DRL) techniques have been increasingly employed in robotic control tasks, offering the ability to learn complex behaviors directly from raw sensor data. In a study by Gu et al. [3], deep Q-learning was applied to a simulated robotic arm control scenario, where the agent learned to manipulate the arm to reach target positions. Through the integration of convolutional neural networks (CNNs) and Q-learning, the model exhibited the capability to acquire effective control policies, enabling the robotic arm to navigate its environment and accomplish specified objectives autonomously.

D. Imitation Learning from Observation

Recent work has explored the advancement of learning methodologies that allow an agent to learn from observing the state transitions of an expert, eliminating the need for explicit action data [4]. This approach opens up new possibilities for leveraging vast amounts of unlabelled video data for learning complex tasks, addressing the challenges of perception and control in a diverse range of applications.

E. Synthesis

The literature review highlights the growing interest in leveraging machine learning methodologies, including reinforcement learning and deep learning, to address challenges in robotics, particularly in the domains of inverse kinematics and imitation learning. By drawing insights from studies spanning RL-based IK solutions, handwriting recognition using deep learning architectures, and DRL for robotic control, this review aims to provide a comprehensive overview of the current state-of-the-art techniques and their applicability to robotic manipulation tasks.

While reinforcement learning and deep learning have shown remarkable success in various robotic applications, they often require extensive training data and computational resources. In contrast, imitation learning offers a promising alternative by leveraging expert demonstrations to learn complex behaviors efficiently. Inspired by the demonstrated capabilities of imitation learning in acquiring sophisticated motor skills from human demonstrations, we propose to utilize this approach for teaching a robot to catch a ball. By imitating human-like movements, the robot can potentially acquire robust ball-catching strategies without the need for explicit programming or exhaustive training.

III. Methodology

A. Overview of the Robotic Platform

The experimental setup in Figure 1 aims to showcase the capabilities of imitation learning in complex robotic hand manipulations. The key components of the platform included:

1) UR10e Robotic Arm: The Universal Robots UR10e was employed as the primary manipulator. Its versatility and precision made it suitable for tasks requiring intricate hand-eye coordination.
2) Dynamic Environment: The environment was deliberately designed to be complex, featuring balls launched from various directions, differing in starting heights and velocities. This variability allowed for a comprehensive evaluation of the robot's adaptability and responsiveness.
3) End Effector (EEF) with Vacuum Gripper: The robot's end effector was equipped with a vacuum gripper mechanism. This enabled the robot to grasp and manipulate objects with precision. A vacuum gripper was used as this project's goal was to demonstrate imitation learning in robot manipulations, not object grasping.

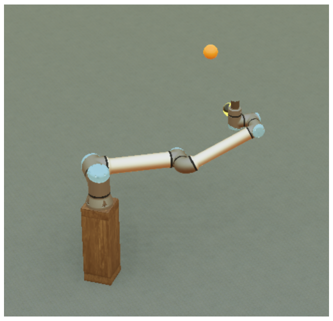

Fig. 1. Webots simulation of robotic platform

The combination of the UR10e robotic arm and the dynamic environment with diverse ball trajectories provided an ideal testing platform for evaluating the efficacy of imitation learning techniques in real-world scenarios.

B. Inverse Kinematics

Inverse kinematics plays a pivotal role in robotics, facilitating the determination of joint parameters necessary to achieve a desired end-effector position and orientation. In this project, inverse kinematics was utilized to enable precise control of the robotic arm for catching a ball thrown towards it.

To compute the inverse kinematics for the robotic arm, the Denavit-Hartenberg (DH) parameters were defined. These parameters describe the geometric and kinematic properties of each joint in the robotic arm. Table I summarizes the DH parameters employed in the system.

TABLE I
Denavit-Hartenberg (DH) Parameters of UR10e Arm [5]

Link	θ_i (rad)	a_i (m)	d_i (m)	α_i (rad)
1	θ_1	0	0.1807	$\frac{\pi}{2}$
2	θ_2	-0.6127	0	0
3	θ_3	-0.57155	0	0
4	θ_4	0	0.17415	$\frac{\pi}{2}$
5	θ_5	0	0.11985	$-\frac{\pi}{2}$
6	θ_6	0	0.11655	0

With the DH parameters established, transformation matrices were employed to represent the orientation and position of each link relative to the preceding link. These transformation matrices were computed using trigonometric functions and the DH parameters.

The final transformation matrix delineates the end-effector's position and orientation in relation to the robot's base frame. This matrix is derived by combining the transformation matrices of all the links.

The Jacobian matrix is computed to establish the relationship between joint velocities and end-effector velocities. This matrix includes both linear and angular components, facilitating motion planning and control.

To minimize the error between the desired and actual end-effector positions and orientations, joint velocities are computed utilizing the error vector and the Jacobian matrix. Proportional control gains are applied to adjust the velocities, ensuring smooth and accurate motion.

The following equations summarize the key calculations involved:

$$\text{Homogeneous Transformation Matrix: } \mathbf{H}(\theta, a, d, \alpha) \tag{1}$$

$$\text{Final Transformation Matrix: } \mathbf{H}_6 = \prod_{i=1}^{6} \mathbf{H}_i \tag{2}$$

$$\text{Jacobian Matrix: } \mathbf{J} = \begin{bmatrix} \mathbf{J}_v \\ \mathbf{J}_\omega \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} R_0^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_n^0 - d_0^0) & \cdots & R_{n-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times (d_n^0 - d_{n-1}^0) \\ R_0^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} & \cdots & R_{n-1}^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{bmatrix}$$

$$\text{Error Vector: } \mathbf{e} = \begin{bmatrix} e_x \\ e_y \\ e_z \\ e_\phi \\ e_\theta \\ e_\psi \end{bmatrix} \tag{4}$$

$$\text{Proportional Gains: } \mathbf{k_p} = \begin{bmatrix} k_{p_t} \\ k_{p_t} \\ k_{p_t} \\ k_{p_a} \\ k_{p_a} \\ k_{p_a} \end{bmatrix} \tag{5}$$

$$\text{Joint Velocities: } \dot{\boldsymbol{\theta}} = \mathbf{J}^\dagger(\mathbf{e} \cdot \mathbf{k_p}) \tag{6}$$

where $\mathbf{H}_i$ represents the homogeneous transformation matrix for joint i, $\mathbf{J}_v$ and $\mathbf{J}_\omega$ are the linear and angular components of the Jacobian matrix, $\mathbf{e}$ is the difference between desired and current pose, $\mathbf{k_p}$ is the vector containing the proportional gains for translation and angular error, and $\mathbf{J}^\dagger$ denotes the pseudoinverse of the Jacobian matrix.

C. Data Collection Process

The data collection process encompasses several sequential steps to gather information requisite for the training of the imitation learning model and the determination of inverse kinematics for ball-catching. Initially, trajectory calculation employs physics-based equations to forecast the anticipated path of the ball, considering variables such as initial velocity, position, and gravitational influences. Subsequently, employing the Denavit-Hartenberg (DH) parameters of the robotic system, inverse kinematics computations provide the necessary joint velocities for the robotic arm to align with the projected ball position. This ensures the execution of robotic movements, facilitated by velocity-based joint control mechanisms. Observational data from two cameras capturing visual cues of the environment and the ball's position are recorded. Furthermore, ground truth actions, denoting the precise location where the ball is caught, are documented. Lastly, the imitation learning model undergoes training, wherein it learns to correlate the visual data acquired from the cameras with the corresponding catching actions. The model is trained without explicit knowledge regarding trajectory calculations, relying solely on the observed data from the camera for learning.

D. Imitation Learning Model

The imitation learning model utilized in this project comprises multiple fully connected layers. The architecture of this model can be found in Figure 2. Three variations of the model architecture were explored:

1) **ImitationModel**: This model consists of three fully connected layers with 64, 32, and *output_size* neurons respectively. Rectified Linear Unit (ReLU) activation functions are applied after each layer to introduce non-linearity.
2) **ImitationModelFiveLayers**: Extending the architecture with two additional fully connected layers, this model includes layers with 64, 128, 64, and 32 neurons followed by ReLU activations. The output layer remains unchanged.

The training process involves optimizing the model parameters using the Adam optimizer with a learning rate of 0.001. Training progresses over multiple epochs, each epoch comprising mini-batch updates with a batch size of 32.

During each epoch, the model is trained on the training dataset, and the mean squared error (MSE) loss between predicted and actual actions is computed. The training loop iterates through the entire dataset in mini-batches, updating the model parameters to minimize the loss.

After training, the model's performance is evaluated on a separate test set to assess its generalization ability. The MSE loss on the test set provides insights into the model's predictive accuracy.

The trained model is then saved for future deployment and analysis. These models have been saved and can be found and tested on our github.

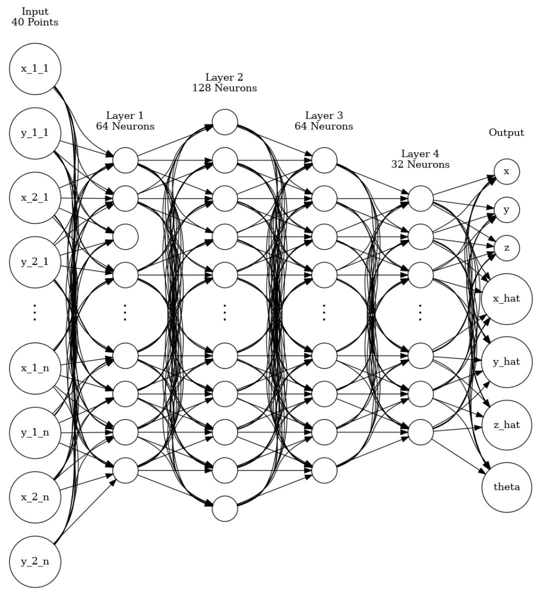

Fig. 2. Five Layer Neural Network Architecture for Imitation Learning Model

IV. Experimental Setup

The experimental setup for the Grab-O-Matic 3000 project involved the installation of necessary software dependencies and the configuration of the robotic simulation environment using Webots. The following steps outline the installation process and setup details:

A. Installation of Dependencies

To ensure proper execution of the Grab-O-Matic 3000 project, the required dependencies are listed in the `requirements.txt` file available in the project's GitHub repository at https://github.com/ericdvet/grab-o-matic-3000.

These dependencies are installed using the following command:

```
pip install -r requirements.txt
```

The `requirements.txt` file contains a list of Python packages essential for the project.

B. Installation of Webots

Webots, a professional robot simulation software, serves as the platform for simulating the robotic environment. The software is obtained from the official website at cyberbotics.com. The appropriate version of Webots corresponding to the operating system (Windows, macOS, or Linux) must be selected and downloaded.

C. Configuration of Robotic Environment

Once Webots and the project dependencies are installed, the robotic simulation environment is configured. The Grab-O-Matic 3000 project repository provides all necessary files and scripts for setting up the environment. This includes Python scripts for controlling the robot's behavior and configuration files defining the robot's physical characteristics and operating environment.

The `imitation_learning.py` file contains the main code for having the robotic arm interact with the environment. The arm's controller can be set to training or testing mode by setting the `LEARNING` flag to `True` or `False`. Setting the flag to `True` has the robot arm run for 10,000 iterations, gathering data on camera frames and the balls target position. Setting the flag to `False` loads the model stored in `imitation_model_five_layers.pth` and uses that to determine where to catch the ball.

V. Results

A. Performance Metrics

To evaluate the performances of the models, we visually observed the success rates, plotted the expected vs generated positions, and also plotted the error of the models in the x, y, and z directions.

From visual observations of the robot arm in action, our three layer model was close to the ball's position but only successfully caught it about 80% of the time. On the other hand, five layer model was successful in catching the ball about 90% of the time. For comparison, using trajectory generation given the ball's initial positions and velocity had success about 98% of the time.

B. Analysis of Imitation Learning Models

Below are plots of the generated vs expected positions, and the errors in the x, y, and z directions. The y-axis of the plots is in the units of meters, while the x-axis represents the sample number being tested.

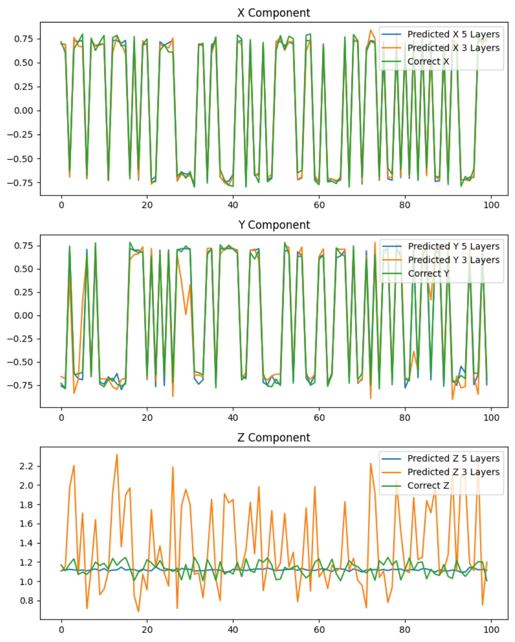

Fig. 3. Comparison of Generated vs Expected Positions for 3 and 5 Layer Models

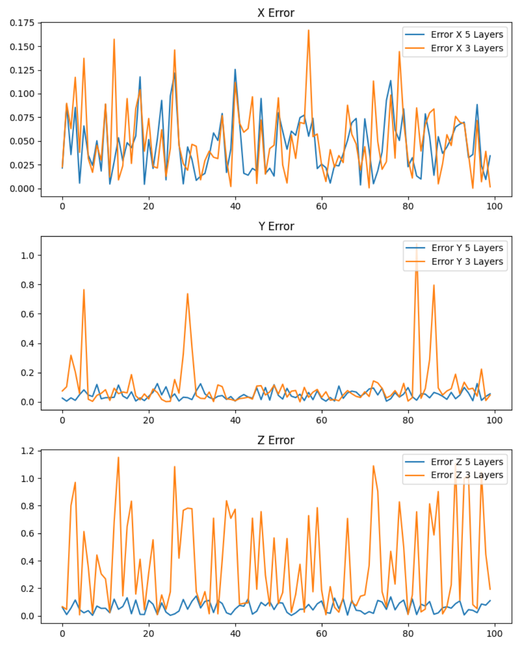

Fig. 4. Comparison of Position Error for 3 and 5 Layer Models

VI. Discussion

A. Interpretation of Results

The results show that using imitation learning with robotic manipulation is possible. Even with very simple models, we are able to see the robot arm catch the ball using information only from the cameras.

B. Potential Improvements

Currently the camera models used are too unrealistic to be put in the real world. They have a near infinite detection ability of the ball, and have no distortion. To be able to use a system like this, a trained ball detection model should be used with the cameras, and the intrinsics of the camera to be used would need to simulated as well.

A slight issue that we cannot figure out is that the robot arm moves slightly after the ball is caught, but only when using the imitation models. We believe the ball is inducing a torque on the joints, but we cannot figure out why this does not happen when we use the trajectory generation method of catching, as it uses the same inverse kinematics as the models.

VII. Conclusion

A. Summary of Findings

In this report, we presented the Grab-O-Matic 3000, a robotic model designed for ball-catching tasks using imitation learning techniques. Through a combination of inverse kinematics calculations and imitation learning from observation, the robot arm demonstrated the ability to successfully catch balls in a dynamic environment. Experimental results showcased the effectiveness of the proposed approach, with the imitation learning models achieving success rates of up to about 90% in ball catching tasks.

B. Contributions and Implications

This project explores using imitation learning to improve robotic manipulation tasks. By using human demonstrations and visual observations, the Grab-O-Matic 3000 provides a simple way to teach robots complex skills without extensive programming. This approach, combining imitation learning and inverse kinematics, has potential applications in industrial automation and assistive robotics.

This research also shows how machine learning can enhance human-robot collaboration. The Grab-O-Matic 3000 demonstrates how combining human expertise with computational intelligence can lead to more advanced robotic systems.

C. Future Directions

Looking ahead, integrating reinforcement learning (RL) into the Grab-O-Matic 3000 could further enhance its capabilities. RL, a type of machine learning where agents learn to make decisions by interacting with an environment, offers exciting prospects for improving robotic systems.

By incorporating RL algorithms, the Grab-O-Matic 3000 could learn from its own experiences and optimize its actions over time. This would enable the system to adapt to changing environments and tasks dynamically.

Moreover, RL could enable the Grab-O-Matic 3000 to learn more complex manipulation strategies autonomously, going beyond imitation learning. Through trial and error, the system could discover novel techniques and refine its behavior based on feedback from its interactions with the environment.

However, it's essential to note that integrating RL poses challenges such as reward design and exploration-exploitation trade-offs. Careful consideration and experimentation would be needed to address these challenges effectively.

VIII. References

[1] O. Khatib, K. Yokoi, and K. Chang, "Reinforcement learning applied to robotic systems," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1609–1624, 1999.

[2] A. Graves, S. Fernández, and F. Gomez, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 545–552.

[3] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," *arXiv preprint arXiv:1603.00748*, 2016.

[4] F. Torabi, G. Warnell, and P. Stone, "Recent advances in imitation learning from observation," 2019.

[5] [Online]. Available: https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/

[6] O. Michel, F. Rohrer, D. Mansolino, and H. W. Guesgen, "Webots: Professional mobile robot simulation," in *Proceedings of the 2022 Conference on Robot Simulation and Synthesis (RoSS)*, 2022.

Acknowledgment

The authors would like to thank Webots for providing the simulation environment used in this research [6].