# Learning-Based Robotic Manipulation for Task Versatility with Kinova Arm

Amirhossein Soltani, Arunima Chaurasia, Gokul Krishna Chenchani, Nigitha Selvaraj

## I. Introduction

Robotics can augment or replace repetitive and intricate human actions. The ability to perform versatile, unstructured manipulation tasks is essential for robots. This enables them to assist with a variety of manual tasks with diverse settings, enhancing functionality and autonomy. This project addresses the challenge of developing an autonomous robotic system capable of visual-guided manipulation using deep reinforcement learning. Specifically, we focus on training a Kinova Gen3 7-DOF robotic arm to locate and track a task board using a combination of visual and proprioceptive feedback, implementing a modern deep learning approach that integrates both image processing and robot state information. The collected dataset is available here.

### A. Motivation

Traditional programming and predefined motion paths would lack the flexibility required for consistent and accurate task execution under changing conditions. Reinforcement learning (RL) presents a suitable solution by allowing the robot to autonomously learn effective policies for each task. With RL, the robot can develop robust strategies to generalize across tasks with complex input-output mappings, adapting to dynamic variations in object placements and control requirements. RL also enables the development of reusable skills, making it possible to generalize learned policies for other similar manipulation tasks.

The ability for robots to perceive and interact with their environment using visual feedback is crucial for many real-world applications, from manufacturing to healthcare. Deep reinforcement learning offers a promising alternative, allowing robots to learn policies that can generalize across different scenarios while handling the complexities of real-world visual input.

### B. Problem Statement

In our project, we aim to develop a learning-based robotic manipulation framework for versatile manipulation tasks using the Kinova Gen3 7-DOF arm with a Robotiq 2F-85 gripper on the task board. The specific challenges we address include: Multi-modal Learning: Developing a system that can effectively combine visual information (from an RGB camera) with proprioceptive feedback (end-effector position and orientation) to guide robot actions. Visual Tracking: Training the robot to actively search for, locate, and maintain visibility of a task board, requiring the system to learn appropriate movement strategies based on visual feedback.

### C. Proposed Approach

Our solution implements a two-phase learning strategy that combines behavior cloning with deep reinforcement learning. Initially, we collect expert demonstrations by recording ROS bags from a real Kinova Gen3 7-DOF arm, capturing synchronized RGB images and end-effector states during successful task execution. These demonstrations are used to train an initial policy through behavior cloning, providing a foundation of human-like tracking behavior. The policy network employs a custom architecture that processes both visual and proprioceptive information: a CNN module handles 84x84x3 RGB images, while a parallel fully connected network processes the end-effector state information. These streams are then fused to output appropriate control actions.

Second solution employs a PPO (Proximal Policy Optimization) algorithm with a custom neural network architecture that processes both visual and state information. The network consists of: A CNN module for processing 84x84x3 RGB images A fully connected network for processing end-effector state information A combined network that fuses both information streams to output appropriate actions The system learns through interaction with a simulated environment in Gazebo, using ROS for robot control and communication. The reward function encourages the robot to maintain visual contact with the task board while executing smooth and efficient movements. This approach represents a step toward more adaptive and generalizable robotic systems that can operate in less structured environments while maintaining robust performance.

## II. Related Work

Hermann et al. [1] address the challenge of training visuomotor control policies in reinforcement learning, particularly in scenarios with sparse rewards and limited demonstrations. They introduce Adaptive Curriculum Generation from Demonstrations (ACGD), a method that adaptively adjusts the difficulty of tasks during training. Instead of relying on manually shaped reward functions or extensive demonstration datasets, ACGD uses a small number of demonstrations and progressively increases task complexity by sampling from different parts of the demonstration trajectories and scaling simulation parameters like domain randomization. The approach ensures that the agent is consistently challenged within a success-rate window, which promotes stable learning. Notably, their method enables effective zero-shot transfer of policies trained entirely in simulation to real-world robotic tasks, such as block stacking and pick-and-stow, achieving high success rates

without additional fine-tuning. This work demonstrates how adaptive curricula and minimal supervision can significantly improve sim-to-real transfer in vision-based manipulation.

## III. Methodology

### A. Simulation

For our study, we used the ros_kortex[1] Gazebo simulation by Kinova-Robotics and integrated it with the task board description provided by HRII Lab[2]. We modeled our Gazebo world with walls and different textures as shown in Fig.1 to differentiate visual images when training. The other views of the simulation setup are shown in Fig.2.

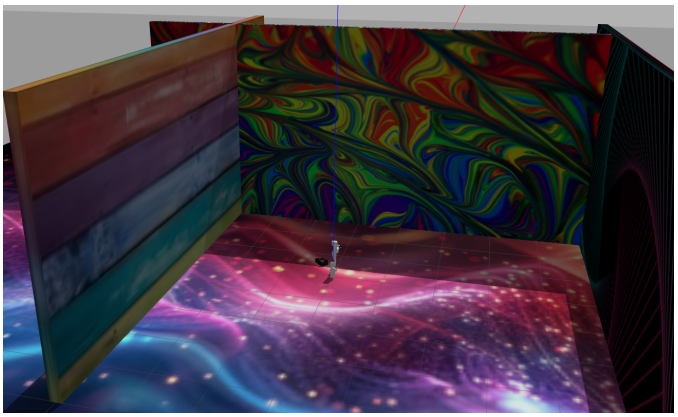


Fig. 1: Simulation world with walls and different textures

### B. Behavior Cloning

Our behavior cloning approach employs a custom neural network architecture to learn manipulation policies from expert demonstrations. The network is named after the approach proposed by Hermann et al. in [1], namely ACGD (Adaptive Curriculum Generation from Gripper Demonstrations). It processes both visual and proprioceptive information to predict the appropriate end-effector movements. For this purpose, we utilized the policy architecture represented in Fig.3, proposed by Hermann et al. in [1].

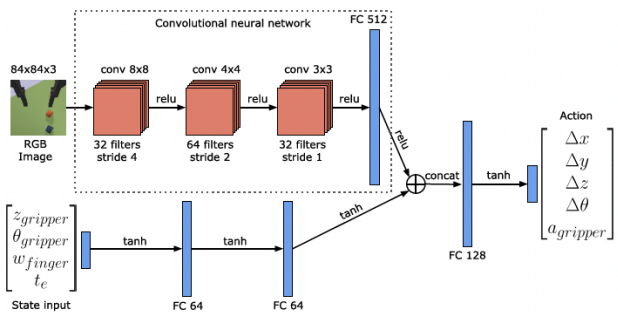
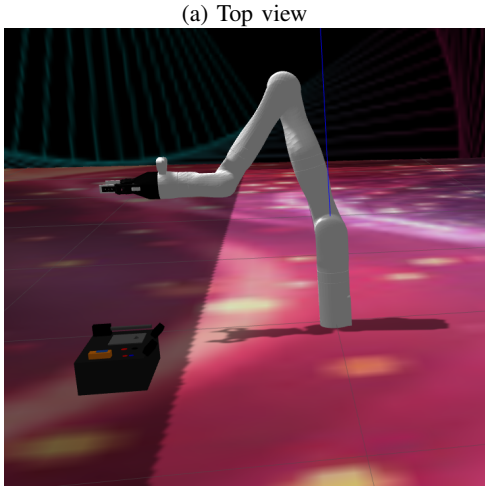


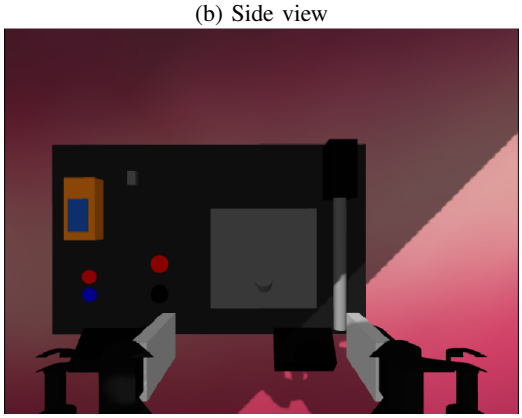

Fig. 3: Architecture of the policy network [1]

[1]ros_kortex GitHub
[2]Task board description GitHub

(a) Top view



(b) Side view



(c) Arm camera view in simulation when end-effector is on top of the task board

Fig. 2: Simulation setup views of Kinova Gen3 7-DOF arm and task board

*1) Network Architecture:* The ACGD network consists of three main modules: an image processing CNN, a state processing network, and a combined action prediction network.

*a) Image Processing Module:* The image processing module processes RGB images of size $84 \times 84 \times 3$ using a CNN architecture:

- Conv1: 32 filters of size $8 \times 8$ with stride 4
- Conv2: 64 filters of size $4 \times 4$ with stride 2

- Conv3: 32 filters of size $3 \times 3$ with stride 1 (All CNN layers are followed by ReLU activations)
- Fully connected layer: Maps the flattened feature map (size $32 \times 7 \times 7 = 1568$) to a 512-dimensional feature vector
- Dropout with probability 0.5 to prevent overfitting

*b) State Processing Module:* The state processing module processes the 8-dimensional robot state vector containing:

- End-effector Cartesian position $(x, y, z)$
- End-effector orientation (quaternion representation: $\theta_x, \theta_y, \theta_z, \theta_w$)
- Normalized remaining episode time

This module consists of:

- First fully connected layer: Maps 8-dimensional input to 64-dimensional hidden layer, followed by tanh activation
- Dropout with probability 0.5
- Second fully connected layer: Maps 64-dimensional input to 64-dimensional output, followed by tanh activation

*c) Action Prediction Module:* The action prediction module fuses the embeddings for the processed image and state information to predict the robot's next action:

- Concatenation layer: Combines the 512-dimensional image features with the 64-dimensional state features
- Fully connected layer: Maps the combined 576-dimensional vector to a 128-dimensional hidden layer, followed by tanh activation
- Dropout with probability 0.5
- Output layer: Maps 128-dimensional hidden layer to 4-dimensional action vector representing $(\Delta x, \Delta y, \Delta z, \Delta\theta_w)$

*2) Experimental setup:* The network was trained using demonstrations collected through expert operators in the form of ROS bags, by manipulating the Kinova Gen3 7-DOF arm to locate and track a task board. Each training sample consists of:

- Input: Input state information, which includes RGB image of the current state and respective end-effector recordings.
- Target: Action vector representing end-effector movement $(\Delta x, \Delta y, \Delta z, \Delta\theta_w)$

We used the SmoothL1Loss (Huber loss) as our objective function, which provides robustness against outliers while maintaining stability for small error values. The main drive for using this loss and dropout layers was to avoid the vanishing gradients problem in minor/no the input state value changes. The model was optimized using Adam optimizer with different training configurations (i.e., learning rates, batch sizes, epochs, etc.). Results are depicted in Fig.4 and Fig.5. We found that the learning rate of $5 \times 10^{-7}$ and batch size of 1 were the best match in our case. To prevent overfitting, we applied dropout with a rate of 0.5 after each fully connected layer. The dataset was split into 80% training and 20% validation sets.

The training process utilized TensorBoard for visualizing loss metrics and model performance. Checkpoints were saved periodically to capture model states throughout the training process, allowing for the selection of optimal model parameters based on validation performance.

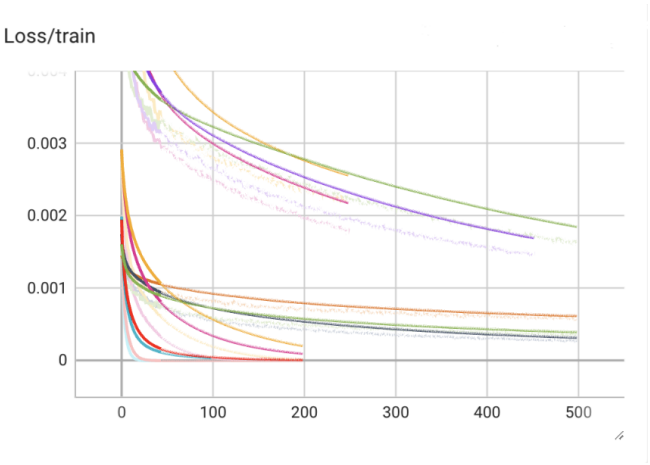



Fig. 4: Training loss curves for different experiments

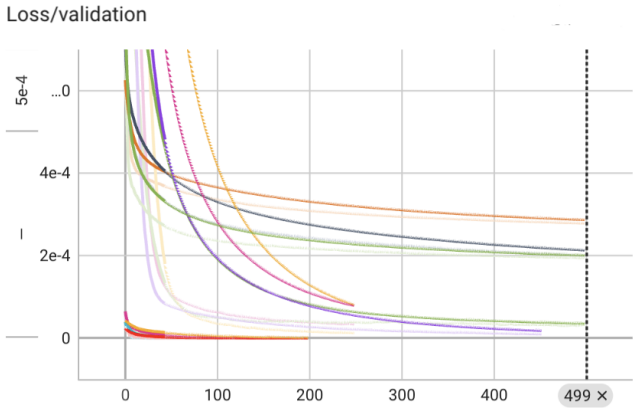



Fig. 5: Validation loss curves for different experiments

### C. PPO for reinforcement learning in simulation

This approach for training the Kinova Gen3 7-DOF arm utilizes reinforcement learning (RL), specifically the Proximal Policy Optimization (PPO) algorithm. PPO is an on-policy, model-free RL algorithm that effectively balances exploration and exploitation while maintaining stability throughout training. Its stability makes it particularly well-suited for robotic control tasks, where smooth and reliable learning is crucial for real-world applications. A custom environment was implemented to utilize the PPO implementation of Stable-Baselines3 [2]. The system architecture integrates ROS for robot control and Gazebo for physics simulation, with the environment following the Gymnasium interface standards. A demonstration of the training process in simulation can be viewed at [3].

*1) Environment Design:* A custom Gymnasium [4] environment (`KinovaEnv`) was developed to interface with Robot Operating System (ROS) Noetic [5] and Gazebo [6] for controlling and simulating the robotic arm. The environment provides:

*a) Observation Space:*

- Visual Input: 84×84×3 RGB images from the robot's camera
- Proprioceptive State: 7-dimensional vector containing:
  - End effector position $(x, y, z)$

  - End effector orientation $(\theta_x, \theta_y, \theta_z)$
  - Timestep value

*b) Action Space:* A continuous 5-dimensional space normalized to $[-1, 1]$:

- Translational movements - Changes in position along the $x$, $y$, and $z$ axes, namely $(\Delta x, \Delta y, \Delta z)$
- Rotational movement - Changes in the robot's orientation along the $z$ axis, namely $(\Delta\theta_z)$
- Gripper control - A value indicating whether the gripper should open (0.0) or close (1.0)

*2) Policy and Value Network Implementation:* The PPO algorithm utilizes a dual-network architecture consisting of a policy network (actor) and a value network (critic), both sharing the same feature extractor but with separate output heads. The policy and value network architecture was implemented as detailed by Hermann et al. in [1] and can be visualized in Fig.3.

*a) Feature Extractor:* The shared feature extractor is implemented as per interface of custom environemnt in Gymnasium [4], it processes both visual and proprioceptive inputs:

- Processes 84×84×3 RGB images through CNN layers
- Handles 7-dimensional state vectors through fully connected layers
- Outputs a 128-dimensional feature vector representing the combined state

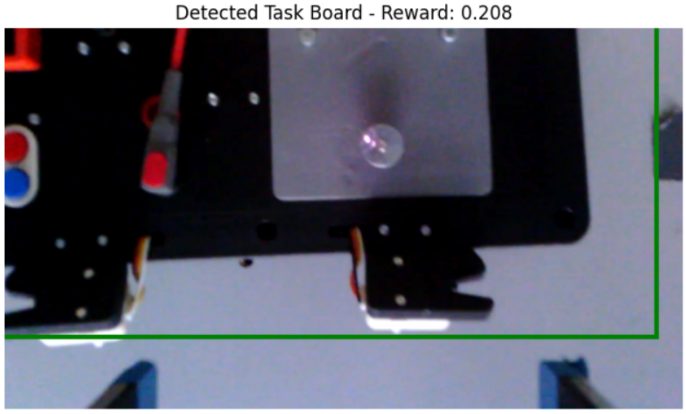
A reward function is designed to encourage the agent to focus on key aspects like visibility of the task board and smoothness of its actions, as described in the next sub-section.

Initially, the environment is set up by initializing ROS services and topics to allow communication with the robotic arm and to provide sensory feedback, such as camera images and the arm's state. The agent then executes actions, collects observations of the state, and receives rewards based on the outcomes of those actions. It computes the advantage, which quantifies how much better an action was compared to the expected reward, using Generalized Advantage Estimation (GAE). PPO optimizes a surrogate objective function, which includes a clipping mechanism to prevent drastic policy updates that could destabilize learning while encouraging steady improvements. The policy is updated using gradient descent to maximize this objective function, leading to gradual improvements in the agent's behavior. The agent's performance is periodically evaluated to monitor progress, with the best-performing models being saved for future use.

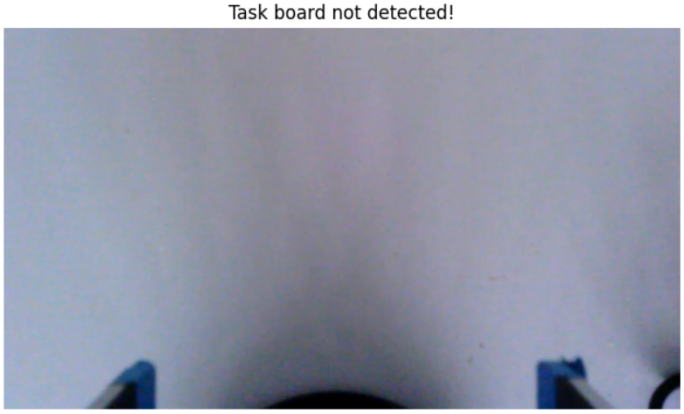
### *D. Reward estimation*

The visibility of the task board is crucial for the manipulator to perform any action, making it a key component in the reward estimation process. Prior to training the RL agent, the most desired view of the task board is saved as a template image. At each timestep, the task board is detected in the input image using computer vision techniques, including grayscale conversion, adaptive thresholding, and contour detection. A bounding box is then computed around the largest detected contour. The bounding box provides a region of interest where the task board is located. ORB (Oriented FAST and Rotated BRIEF) keypoints and descriptors are detected in both the input image and the task board template. The match score is based on the number of good keypoint matches, with a higher score indicating a better alignment between the detected and template task boards (Fig. 6).


(a) Task board fully visible




(b) Task board partially visible




(c) Task board not visible

Fig. 6: Visual representations of task board detection and match scoring.

The reward function encourages the agent to improve its performance in detecting a task board through four key components:

- Visibility Reward: Based on the match score between the current image and the task board template.
- Match Score Improvement: The agent is rewarded for improving its match score over time.
- Action Smoothness: Large action changes are penalized to encourage smooth movement.
- Energy Efficiency: Larger actions incur penalties to promote energy-efficient behavior.

The final reward is a weighted sum of these components, with additional bonuses for high match scores (>0.8) and penalties for low match scores (<0.1), determining when the task is considered successful or failed.

## IV. Evaluation

We evaluated the model's ability to predict robot arm movements based on visual and state inputs. We utilized a trained policy model and evaluated it on randomly selected samples from the test dataset. We could realize that the pre-trained policy network can perform relatively well, especially when there is a change in the input state. In this case, the model could acceptably predict the output action values. We noticed inconsistencies in the model predictions when there is no change in the input state ( e.g., multiple consequent states with input image being freezed and no change in the joint measurements). In another word, the model was not learnt to predict no output actions in freezing state. Some relevant images are presented in Fig.7

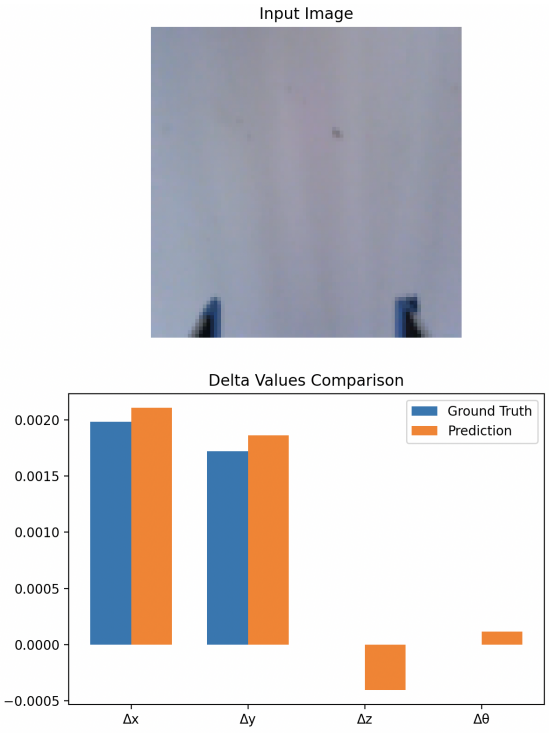
## V. Conclusions

### A. Summary

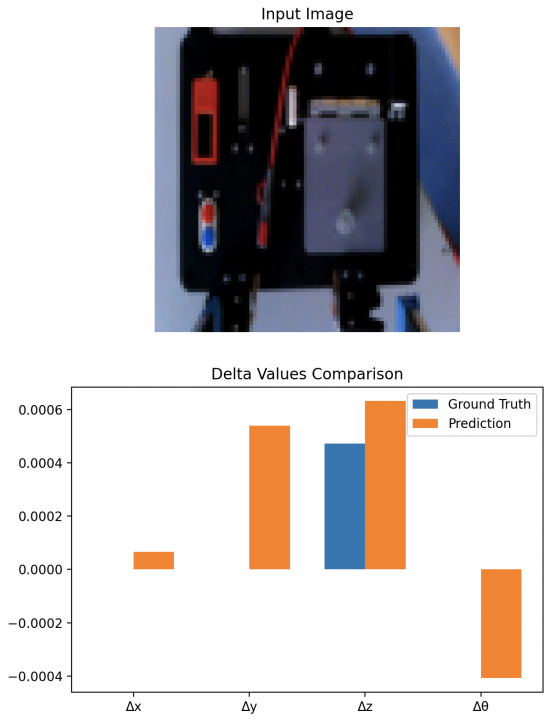
In this project, we developed a learning-based system to allow a Kinova Gen3 7-DOF robotic arm to visually track and interact with a task board. The approach combines both camera images and end-effector state information to guide the robot's movements. We first trained the model using expert demonstrations (behavior cloning), then refined it using reinforcement learning with the PPO algorithm. The core idea was to teach the robot to learn meaningful movements based on visual input and proprioception, while rewarding it for keeping the task board in view, moving smoothly and using minimal effort.

### B. Contributions

Our main contribution is a two-phase learning pipeline that brings together behavior cloning and reinforcement learning for visual-based manipulation. We designed a custom neural network based on the work of Hermann et al. [1] that processes both images and state information to predict actions. Additionally, we created a Gazebo training environment that includes the Kinova robot and a task board setup and developed a reward function that helps the robot improve its visual tracking over time. We also built a custom ROS-based interface, KinovaEnv, to support training and control. Overall, the system shows promising responses to changing inputs and lays the foundation for more flexible manipulation tasks.




(a)




(b) Task board not visible

Fig. 7: Policy model predictions wrt. to input state changes for a) Task board not being visible b) Task board visible states.

### C. Future Work

Future work includes improving response to static inputs, replacing handcrafted rewards with learned visual similarity models, and expanding to the next tasks, such as the slider and the change of probe on the task board.

## REFERENCES


[1] L. Hermann, M. Argus, A. Eitel, A. Amiranashvili, W. Burgard, and T. Brox, "Adaptive curriculum generation from demonstrations for sim-to-real visuomotor control," *CoRR*, vol. abs/1910.07972, 2019. [Online]. Available: http://arxiv.org/abs/1910.07972

[2] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," https://github.com/DLR-RM/stable-baselines3, 2021.

[3] Chaurasia Arunima, "Kinova arm ppo training in gazebo simulation," Online, 2025, available: https://youtu.be/1NrH1sK3ZCE.

[4] F. Foundation, "Gymnasium: A toolkit for developing and comparing reinforcement learning agents," https://github.com/Farama-Foundation/Gymnasium, 2023.

[5] Open Source Robotics Foundation, "Robot operating system (ros) noetic ninjemys," https://wiki.ros.org/noetic, 2020.

[6] ——, "Gazebo 11: Robot simulation for ros noetic," https://gazebosim.org, 2020.