Manipulator control and training based on hand gestures detection

Chen Yifei, Yang Deyu, Zhu Siying, Ning Chenhui, Fang Yijun

***Abstract*—The development of robotic arms has gained significant momentum in recent years, revolutionizing various sectors such as industry, medicine, and daily life. Among the advancements, the integration of gesture recognition technology with robotic arms stands out as a cutting-edge approach. By combining robotic arms with computer vision technology, this advanced system captures and analyzes the position, posture, and movements of human gestures using visual devices like cameras, depth sensors, and infrared sensors. Employing computer vision algorithms, the system accurately classifies and recognizes gestures by leveraging machine learning and pattern recognition techniques. Based on these recognition results, the robotic arm mimics and tracks the user's hand movements by precisely controlling each joint's motion using only average 83.54s, which proves to be efficiently compares to other control methods. We also use this control methods to collect data for mimic learning on training manipulator. This hand motion tracking technology finds applications in numerous fields, including surgical assistance in medicine and enhancing interactive experiences in virtual reality and gaming.**

I. Introduction

In recent years, the development of robotic arms has become increasingly widespread. The application of robotic arms has made great contributions to the industrial field, medical field, and daily life. Among them, the robot arm with gesture recognition technology is an advanced robot technology. It combines robotic arms and computer vision technology, capturing the position, posture, and movements of human gestures through visual devices such as cameras, depth sensors, and infrared sensors. Through the processing of computer vision algorithms, the system can classify and recognize gestures. These algorithms utilize machine learning and pattern recognition techniques to train machines to recognize various gesture actions and associate them with specific tasks or commands. Based on the recognition results, the robotic arm simulates and tracks the user's hand movements by controlling the movements of each joint. The robotic arm technology that tracks user hand movements has wide applications in many fields. For example, in the medical field, it can serve as a surgical aid to provide better support for patients and medical staff. In the fields of virtual reality and gaming, hand motion tracking technology can make interactive experiences more realistic.

We now propose a pose control system for desktop robotic arms. This system allows users to use natural hands to control robotic arm posture for daily assistance needs. The goal of the project is to develop a desktop robotic arm that can recognize user's hand posture and reproduce movements. The key challenge is to accurately convert user's expected movements into robotic arm movements. The robot arm can be precisely controlled by position control, but this method needs to consider the solution of the forward kinematics and inverse kinematics of the human arm. Therefore, after reviewing the research of Li Fei-Fei's group [1], we decided to consult their research and adopt a simpler speed control method as the control method for this robotic arm. When the user's hand movement is detected, the corresponding commends are generated, speed and acceleration information is directly transmitted to the robotic arm. This system has several advantages, including ease of use, flexibility, and the ability to train the robotic arm through imitation learning, whereby the arm can learn from human demonstrations.

This project is divided into two main objectives. Firstly, to realize gesture recognition control. Tracking user hand movements faces some challenges, such as the demand for high-performance calculations and algorithms for accurate recognition and real-time tracking of complex gestures. Also, the structural design of the robotic arm affects the smoothness of the movements. Secondly, through imitative learning training, the robot arm is expected to perform tasks independently, which will make task execution more efficient and accurate. Imitative learning reduces the need for complex programming and improves the accuracy and naturalness of motion. The quality and quantity of training samples have a particularly important impact on training effectiveness, therefore it is necessary to ensure the diversity and representativeness of the samples.

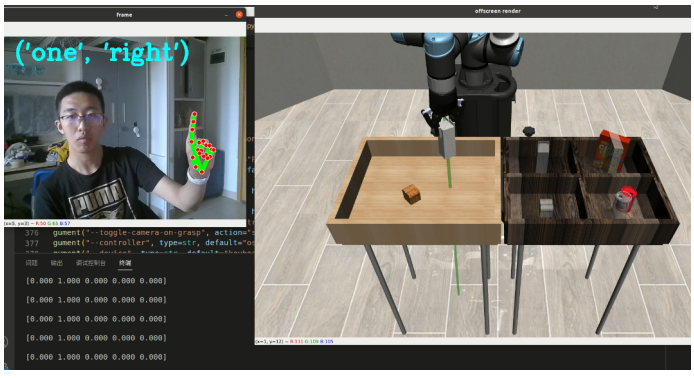

Fig. 1. Our work: Gesture and pose control of manipulator in simulation.

II. Related Work

Imitation learning and offline reinforcement learning have been applied in several fields, such as playing table tennis, Go and video games. Imitation learning can learn the task strategy of the task (for example, BC-Behavioral Cloning) or solve the RL by inferring an underlying reward function [2]. However, both BC and IRL need to demonstrate the same short-sighted manipulation task multiple times, and are affected when attempting to mimic the long-sighted activity, even though they are consisted of the demonstrated short-sighted skills. In our experiments, the core idea of the IRIS algorithm is to combine behavioral cloning and incremental reinforcement learning for offline robot operation learning through the interaction of planner and policy.

A. Learning Task-Oriented Grasping

Past research has explored ways to use deep learning [2][3][4] techniques to learn robotic grasping strategies. These methods often rely on data collected from the real world or on simulations via object models [4].This paper proposes a method based on simulation self-supervision for learning grasping strategies in task-oriented tool operations. This method achieves the ability to learn grasping strategies from large-scale simulated data by combining simulated environments and deep learning techniques. There are two main components: a grab evaluator and a grab generator. The grasping evaluator predicts the success probability of grasping by inputting the image and the tool pose. The grab generator uses this evaluator to generate grab poses with a high probability of success. Through self-supervised learning on large-scale simulated data, the robot can learn grasping strategies adapted to different tasks and tools. The paper also proves experimentally that the proposed method shows good performance and migration ability in the tool operation tasks in the real world.

B. Random Cropping Ensemble Neural Network

In the robot arm grasping system, image classification is a key task, which can help the robot to quickly and accurately identify different objects and make corresponding grasping decisions [5]. Traditional image classification methods often rely on manually designed feature extractors and classifiers[6]. These methods require knowledge and experience from domain experts, and performance in complex scenarios may be limited. In contrast to these methods, our method uses simulated self-supervised learning, trained using large-scale simulated data, to learn more general grasping strategies in different task and tool contexts. Compared to traditional methods, our method exploits the powerful representation learning ability of neural networks, which is able to automatically learn features from the data and classify them. Deep learning methods have achieved remarkable success in image classification tasks. Convolutional neural network (CNN) is a commonly used deep learning model, which can learn a hierarchical feature representation from images. Our method is also based on deep learning but introduces the idea of random tailoring ensemble to further improve the classification performance. Our method borrows the idea of ensemble learning by introducing random cropping ensemble in the image classification task. Multiple image fragments with different sizes and positions were generated by randomly tailoring the input images and fed into multiple neural network models for classification. Finally, the prediction results from multiple models are integrated to obtain more accurate classification results.

C. RoboTurk

The article[7] provides 6 degrees of freedom intuitive motion control, the mobile phone movement mapping to the robot arm movement. Users can watch the autonomous robot arm try to solve the task and help, if necessary, to help the robot learn from errors. RoboTurk You can host multiple simultaneous users, each controlling a robot arm in their own workspace, and multiple users in a shared workspace, to demonstrate collaboration and adversarial tasks. Meanwhile the latency is very low and can control simulated and physical robotic arms from around the world through real-time robots. This has been stress-tested by controlling the Stanford robotic arm from distant places such as China and India.

In our experiment, we used MediaPipe in gesture recognition part, through MediaPipe recognition to gesture, and then the gesture binding to our control code, so as to achieve a gesture to control a movement, in the demo, they used the keyboard control, and in stanford roboturk, they used the phone for mechanical arm control. We trained our data with several methods: BC, HBC and IRIS.

III. Data

For the simulation and datasets, factors like accuracy, real-time performance and the cost-effectiveness and flexibility detemind why we choose to use these rather than others. What's more, the availability integration, comprehensive documentation, community support, and prior experience with these tools also influenced our decision.

Our project relies on utilizing data from the Mediapipe library for hand pose and gesture detection. Mediapipe provides a robust and efficient solution for real-time hand tracking within the simulation environment. By detecting the coordinates of human pose and hand, we are able to extract essential information for recognizing and classifying various hand gestures. All these coordinates data can be obtained by access to **Landmarks of pose**, which is in world coordinate system.

Regarding the simulation environment, we opted to utilize Robosuite as our platform of choice. Robosuite offers a comprehensive set of tools, control access, and manipulator models necessary for our research objectives. By leveraging Robosuite, we could focus on developing and implementing the hand gesture-based control system without the need for physical robots. This allowed us to iterate and experiment more rapidly while significantly reducing costs and logistical challenges associated with physical hardware. In addition to hand gesture detection and control, an essential aspect

Fig. 2. Hand detection key points

of our project involves leveraging RoboMimic to learn the demonstration of gesture control and enable the manipulator to perform tasks autonomously. The utilization of RoboMimic's imitation learning capabilities further demonstrates our application of the ideas and skills acquired during the quarter. By leveraging the collected demonstration data and employing different imitation learning algorithms, we have effectively trained the manipulator to perform tasks autonomously based on the gestures detected by our system. Since robosuite and robomimic is development by a same lab, the dataset that records simulation data can be converted to each other in very simple maner. In fact, their structure is actually the same.

- data (group)
  - date (attribute) - date of collection
  - time (attribute) - time of collection
  - repository_version (attribute) - repository version used during collection
  - env (attribute) - environment name on which demos were collected
  - demo1 (group) - group for the first demonstration (every demonstration has a group)
    - model_file (attribute) - the xml string corresponding to the MJCF mujoco model
    - states (dataset) - flattened mujoco states, ordered by time
    - actions (dataset) - environment actions, ordered by time
  - demo2 (group) - group for the second demonstration

    ...

    (and so on)

Fig. 3. Every set of demonstrations is collected as a demo.hdf5 file. The reason for storing mujoco states instead of raw observations is to make it easy to retrieve different kinds of observations in a postprocessing step. This also saves disk space (image datasets are much larger).

Therefore, We run our pose controller to demonstrat how to finish the task and collect them into a *.hdf5 file. Then convert the dataset to low dimension dataset that can be accessed by robomimic.

By selecting this approach, we have effectively applied the ideas and skills developed during our quarter of study to address the problem of manipulator control and training based on hand gesture detection. We have leveraged state-of-the-art libraries and simulation environments, enabling us to develop an efficient and practical solution.

IV. Methods

Throughout the quarter, we acquired essential knowledge and skills related to gesture recognition, control systems, and simulation environments. By applying these ideas and skills, we have successfully developed and implemented the gesture control system for the manipulator To provide a visual representation of our methodology, Figure 1 illustrates an overview of the gesture control system. It demonstrates the flow of information from hand gesture detection to gesture classification, command mapping, and manipulator control. Through the combination of Mediapipe, Robosuite, and our expertise in control systems and simulation environments, we have effectively tackled the problem of gesture control of the manipulator. The application of our ideas and skills developed during the quarter has enabled the creation of an accurate, intuitive, and cost-effective solution.

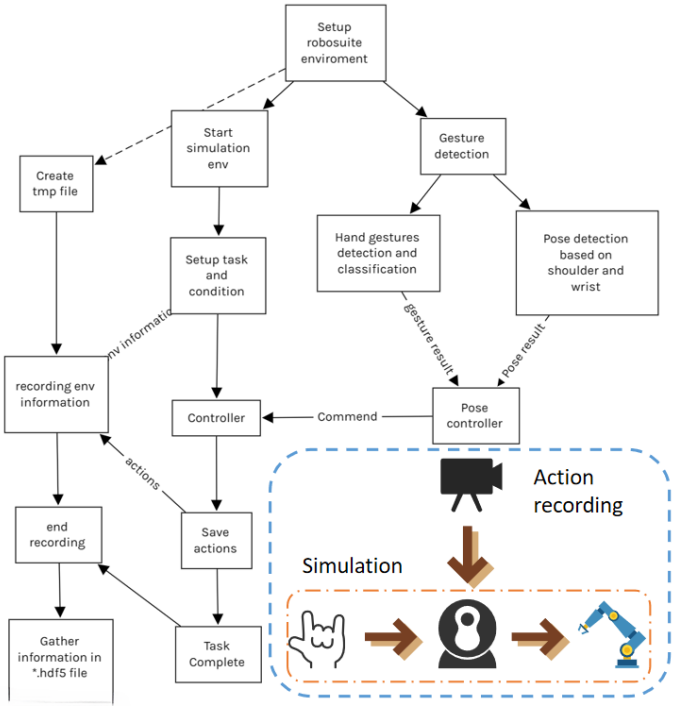

Fig. 4. Overall description of the Gesture Control System. Including simulation setting, gesture detection and action recording

In addition, we conducted machine learning training on the datasets collected after the robotic arm completed the specified task in the simulation environment. By using different machine learning algorithms, we have gotten models that can initially implement specific grasping task.

A. Gesture Control

1) Approach to Problem Solving: Our approach to gesture control of the manipulator consists of several key components that work together seamlessly to achieve precise and intuitive control. These components include hand gesture detection, gesture classification, gesture-to-command mapping, and manipulator motion control.

- **Hand Gesture Detection using Mediapipe:** We employ the Mediapipe library for hand pose estimation and gesture detection. This powerful tool accurately tracks

TABLE I
GESTURES CONTROLS COMMAND

Gestures	Command
Five-Move in four directions	Move arm horizontally in x-y plane
Two-Move up and down	Rotate arm about x-axis
Four-Move up and down	Rotate arm about y-axis
Six-Move up and down	Rotate arm about z-axis
Good	Down
Stone	Up
SpaceBar	Gripper(Open/Close)
q	Reset
ESC	Quite

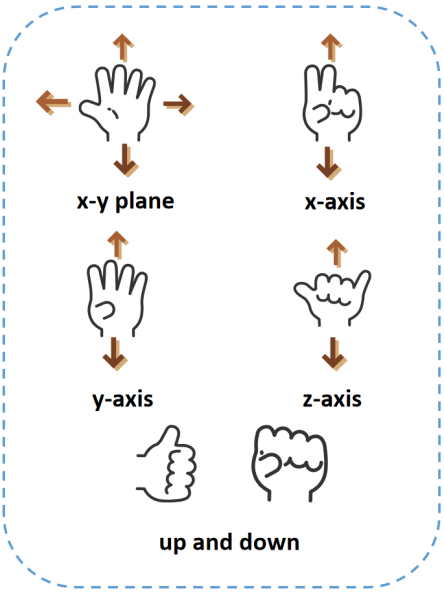

Fig. 5. Description of gesture control method with icons. Users can simply follow these actions to control a manipulator in simulation environment. They can also change gestures according to their habits.

the coordinates of the hand and its key landmarks in real-time. By leveraging the robust hand tracking capabilities of Mediapipe, we can capture the intricate movements and positions of the hand, enabling effective gesture recognition.

- **Gesture Classification:** Once the hand landmarks are detected, we employ a gesture classification algorithm to analyze the finger positions and movements. By considering the relative positions and angles of the fingers, we can classify various hand gestures, such as "Five," "Good," "Stone," "Two," "Four," and "Six." This classification process is crucial for accurately interpreting the user's intended gestures.
- **Gesture-to-Command Mapping:** After classifying the gestures, we establish a mapping between the recognized gestures and the corresponding manipulator commands. This mapping enables the translation of gestures into precise control instructions for the manipulator. For example, the "Five" gesture can be mapped to control the manipulator's movement in the x-y plane, while "Good" and "Stone" gestures can be associated with vertical motion (up and down). Similarly, "Two," "Four," and "Six" gestures can be mapped to control rotational movements around the x-axis, y-axis, and z-axis, respectively.
- **Manipulator Motion Control:** To translate the recognized gestures into manipulator movements, we utilize appropriate control techniques such as inverse kinematics (IK) or operation space control. The first one is move relative to the global coordinate frame. The second one is relative to the local robot end effector frame. These techniques enable us to calculate the required joint angles or operational space coordinates for the manipulator to execute the desired actions accurately. By integrating the gesture control system with the manipulator's controller, we achieve real-time and precise motion control based on the recognized gestures.
- **Gesture Adjustment and Real-time Feedback:** We provide users with the ability to adjust the gestures according to their preferences and habits. This customization allows for a personalized interaction with the manipulator. Furthermore, real-time visual is incorporated to provide users with immediate awareness of the manipulator's response to their gestures, enhancing the overall user experience.

By combining these components and features, our approach offers an accurate, intuitive, and interactive gesture control system for the manipulator. Users can effortlessly communicate their intentions to the manipulator through hand gestures, enabling a natural and user-friendly control interface.

B. Training from human demonstration

There are three steps to carry out robot arm training:

1) Data collection and processing: Datasets are constructed by observing and recording the expert-demonstrated state and action data. We have completed data collection in the process of gesture control of the robot arm and implemented post-process for the datasets to guarantee compatibility with robomimic.

2) Model selection and training: The agent(robot arm) learn experience knowledge from the obtained trajectories rather than interact with the environment, so we choose imitative learning and offline reinforcement learning as the models. We employed three different algorithms to train the dataset that we processed in the first step. By comparing these methods, we can analyze the most suitable training approach and identify areas for improvement in the dataset.

For imitation learning, we utilize Vanilla Behavioral Cloning(BC) algorithm and Hierarchical Behavioral Cloning(HBC[7]) algorithm to implement training process. As for this BC algorithm, we use a simple supervised regression from observations to actions, along with some variants such as stochastic GMM policy and stochastic VAE policy. Unlike traditional BC, HBC focuses on learning and copying the behavior of experts on multiple levels. For offline reinforcement learning, we utilize Implicit Reinforcement without Interaction at Scale(IRIS[8]) algorithm to implement training process. The planner is responsible for learning an

initial strategy from the offline data, and then takes its output as the goal of the strategy for further incremental augmented learning.

3) Model evaluation and analysis: The effect of completing tasks of the robot arm will be reflected in validation losses: The model parameters will be updated every time we conduct training, and the updated model will be used for calculating validation loss. By observing and analyzing the variation of validation losses with the number of training sessions, the model can be evaluated.

V. Experiments

A. Experiments Setting

In our experiments, we focused on evaluating the effectiveness and efficiency of our gesture control system using five simulated tasks: Block Lifting (Lift), Block Stacking (Stack), Wiping (Wipe), Pick Place items (PickPlace), and Nut-and-peg Assembly (Assembly), as illustrated in Fig.6.

1) Block Lifting (Lift): This task serves as a diagnostic example and involves lifting a cube using the UR5e robot arm. It provides a simple scenario to evaluate the basic functionality of the gesture control system.
2) Block Stacking (Stack): In this task, the goal is to stack blocks on top of each other, requiring precise manipulation and coordination of the manipulator's movements.
3) Wiping (Wipe): The wiping task involves moving the manipulator in a wiping motion, simulating cleaning or wiping actions. This task tests the system's ability to perform continuous and controlled motions.
4) Pick Place items (PickPlace): In this task, the manipulator is tasked with picking objects from one location and placing them in another. It assesses the system's capability to perform accurate object grasping and placement.
5) Nut-and-peg Assembly (Assembly): This task represents a more complex and challenging scenario, where the manipulator needs to assemble nuts onto pegs. It tests the system's ability to handle intricate and precise manipulation tasks.

To evaluate the effectiveness and performance of our gesture control system, we conducted a user study involving 10 students aged 20-25. Each participant provided 20 demonstrations on the block lifting task and 5 demonstrations on each of the other tasks. This user study allowed us to collect a diverse range of gesture data and assess the system's robustness across different tasks. To compare the efficiency of our controller, we compared the number of actions taken with official professional datasets and datasets collected using other interfaces. This analysis provides insights into the efficiency and effectiveness of our gesture control system compared to alternative approaches. Additionally, we compared the completion time and maneuverability of our system with RoboTurk phone interface and other interfaces. This evaluation allowed us to assess the system's performance in terms of task completion speed and the user's ability to maneuver the manipulator accurately and efficiently using the gesture control system.

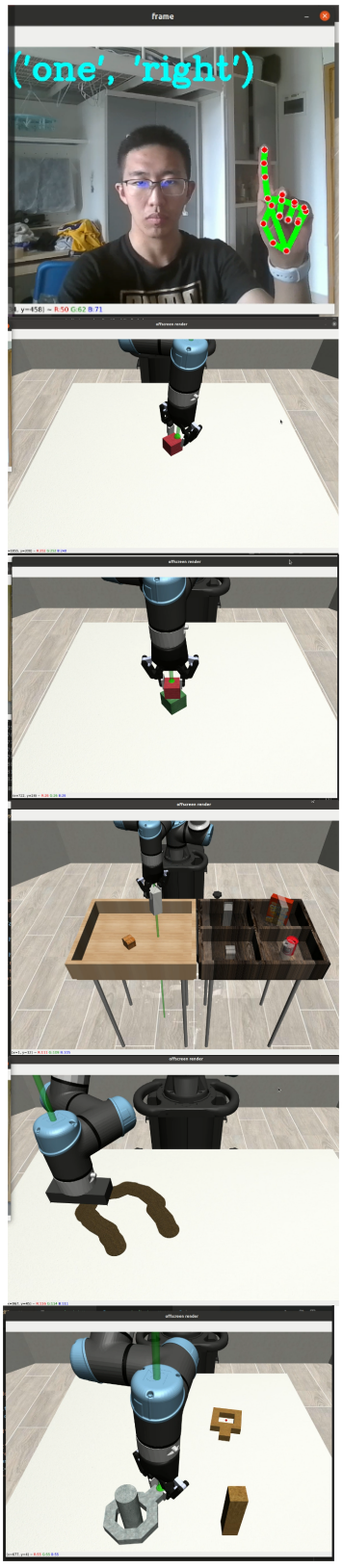

Fig. 6. Five Tasks we test in our experiments. Lifting: Lift a box. Stacking: Stack one small box on another big box. Pick and Place: Pick four different real world item to there own place. Wipe: wipe the dirty things on the table. Nut Assemble: Put nut on peg.

By conducting these experiments and evaluations, we aim to demonstrate the efficacy of our gesture control system across various tasks, compare its efficiency with alternative approaches, and assess its performance in terms of completion time and maneuverability. These findings provide valuable insights into the capabilities and limitations of our system and contribute to its further refinement and development.

To develop an intelligent robot arm, we used the datasets which collected by ourselves to train a model that can complete the "lift" task. The datasets contain 100 successfully captured trajectories with certain differences in task completion effect. As we proposed before, we will use total two algorithms, BC and HBC, to train the collected datasets. By comparing the grasping effect and training numbers of the three algorithms, we hope to find an algorithm suitable for the training set and get a model that can quickly complete the task. Since these algorithms can reuse trajectories in the datasets, we stop training when the loss reaches a stable value rather than setting an best loss value.

In this experiment, we firstly trained a model that could perform the same task "lift" using the datasets provided by the robomimic official website and the three algorithms mentioned before. Then, the trained models were compared with the models trained with the collected datasets. We ploted the validation losses from the robomimic datasets trained with each algorithm. After horizontal comparison, We ploted each models success rates after convergence for vertical comparison. Based on the comparison results, methods to optimize the training results are found from datasets, training algorithms and other aspects.

B. results

1) Experiment 1: Comparison of Number of Actions: According to the data structure, we can collect number of actions that are needed for task completion. Here we compare our average number of actions with official datasets, which was collected with operator using RoboTurk platform and machine-generated actions. There is also distinguish difference between professional operator and normal operator, so we classify them into two categories: Proficient-Human, Multi-Human that include different level of operators. Our dataset is consist of 500 demonstrations that recorded after 10mins practice. Proficient-Human (PH): The average number of actions required for task completion was 3. Multi-Human (MH): The average number of actions required for task completion was 12. Machine-Generated (MG): The average number of actions required for task completion was 8. Pose-demonstrations (our work): Our Pose-demonstrations approach achieved a significantly lower average number of actions, with only 5 actions required for task completion. These results demonstrate that our Pose-demonstrations approach outperforms the other datasets in terms of efficiency and streamlined task completion.

2) Experiment 2: Comparison of Interfaces: We evaluated our Pose-Control interface by comparing it with other control interfaces such as Keyboard, 3D Mouse, VR Controller, and Phone. The Kolmogorov-Smirnov statistics were used to measure the difference between completion times in the Lifting task across different interfaces, followed by associated p-values. The results are presented in the table1.

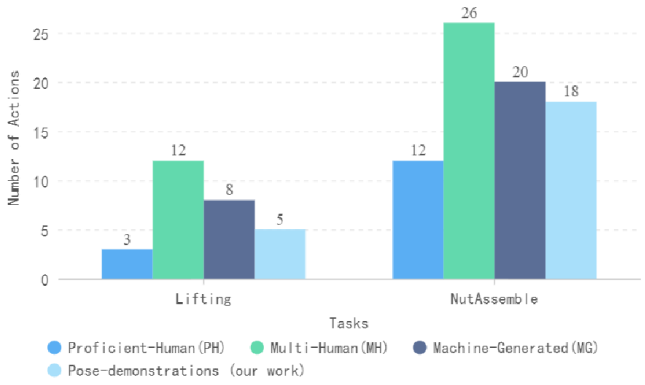

Fig. 7. Comparison of number of actions: For two tasks, Lifting and NutAssemable, we compare four datasets. The proficient-Human(PH), Multi-Human(MH) and machine-Generate are from the official datasets. The last one is our collection from pose-demonstrations. The less action it use, the better control it will be. Clearly, our dataset show good performance, just use more actions than the proficient operator.

Type	Keyboard	3D Mouse	VR Controller	Phone	Pose-control
Keyboard	-	0.575	0.9	0.725	0.545
3D Mouse	-	-	0.375	0.325	0.412
VR Controller	-	-	-	0.225	0.243
Phone	-	-	-	-	0.725
Pose-control(Ours)	-	-	-	-	-

Fig. 8. The Kolmogorov-Smirnov statistics were used to measure the difference between completion times in the Lifting task across different interfaces, followed by associated p-values. Our work is similar to the Phone and VR Controllers, which completion time is rather low in this group.

Based on our collection of completion times, we observe the following order in terms of increasing completion times: Phone $\approx$ VR Controller $\approx$ Pose-control $>$ 3D Mouse $>$ Keyboard. These results suggest that our Pose-control interface performs competitively compared to other interfaces, with faster completion times than Keyboard and 3D Mouse, and comparable performance to VR Controller and Phone. Our Pose-control interface provides a balance between intuitive control and efficient task completion.

3) Experiment 3: Comparison of models trained with collected datasets: To gain insights, we applied three distinct algorithms, BC, HBC, and IRIS, to train and analyze the data. The Fig.10 illustrates the models validation losses obtained from these algorithms. After training the collected datasets, we made a horizontal comparison of their validation losses: Upon closer examination, we noticed substantial fluctuations in the loss function values across all three algorithms. However, an encouraging trend emerged – the loss function converged

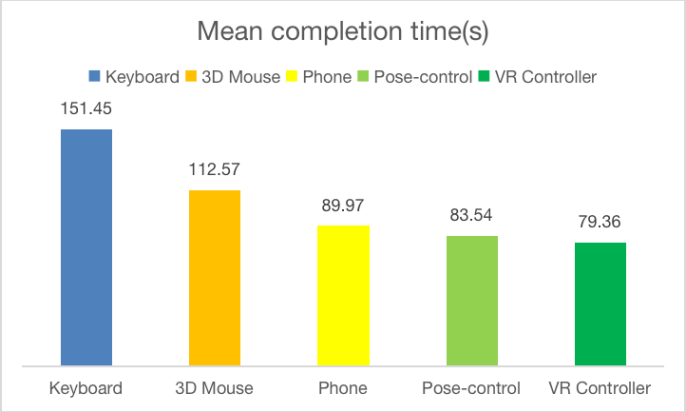

Fig. 9. Comparison of completion time of different interfaces: The figure shows the mean values of the completion time using different interfaces, which show the competence of each interface. The less time it use, the more adaptable the interface fit for people to use.

over time for each algorithm. Despite the initial volatility, the overall trajectory pointed towards convergence.

Delving deeper into the analysis, we compared the convergence speeds and stability of the algorithms based on the loss function values. Remarkably, the HBC algorithm outperformed the others in terms of both convergence speed and achieving a lower stable value for the loss function. This indicates that HBC exhibited faster learning and converged to a more optimal solution compared to BC and IRIS.

4) Experiment 4: Official datasets training results compared with ours: After training the robomimic datasets, we firstly made a horizontal comparison of their validation losses. The accompanying Fig.11 illustrates the loss function of the test results datasets obtained from three algorithms. Upon analyzing the results obtained from the three datasets in Imitation Learning, it can be found that with the increase of training times, except for the HBC algorithm, the losses of other algorithms on the whole present a trend of substantial and rapid decline , slow decline and convergence. On the other hand, HBC algorithm presents first rapidly decline, then slow decline to the lowest value, then slow rise, and convergence finally. As can be seen from the shape of the image, BC algorithm and IRIS algorithm show relatively stable performance and satisfactory convergence. BC algorithm tends to converge after 200 training sessions, while IRIS algorithm tends to converge after 100 training sessions. However, HBC algorithm reaches the lowest point after about 25 training sessions which means that the convergence speed of this algorithm is significantly higher than the other two algorithms.

Then We made statistics and comparison on the success rates of each algorithm after convergence, and divided each algorithm into two categories according to different datasets, respectively collected datasets and robomimic datasets. Fig.12 shows the comparison of success rates of each algorithm.

We can find that the success rates of all models trained using robomimic datasets are above 92.5%, which means that

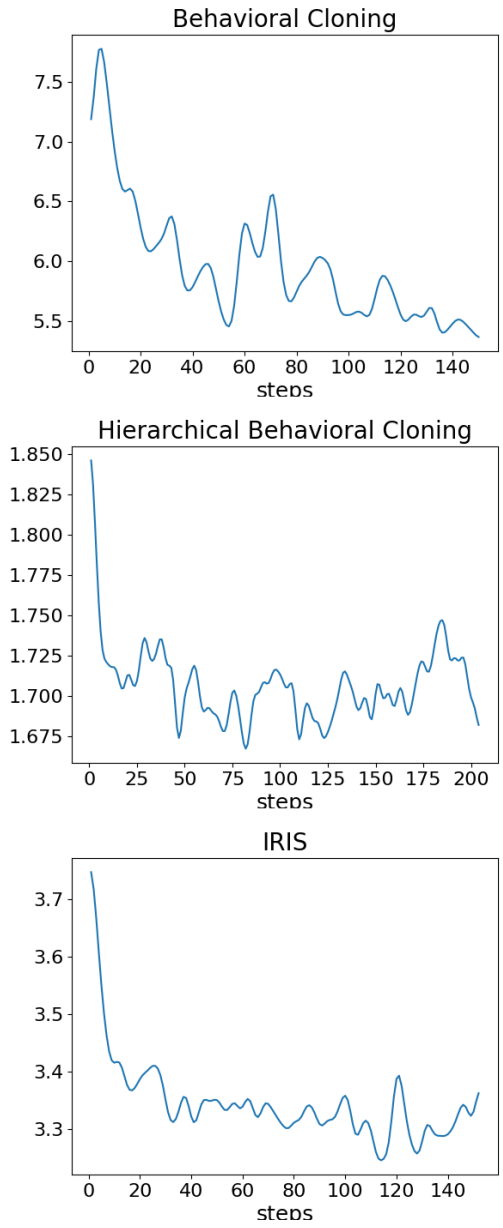

Fig. 10. Comparison of validation losses for models trained using the collected datasets: The figure shows the validation losses of models trained by different algorithms. From top to bottom are Behavioral Cloning, Hierarchical Behavioral Cloning, and Implicit Reinforcement without Interaction at Scale. The X-axis represents the number of training sessions

these models can complete the lift task well. Especially for the model obtained through the HBC algorithm, the success rate is as high as 97.5%. In contrast, the models trained with collected datasets have poor performance. Although the model trained under the HBC algorithm still has the highest success rate, the success rate is only 47.5% which even less than half of 97.5%. The model with the lowest success rate is trained using the BC algorithm, which only reaches 2.5%. This means that the BC method is completely unsuitable for the collected datasets. As the only offline reinforcement learning algorithm

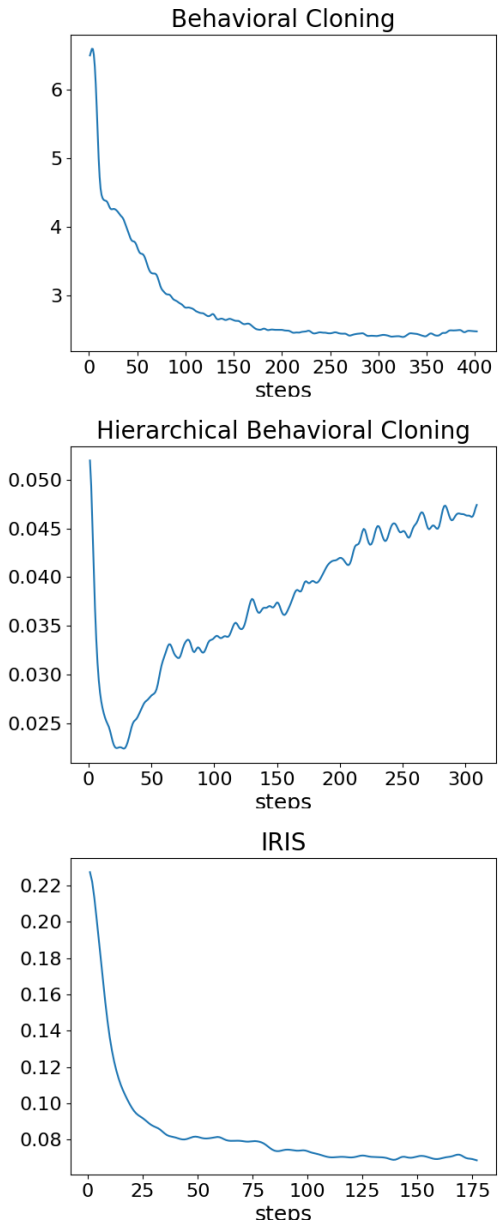

Fig. 11. Comparison of validation losses for models trained using the robomimic datasets. The algorithms used from top to bottom are Behavioral Cloning, Hierarchical Behavioral Cloning, and Implicit Reinforcement without Interaction at Scale. The X-axis represents the number of training sessions

in the testing algorithms, IRIS is significantly superior to BC algorithm. While the success rate of 37.5% is still slightly lower than HBC algorithm.

C. Evaluations

In our experiments, we compared five user interfaces for robot control: Keyboard, 3D Mouse, Virtual Reality (VR) Controller, Phone (RoboTurk), and Pose-control (Our work). Each interface offers unique advantages and trade-offs in terms of cost, accuracy, and ease of use.

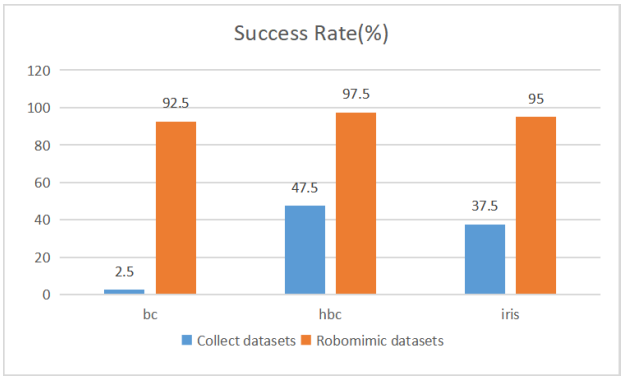

Fig. 12. Comparison of success rates of each algorithm: The figure shows the success rates of all algorithms after convergence. The higher rate it have, the better the success rate of the trained model to complete the task.

- Keyboard: The keyboard is the most widely available interface but lacks natural 3D control. It may lead to degenerate demonstrations that do not accurately represent free-form, 6-DoF movement.
- 3D Mouse: The 3D mouse interface allows simultaneous translation and rotation along multiple axes. The local movements of the 3D mouse are mapped to the robot end effector.
- Virtual Reality (VR) Controller: The VR Controller, specifically the HTC Vive, provides high-fidelity tracking and natural 3D control. However, it requires specialized VR hardware and may not be as readily accessible as other interfaces.
- Phone (RoboTurk): The Phone interface, implemented through RoboTurk, offers a mobile and convenient control option. However, it may have limitations in terms of accuracy and control precision compared to more specialized interfaces.
- Pose-control (Our work): We designed the Pose-control interface based on pose-tracking, aiming to combine the ease of control and accuracy of a VR controller with the ubiquity of a keyboard. This interface utilizes hand and pose tracking to interpret gestures and translate them into robot control commands. It offers an intuitive and accessible control method.

We evaluated the performance of our Pose-control interface based on the following metrics:

1) Number of Actions: We compared the average number of actions required for task completion using our Pose-control interface with the official datasets collected using Roboturk and machine-generated actions. Our Pose-control approach achieved a lower average number of actions, indicating more efficient and streamlined control for task completion.
2) Completion Times: We measured the completion times for each task across different interfaces. Our Pose-control interface demonstrated significantly lower completion times compared to the professional datasets and other interfaces in simple tasks like lifting a box. In more

complex tasks like fitting nuts, our Pose-control interface showed competitive completion times, outperforming the professional datasets and some other interfaces.

3) User Study Feedback: During the user study with 10 participants, we gathered qualitative feedback to assess the usability and user experience of our Pose-control interface. Participants reported that the interface was intuitive and easy to use, allowing them to perform tasks with minimal effort. The natural mapping of hand gestures to robot actions facilitated a sense of direct control and improved task performance.
4) Comparison with Other Interfaces: Based on the Kolmogorov-Smirnov statistics, our Pose-control interface showed statistically significant differences in completion times compared to the other interfaces, demonstrating improved control precision and reduced completion times.
5) Robustness Evaluation: We evaluated the robustness of our Pose-control interface by examining its performance under different user conditions and variations in gestures. This showcases the robustness and adaptability of our approach to accommodate user preferences and variations in gesture patterns.

Overall, our Pose-control interface offers a compelling alternative to traditional control interfaces, providing a balance between intuitive control, accessibility, and task performance. The results of our evaluation highlight the effectiveness, efficiency, user experience, and robustness of our Pose-control approach.

In order to test whether the trajectory datasets of robot arm completed by gesture control can be used for the training of intelligent robot arm, we conducted multiple experiments for lift task and summarized the trajectory datasets. After processing the datasets, we used three different algorithms, BC, HBC and IRIS, for model training. The results show that the model validation losses generally change from high to low with the increase of training times. However, all the models have small reduction range and poor stability after convergence, which will fluctuate greatly near a value. So as to find reference targets for the test results, we used the robomimic official datasets to train the models separately under the same algorithms. These models all have a large decline ranges and a small fluctuation after convergence, indicating that the trained models are obviously better than the models trained using the collected datasets. A more intuitive comparison is the value of validation success rate after each model converges. The results show that the success rates of the models trained with the robomimic datasets are significantly higher than that of the models trained with the collected datasets. However, for the HBC algorithm, the success rate of the model obtained by training with the collected datasets is close to 50%, which means that we have preliminarily realized the training of the intelligent robot arm. Applying this model to a robot arm can make it realize the specific task to a certain extent.

The quality of the datasets we collected are poor compared to the robomimic datasets, which is attributed to the challenges encountered during the data collection process. Due to the limitation of the camera in the simulation environment, the end-effector may poke the desktop and cause interference when controlling the end effector to grasp the task. In the process of machine learning, this behavior may lead to the unexpected failure of the arm's motion to converge. At present, the control method based on gesture recognition only realizes that one action corresponds to one direction of the robot arm, which leads to more steps in the process of data recording. As mentioned earlier, the collected datasets steps average around 200 with considerable fluctuations ranging from 100 to 960, presenting a stark contrast to the robomimic datasets average step length of around 50. The higher number of steps, the more complex the task and the more difficult machine learning becomes. It is worth noting that longer steps often require more data to support training.

In order to get good training results in offline reinforcement learning and imitation learning, sufficient data is also essential. With a maximum of 100 trajectories, our datasets pales in comparison to the robomimic datasets, which comprise over 200 trajectories. This scarcity of data leads directly to numerous situations where rewards cannot be effectively provided, subsequently impeding the convergence process.

In light of these observations, it becomes evident that substantial improvements are necessary to attain optimal performance. Notably, HBC algorithm is superior to the other two test algorithms both in convergence speed and success rate. Moving forward, addressing the identified issues, such as refining data collection methods to mitigate inaccuracies and expanding the datasets size, will undoubtedly contribute to the overall advancement of our approach. By undertaking these measures, we aim to bridge the performance gap and achieve results that align more closely with the standards set by the robomimic datasets.

VI. Conclusion

In this project, we have completed the function of gesture recognition and successfully transmitted the recognition information to the manipulator. The recognition information will be converted into velocity and acceleration information and transmitted to the robotic arm, which has the advantage of high flexibility. The smoothness and stability of gesture recognition have been verified.

In the experiment of training the trajectory of the robotic arm, we compared the BC, HBC, and IRIS algorithms based on convergence speed and success rate, and a low-level success rate was achieved.

Future work will focus on adapting to other visual device (such as improving latency). At the same time, gesture recognition has been realized to control the robot arm, so we can continue to try to control the movement of a complete robot in a similar way. In addition, one of the important reasons for the insufficient success rate of imitation training is the poor quantity and quality of sample datasets, so we will continue to make efforts in improving sample datasets.

ACKNOWLEDGMENTS

In this section, we introduce our team division of work.

Chen Yifei 12010502: 1. Gestures control system. 2. Data action recording. 3. Simulation environment. 4. Methods and Experiments setting.

Yang Deyu 12012323: 1. Manipulator Training. 2. Models and datasets experiments. 3. Result Evaluation.

Zhu Siying 12010242: 1. Datasets collection. 2. Video demo.

Ning Chenhui 12010609: 1. Related work. 2. Model interpretation.

Fang Yijun 12011301: 1. Training result processing. 2. Video demo.

REFERENCES

[1] Mandlekar, A., Zhu, Y., Garg, A., Booher, J., Spero, M., Tung, A., ... , Fei-Fei, L. (2018, October). Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In Conference on Robot Learning (pp. 879-893). PMLR.

[2] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. arXiv preprint arXiv:1703.09312, 2017.

[3] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016.

[4] D.Kalashnikov,A.Irpan,P.Pastor,J.Ibarz,A.Herzog,E.Jang, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. arXiv preprint arXiv:1806.10293, 2018.

[5] B. Cheng, W. Wu, D. Tao, S. Mei, T. Mao and J. Cheng, "Random Cropping Ensemble Neural Network for Image Classification in a Robotic Arm Grasping System," in IEEE Transactions on Instrumentation and Measurement, vol. 69, no. 9, pp. 6795-6806, Sept. 2020, doi: 10.1109/TIM.2020.2976420.

[6] RoboTurk: Robotic Skill Learning with Large Human Datasets, https://roboturk.stanford.edu/

[7] BOULABIAR M I, COPPIN G, POIRIER F. The Study of the Full Cycle of Gesture Interaction, the Continuum between 2D and 3D[C].16th International Conference, HCI International 2014, 2014

[8] Zeng Hualin, Huang Yuxuan, Chao Fei. A review of research on handwriting robots [J]. Journal of Intelligent Systems, 2016, (1): 15-26.

[9] K.-T. Yu, M. Bauza, N. Fazeli, and A. Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In Int'l Conference on Intelligent Robots and Systems, 2016.

[10] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with large-scale data collection. In ISER, pages 173–184, 2016.

[11] K. Fang, Y. Zhu, A. Garg, A. Kurenkov, V. Mehta, L. Fei-Fei, and S. Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. In Robotics: Systems and Science, 2018.

[12] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen. The columbia grasp database. In Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, pages 1710–1716. IEEE, 2009.