Dexterous Manipulation on Multi-Fingered Hands Based on 3D Diffusion Policy

Author: Dong, Ju
Supervisor: Dr. Li, Shuang

December 26, 2024

Abstract

This work introduces the deployment of **3D Diffusion Policy**(**DP3**), a novel generative framework for robotic behavior synthesis utilizing 3D point clouds as input. The framework is implemented on the Franka Panda robotic arm and the Allegro dexterous hand. A teleoperation system was developed for data set collection, combining the OptiTrack V120:Trio for wrist tracking and the Manus Quantum Mocap Metagloves for finger motion capture. Control of the robotic arm and dexterous hand, teleoperation workflows, and policy deployment were integrated into a unified ROS workspace to streamline deployment and tuning. Realistic models and environments were constructed in simulation using MoveIt, ensuring safety during data collection and policy deployment. The performance of the 3D Diffusion Policy was validated through deployment on the Franka Panda robotic arm and the Allegro dexterous hand, demonstrating its effectiveness in advanced robotic manipulation tasks.

Contents

1 Introduction

Reinforcement learning and imitation learning are two ways for robots to acquire dexterous manipulation skills, with imitation learning being an effective way to teach robots a variety of motor skills (e.g., grasping, mobile manipulation, etc.) [1] [2] [3] [4]. Diffusion models have achieved success in high-fidelity image generation and have been applied in various aspects of robotics, including reinforcement learning, imitation learning, reward learning, grasping, and motion planning. Diffusion Policy(DP) shows very good results and great potential in solving robot dexterity manipulation execution. A breakthrough generative AI approach based on Diffusion Policy can quickly teach robots new dexterity skills [5]. Many teams have made outstanding innovations using DP: [6],[1],[7],[8],[9].

While Diffusion Policy performs very well, its input is 2D images, which results in poor generalization performance under limited data. To address this issue, the was introduced [10]. It combines 3D visual representations with diffusion policies, enabling robust and generalizable skill learning with fewer demonstrations. The core innovation lies in enhancing the model's perceptual capabilities by using richer visual information, such as 3D data. The performance of Diffusion Policy is indeed impressive, but its input is 2D images, which leads to poor generalization performance under limited data. To address this issue, the idea of using richer visual information, such as 3D information, to enhance the model's perception capability was proposed [10] [11]. The **3D Diffusion Policy(DP3)** validated in this paper combines a 3D visual representation with a diffusion policy that can learn robust and generalizable skills with a small number of demonstrations.

As mentioned earlier, expert demonstration data is crucial in imitation learning. Human demonstration data is often collected using teleoperation [12] [13]. In DP3 a keyboard is used to control the robotic arm, and then a camera is used to capture and estimate the finger state to the dexterous hand, which is not integrated in an environment. Moreover, the safety of data collection is ensured only by a manual emergency pause. So in this thesis a complete, real time, secure teleoperation acquisition system was created for data collection.

The Thesis is divided into five main chapters. This chapter is an overview of the thesis. Chapter 2 introduces the structure of DP3, Chapter 3 discusses the construction of the teleoperation system and collection of data sets, and Chapter 4 covers the train and deployment of the policy and its results. Chapter 5 provides a conclusion of the paper and discusses future directions.

2 3D Diffusion Policy

The background and innovations of DP3 have been described in Chapter 1. This section describes the network architecture of DP3 and its features.

2.1 Approach

As shown in Figure 1, the policy structure consists of two parts: perception and decision-making.

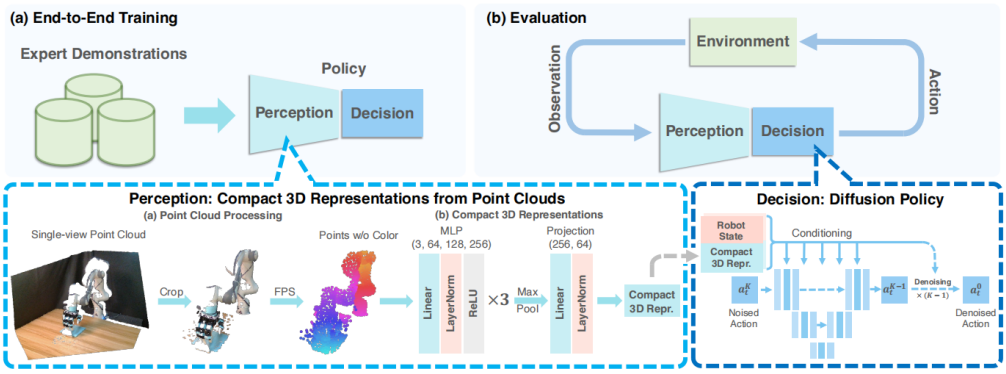


Figure 1: Overview of 3D Diffusion Policy (DP3) [10]

1. **Perception Module:**

   - **Camera Setup:** DP3 employs a single-view camera for policy learning across all tasks. This is in contrast to prior works that set up multiple cameras around the robot, making DP3 more practical for real-world applications.
   - **3D Scene Representation:** The 3D scene is represented using sparse point clouds, which are more efficient compared to other explicit representations like RGB-D, depth maps, or voxels. Depth images of size 84×84 are captured using a single camera, and they are converted into point clouds using the camera's intrinsic and extrinsic parameters. The color channel is excluded to improve appearance generalization.
   - **Point Cloud Processing:** Redundant points (e.g., points on the table and ground) from the depth-converted point cloud are cropped, retaining only the points within the bounding box. Further downsampling is performed using Farthest Point Sampling (FPS), where 512 or 1024 points are found to be sufficient for all tasks in both simulation and real-world experiments. FPS helps cover the 3D space effectively while minimizing sampling randomness.
   - **Compact Representation Encoding:** A lightweight MLP network, the *DP3 Encoder*, is used to encode the point cloud into a compact 3D representation. This network consists of three MLP layers, a max-pooling function to aggregate point cloud features, and a projection head that maps features to a compact vector. LayerNorm layers are interleaved between MLP layers to stabilize training. The resulting 3D feature vector v

has a dimension of only 64. This simple encoder outperforms pre-trained point cloud encoders, such as PointNeXt, in visual-motor control tasks.

2. **Decision Module:**

   - **Conditional Action Generation:** The decision module is based on a conditional denoising diffusion model. It takes the 3D visual feature v and the robot pose q as conditions and denoises random Gaussian noise into an action a. Starting from Gaussian noise, the denoising network iteratively refines the noise over K steps, gradually transforming it into a noise-free action.
   - **Training Objective:** To train the denoising network, a data point is randomly sampled from the dataset, and a diffusion process is applied to generate noise at iteration k. The training objective is to predict the noise added to the original data, where β_k and α_k represent the noise scheduling parameters for a single-step noise addition.

2.2 Experiments and Conclusions

1. **Simulation Experiments:**

   - **Experimental Setup:**
     - **Simulation Benchmark:** A total of 72 tasks across 7 domains were collected, covering various robotic skills with varying task difficulties, ensuring that the benchmark is not limited by the choice of simulator.
     - **3D Observation:** Visualized in point cloud form, expert demonstration data comes from various sources, including human teleoperation, scripted policies, and trajectories from reinforcement learning training.
     - **Main Baseline Methods:** Image-based Diffusion Policy, compared with IBC, BCRNN, and their 3D variants (evaluated on only 10 tasks).
     - **Evaluation Metrics:** Success rate was used as the evaluation metric. Each experiment ran with 3 seeds, and for each seed, 20 episodes were evaluated every 200 training iterations. The average of the top 5 success rates was taken.
   - **Ablation Study:**
     - **Tasks:** Performed on 6 tasks, including Hammer, Door, and Pen from Adroit, and Assembly, Disassemble, and Stick-Push from MetaWorld, covering both high-dimensional and low-dimensional control tasks. Each task only used 10 demonstrations.
   - **Experimental Results:**
     - **Efficiency and Effectiveness:**
       * DP3 achieves an average success rate of 74.4% across 72 simulation tasks, a 24.2% improvement over Diffusion Policy. It achieves over 90% success rate in nearly 30 tasks, while Diffusion Policy reaches this success rate in less than 15 tasks.

    * DP3 converges quickly during training, typically within about 500 iterations, while Diffusion Policy converges slowly or reaches suboptimal results.
    * In Adroit tasks, DP3 achieves comparable accuracy with Diffusion Policy using fewer demonstrations, and in some MetaWorld harder tasks, DP3 performs better when sufficient demonstrations are available.
    * DP3 inference speed is slightly faster than Diffusion Policy, mainly due to the use of sparse point clouds and compact 3D representations.

2. **Real-World Experiments:**
    - **Experimental Setup:** DP3 was evaluated on 4 tasks across 2 different robots (Allegro Hand and Gripper), including Roll-Up, Dumpling, Drill, and Pour, using the RealSense L515 camera for visual observations. The tasks involve multi-stage operations, with object position and appearance randomized. Expert demonstrations were collected via human teleoperation, with only 40 demonstrations provided for each task.
    - **Experimental Results:** The high success rate of DP3 with only 40 demos motivated this thesis to deploy DP3 in a real scenario to validate its effectiveness.

The structure of the DP3 and the experiments are described in detail in this section, and the next section describes the teleoperation system and the process of data acquisition for this thesis.

3 The Teleoperation System and Collection of Data

This chapter introduces the teleoperation system developed in this work, the scenarios set up to validate the DP3 policy, and all the preparations made prior to data collection. As introduced in the previous chapter, the input to the DP3 policy includes both point clouds and robotic arm actions. Therefore, this teleoperation system also encompasses both visual and action dimensions. The user can move their wrist and perform grasping motions in space, and the robotic arm will execute the same actions in real time. At the same time, the camera records the scene of robotics.

As shown in Figure 2, this is a schematic of the real-world setup used in this work. In the teleoperation section, OptiTrack performs real-time tracking of the wrist's attitude in space and publishes wrist pose to the end-effector of the Franka Arm. Meanwhile, the Manus glove captures finger joint angles and publishes and maps them to the Allegro hand, enabling control of both the Franka and Allegro integrated models. In real-world experiments, image- and depth-based diffusion strategies often exhibit unpredictable behavior and require manual termination to ensure safety. An Allegro dexterity hand was also damaged once in the DP3 paper, so MoveIt was used in this paper to avoid collisions. Real environments and models were also created in the simulation environment through MoveIt, and collisions were also detected in tracking and observed in real time in RVIZ.

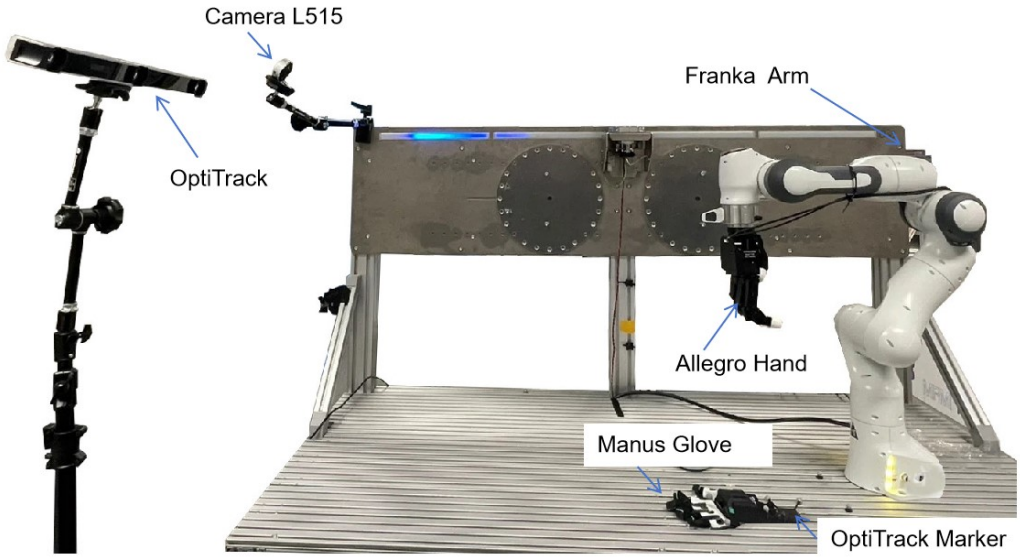


Figure 2: Setting of the experiment in real-world

The camera L515 and Aruco markers were prepared specifically for point cloud acquisition. Since DP3 focuses on visuomotor policy learning, the quality of the point clouds used for training and deployment is crucial. In this work, a LiDAR camera was employed to generate high-quality point clouds. Additionally, by placing Aruco markers on the table, the camera coordinate system was transformed into the coordinate system of the robot's workspace, facilitating more efficient point cloud cropping and farthest-point sampling.

The entire teleoperation and data collection process are integrated in the ROS environment, providing great convenience for operation and debugging. Table 1 lists the hardware, software, and corresponding ROS packages used in the teleoperation system for data collection. In the table referred to the library pyrealsense2 and software RealsenseViewer for L515 are used to tune the configuration of the camera.

Handware	**Library/Software**	**Ros Packages**
Franka Emika Panda	Franka Control Interface	franka_ros, MoveIt [14],[15]
Allegro Hand	/	Allegro Hand Controller, MoveIt [16]
OptiTrack V120:Trio	Motive	NatNet 4 ROS driver [17]
Manus Quantum Mocap Metagloves	/	manus_ros
Realsense LiDAR L515	pyrealsense2, RealsenseViewer	librealsense2, realsense2_camera [18]

Table 1: Hardware and packages for teleoperation systems

The following section will provide a detailed explanation of the experimental setup and the characteristics of the hardware used.

3.1 Franka Emika Panda and Allegro Hand

In the thesis, the robotic arm and dexterous hand are integrated using MoveIt with the entire ROS ecosystem, enabling better control and simulation of the robot. In addition, MoveIt provides trajectory planning for higher-level control of the robotic arm, ensuring safety within the environment.

Figure 3: Franka and Allegro Robot modeled with MoveIt in RVIZ

The robot within the real scenario is the same as demonstrated in DP3 (in Figure 3), using a Franka Emika Panda 7-axis robotic arm integrated with an Allegro dexterous hand. In the chapter 6 at the end of thesis, the transformation tree of the whole robot system is provided for reference. This Figure 16 gives a clear hierarchical representation of the components and relationships within the system. Each node represents a component, and the edges indicate their hierarchical relationships. The Franka robotic arm has a payload of 3 kg and a reach of 850 mm. The robot has a repeatability of 0.1 mm and it weighs approximately 18 kilograms. The Allegro Hand has four fingers with a total of 16 degrees of freedom for flexible gripping and manipulation. The movement of the fingers is fully driven by high-precision servo motors, providing smooth and precise motion control. This combination can effectively accomplish most tasks in real world, such as grasping, rolling, and pushing tasks.

The Franka arm and Allegro Hand are controlled separately using their respective controllers, and then integrated within MoveIt. The scene also includes walls and a table to represent the environment. This allows the robot to check collisions during movement, ensuring tasks are executed safely and without collisions. In the figure, the end-effector of the model is set at the center of the

flange connecting the Franka arm and Allegro hand. The robot arm is controlled by dragging the TCP position in RViz.

3.2 OptiTrack V120:Trio

Figure 4 shows the OptiTrack V120:Trio, a device used in the teleoperation system to capture wrist poses. Its three high-resolution cameras accurately track the rigid body's pose in space with markers and publish the data at a frequency of 120Hz, providing high accuracy and real-time performance.

Figure 4: OptiTrack V120:Trio - 6DoF optical tracking with multiple cameras [19]

The user interface of Motive, the software used to configure OptiTrack, is shown below and can be divided into three sections (Figure 5). The left section allows setting the origin of the world coordinate system, which in this work was positioned 5 cm in front of the robotic arm's base. The right section displays the wrist rigid body setup, where the rigid body is attached to the user's wrist. The middle section shows the tracking of the wrist in space. Additionally, Motive provides options for smoothing the wrist motion trajectory. The 6DoF pose of the wrist in space is published to a ROS topic by configuring NatNet 4 ROS.

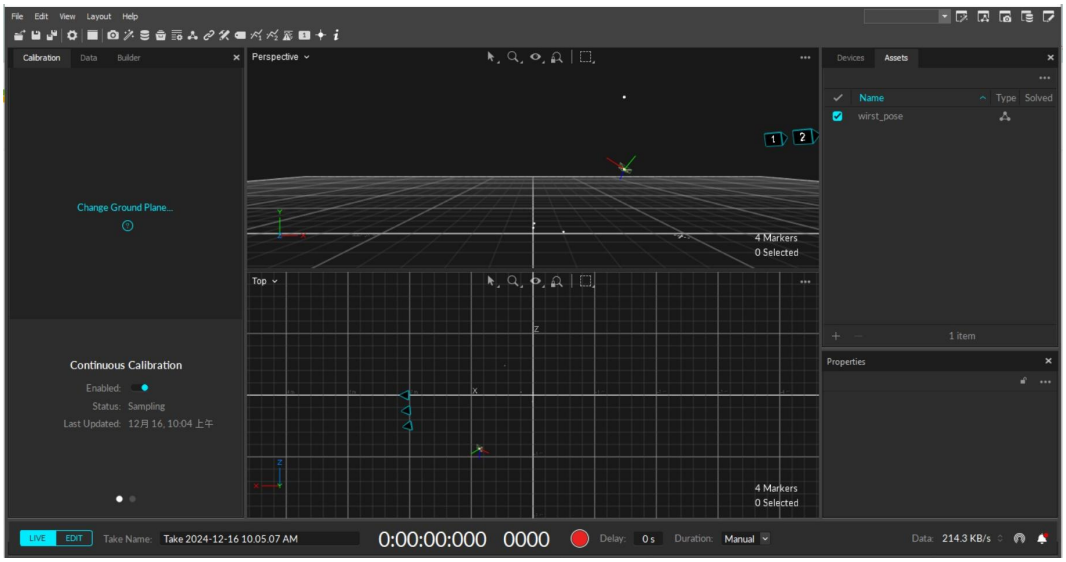


Figure 5: Motive Interface

3.3 Manus Quantum Mocap Metagloves

The Manus Quantum Mocap Metagloves in figure 6 were used in this work to collect finger joint angles, which were then mapped to the dexterous hand. These gloves use Quantum Tracking technology to track hand movements at 300Hz with high speed and accuracy. Since the Allegro used in the paper is a 4-finger dexterous hand, the pinky was omitted from the mapping.

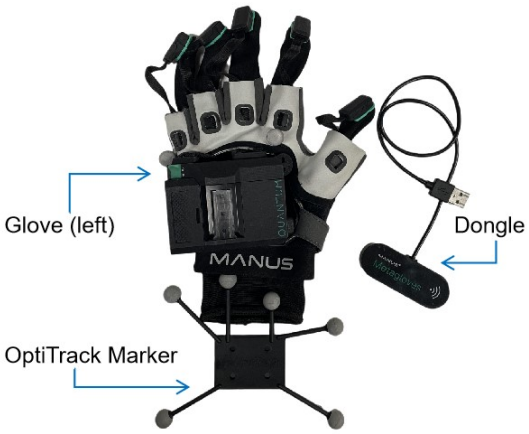


Figure 6: Glove with Dongle and OptiTrack's marker

3.4 Realsense LiDAR L515 and Aruco

The Intel RealSense LiDAR Camera L515 (see Figure 7) provides excellent depth quality for indoor applications. It generates 23 million precise depth points per second, enabling accurate point cloud acquisition.

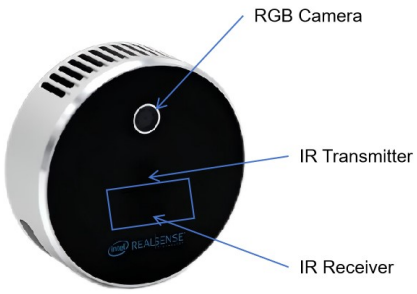
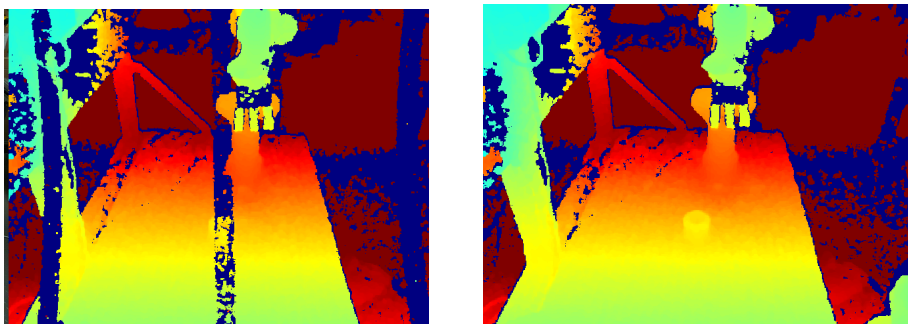


Figure 7: RealSense LiDAR Camera L515

In the teleoperation system used in this thesis, interference occurs in the depth map due to both the OptiTrack system and the LiDAR Camera L515 using 860nm infrared light, resulting in three structured light patterns from the OptiTrack (see Figure 8(a)). From a physical perspective, it is necessary to find appropriate camera positioning and angles to minimize interference, such as avoiding relative or perpendicular placements, which cause the most significant interference. From an algorithmic perspective, interpolation techniques can be used for compensation.



(a) Original depth image (b) Repaired depth image

Figure 8: Comparison of original depth image and repaired depth image in Realsense Viewer

The two ROS packages used here are librealsense2, the underlying library driver for communicating with the camera, and realsense2_camera, the ROS camera node for publishing camera data. Since the Python environment is easy to configure, we first use pyrealsense2 and RealSense Viewer to adjust the camera configuration. During tuning, the “Short Range” is modified via the Realsense viewer to reduce the camera’s laser power and receiver gain values, which has a positive effect on the image, as shown in Figure 8(b). While in the ROS environment, “Short Range”presets for the L515 are selected via a drop-down menu in the *dynamic_reconfigure* interface of ROS.

Before generating the point cloud in the next section 3.5, the depth map was interpolated to further mitigate the impact of voids caused by interference. To address missing values in the depth map, a two-step repair strategy was employed: first, Nearest Neighbor Interpolation was applied to preliminarily fill large missing regions. Then, Median Filtering and Bilateral Filtering were used for refinement. The Bilateral Filter was configured with a neighborhood diameter $d = 5$ and standard deviations $\tau = 50$ for both color and coordinate spaces. This parameter setup effectively removes noise while preserving depth edge features. The proposed method successfully repairs data voids generated during depth sensor acquisition while maintaining the geometric integrity of the scene.

Because occlusion affects data quality, it is important to capture the full scene. Here the camera is suspended on the opposite side of the robot arm, again to minimize the effects of OptiTrack’s structured light. To subsequently crop off the back tabletop and background, Aruco was used to estimate and adjust the camera’s pose. In Figure 9, the position of the Aruco calibration table was used, in order to transfer the camera coordinate system to the coordinate system of the robot arm.

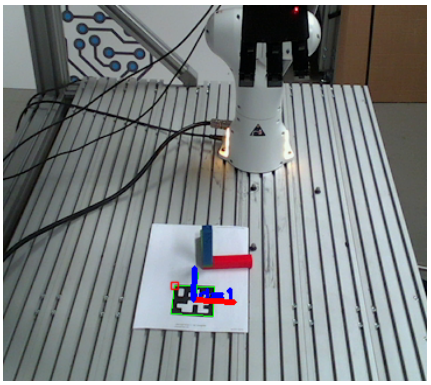


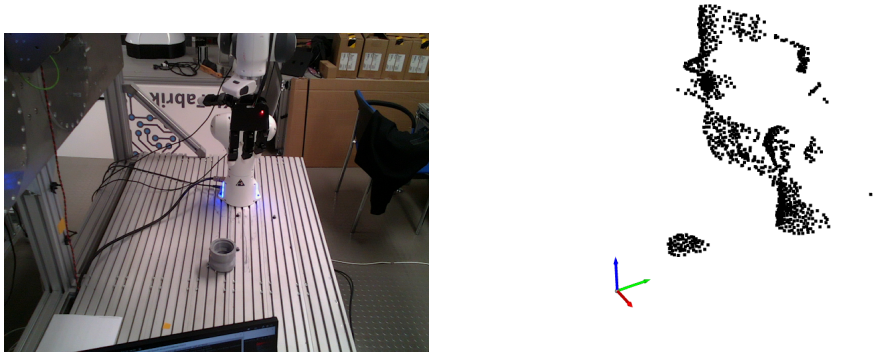
Figure 9: Calibrated coordinate system of Aruco

Theoretically, the camera position remains consistent between data collection and deployment, and does not change after being set. Therefore, the extrinsic matrix is obtained using OpenCV library in python and then hard-coded into the point cloud generation process.

3.5 Data Collection Process

The teleoperated system used, as well as its toolkit, was described in detail in the previous sections, and the system now needs to be used to collect and produce the datasets for the DP3.

In this thesis, tasks similar to those in the DP3 paper, such as Dumpling and Pour, are set up, combining both visual and robotic arm information. The robot is tasked with pushing a cylindrical object placed in front of it, with the goal of pushing it to a marked position. Figure 10(a) shows the raw image captured by the camera. In real-world environments, the influence of the background is inevitable, so point cloud processing is necessary. After cropping the workspace, the point cloud in Figure 10(b) only contains the robot and the object. This allows the network to focus more on processing the critical information rather than the redundant background. It is important to state that the DP3 model generates point clouds from 84x84 images and depth maps, but in actual training it was found that the quality of point clouds generated this way is not good. For this reason, this paper used the original 480x640 image size to get a very accurate point cloud.



(a) Raw image for the demo

(b) Generated point clouds

Figure 10: Original top view image and cropped point cloud

In addition to visual information, the network also requires the robot's state and action information. In the DP3 paper, the end-effector pose of the robot arm was changed from Euler angles to quaternion representation, as quaternions provide a more accurate and efficient way to represent rotations. In the original tasks, action is defined as the relative position between two states. In this Push task, the absolute position is used. So with the 7 dimensions of the end-effector plus Allegro's 16 joint values, there are 23 dimensions of state and action. After 11 rounds, a set of demonstrations with the same target position was collected to form the dataset for the policy. After compression by the zarr toolkit, the final shape of the data and structure of the dataset is shown below in Figure11.

```
 Dataset basic information:
Image shape: (1446, 480, 640, 3), range: [0, 255]
Point cloud shape: (1446, 1024, 3), range: [-0.2483568638563156, 0.8497783541679382]
Depth shape: (1446, 480, 640), range: [0.0, 9160.0]
Action shape: (1446, 23), range: [-1.0, 1.4920414686203003]
State shape: (1446, 23), range: [-1.0, 1.4920414686203003]
Frames: 1446

 Dataset structure:
├── data/
│   ├── action: (1446, 23) (float32)
│   ├── depth: (1446, 480, 640) (float64)
│   ├── img: (1446, 480, 640, 3) (uint8)
│   ├── point_cloud: (1446, 1024, 3) (float64)
│   └── state: (1446, 23) (float32)
└── meta/
    └── episode_ends: (11,) (int64)
```

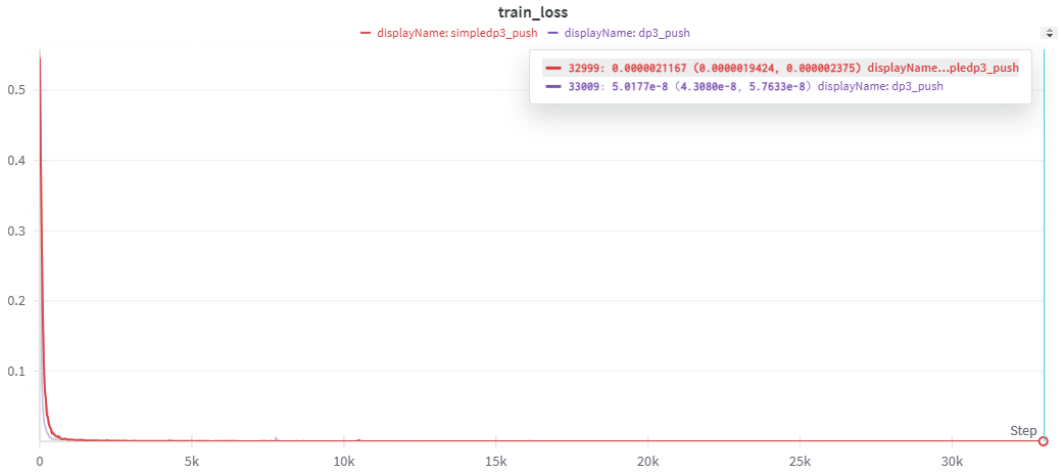
Figure 11: Structure of the processed dataset

The DP3 can then be trained on the collected data to verify the performance of the network on its own task.

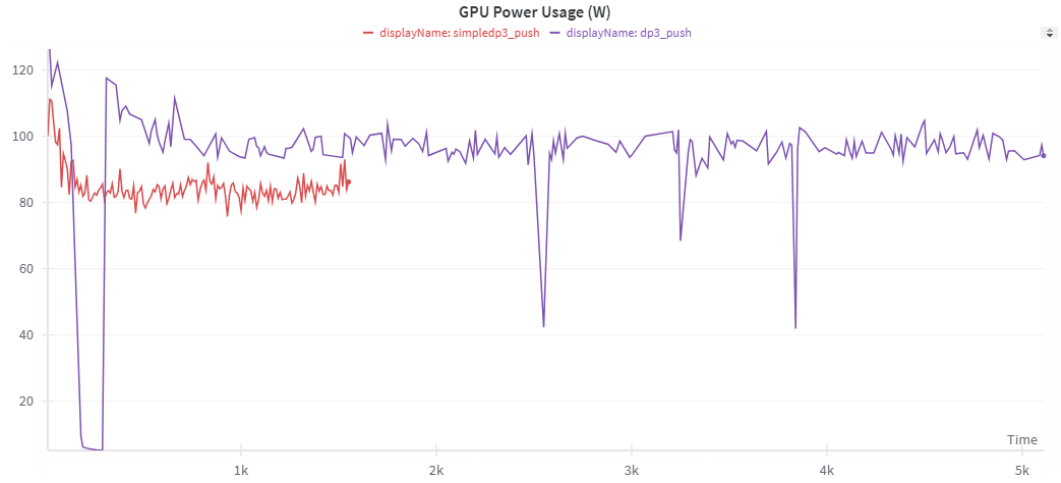
4 Train and Deployment

4.1 Train

Before training, a specific .yaml file is created for the pushing task, where the input and output dimensions are configured. Figures 12(a) and 12(b) illustrate two models provided by the DP3 paper: DP3 and SimpleDP3. DP3 is the underlying network architecture and SimpleDP3 is lightweight.



(a) Performance Metric



(b) Resource Usage

Figure 12: Comparison of training results between DP3 and SimpleDP3

From the performance metrics and resource usage observed during runtime, the strengths and weaknesses of each network can be evaluated as follows:

- **DP3:**
  - **Strengths:** Achieves extremely low loss values, demonstrating its ability to capture fine-grained details and complex behaviors from the dataset.
  - **Weaknesses:** High computational cost in terms of training time and GPU resource consumption, making it less practical for real-time applications or deployment on resource-limited hardware.
- **SimpleDP3:**
  - **Strengths:** Efficient in training and inference, with lower GPU memory consumption and faster deployment times. Suitable for environments where computational efficiency is a priority.
  - **Weaknesses:** While the performance is acceptable, the loss valuesare slightly higher than DP3, which might result in minor compromises in precision or generalization capability.

By comparing Figures 12(a) and 12(b), it becomes evident that the choice between DP3 and SimpleDP3 should be guided by the specific requirements of the application, such as accuracy versus computational efficiency. While DP3 achieves a lower loss value (e.g., 0.0000000501) compared to SimpleDP3 (e.g., 0.00000212), the SimpleDP3 strikes a balance between computational efficiency and acceptable performance, making it a preferred choice for initial deployment tests.

4.2 Deployment

After training, a .ckpt file for this task will be generated and saved. This file is then deployed onto the robot. During deployment, it is necessary to set up an environment wrapper to help the robot understand the working environment. As described in chapter 3, all operations are integrated into a ROS environment. In this thesis, the computer controling robot operations and the one performing network inference are not the same. Therefore, it is necessary to set up ROS for bidirectional communication between the two computers. The whole process is shown by the figure 13 below. On the control computer, a state topic is published, which combines the end-effector pose and Allegro Hand joint variables, along with topics for images and depth maps from the L515 camera. The training computer subscribes to these topics, processes and crops the images to generate real-time point clouds, and combines them with the state data as input to the network. The network performs inference, outputs actions, and publishes them back. The execution computer then subscribes to the action topic and forwards the commands to the controllers of the Franka arm and Allegro Hand. The entire execution process is monitored within the safety constraints provided by MoveIt.

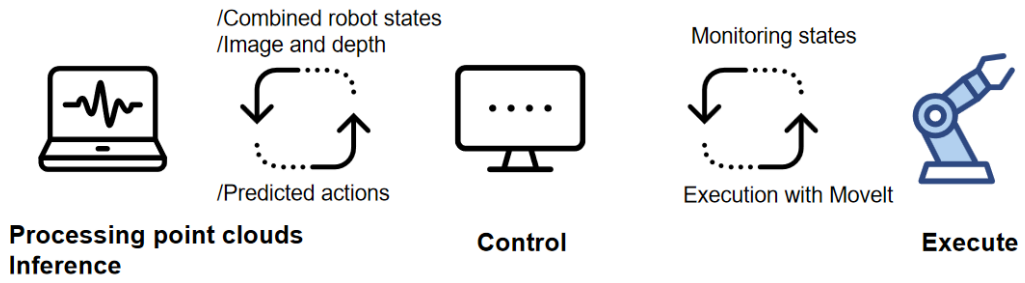


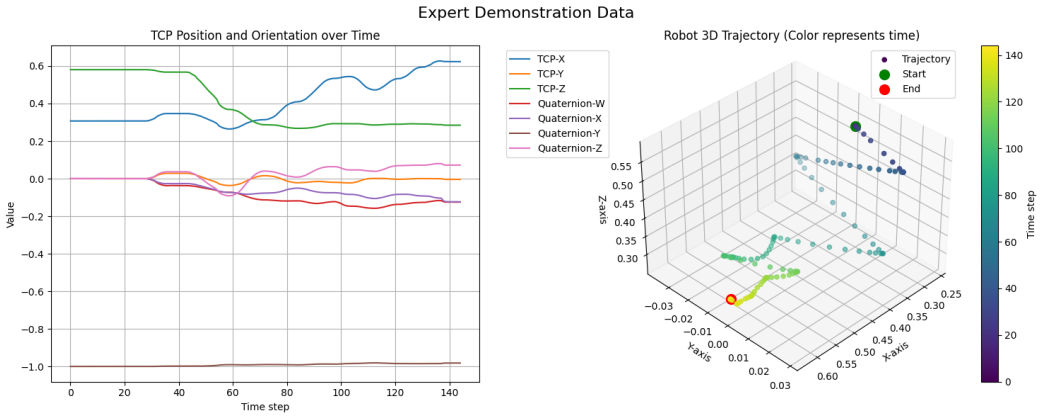
Figure 13: Flow diagram of deployment

5 Conclusion and Outlook

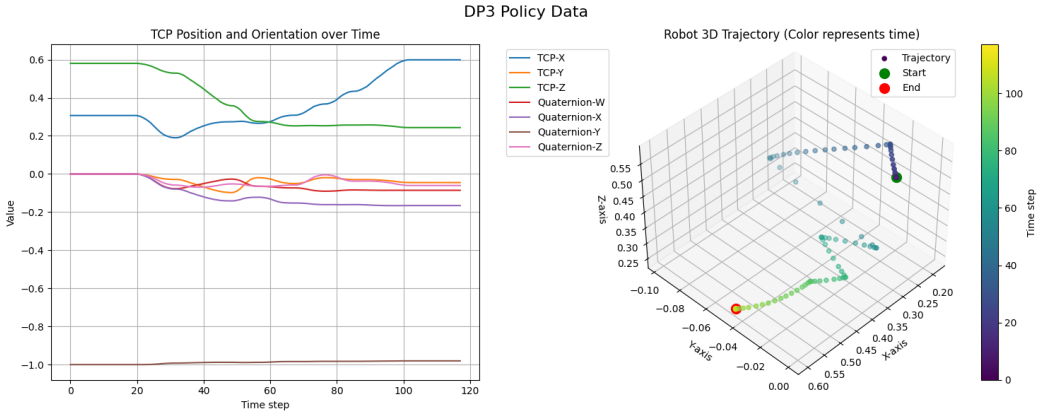
After deployment, DP3 performs push tasks on a real robot. After collecting eleven sets of demonstrations of pushing a cylindrical object in front of the robot, the robot could understand the task. It would approach the cylinder and then push forward until it was over the marked line. In addition, after replacing the object from the cylinder of the demo data to an apple, it was also able to perform the pushing task(in Figure 14). So it is concluded that the policy is generalizable.

Figure 14: Try a new task: push an unseen object (An apple)

Then compare the manual data with the results reasoned by the DP3 policy. As shown in Figure 15(a), the left side is an expert demo showing the 7-DoF of TCP, and the right side is a trajectory plotted based on the step size. Figure 15(b) shows the trajectory of DP3 reasoning executed on a real robot. By comparing the two, it is clear that the DP3 reasoning results are consistent with the expert demonstration results and successfully accomplish the expected task.



(a) Expert Demonstration: TCP and Trajectory



(b) DP3 Policy: TCP and Trajectory

Figure 15: Comparison of trajectory of expert demonstration with trajectory of DP3 policy

After validation, the DP3 policy performs well in real-world environments. However, there are still parts for improvement.

1. In the validation it was found that DP3 is very dependent on the quality of the point cloud. From if a single depth camera must be L515 or ZED2, and the light has a strong influence on the test results. In the vision module only a simple MLP is used, whereas in the latest RDT-1B and Pi0 both use Transformers, which should be a boost.

2. When attempting to use a small number of demonstrations, the generalization ability of DP3 is poor. If the target position is changed, the robot arm cannot understand the task. Additionally, each task requires separate demonstration data collection and cannot be applied in a multi-task environment.The authors of RDT-1B propose that diffusion networks are typically small, often only tens of megabytes, allowing for fast inference in robot action tasks, but this also limits the expressive power of the model.

3. Whether it is a strategy such as DP, DP3 or more recently RDT-1B, TCPs of real robots is mostly a two-finger gripper, not dexterous hands with a dozen or so degrees of freedom. DP3 uses multiple fingers, but the multiple fingers are not reflected in the actual task. During the experiment, 30 demonstrations of grasping the box wall over the top of the box were collected, but the dexterous hand control was found to be poor. It was verified that the DP3 still needs to be improved in dexterous hand grasping.

6 Appendix

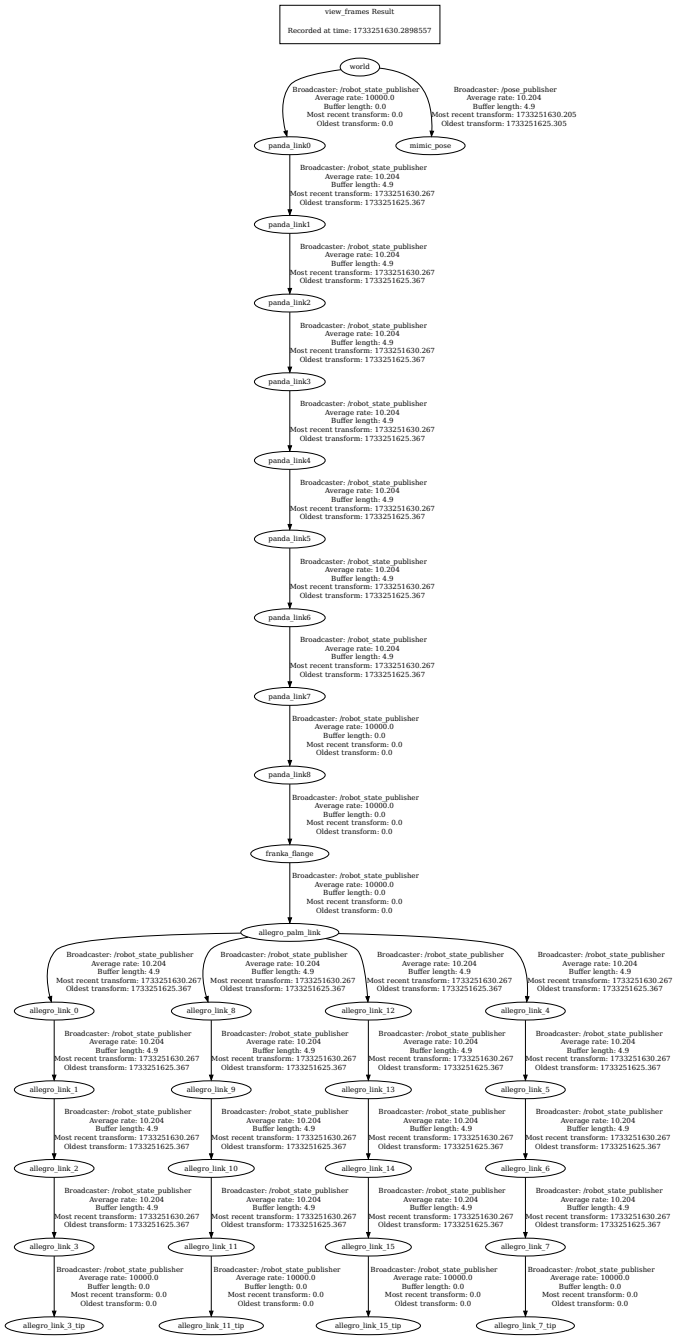


Figure 16: Transformation tree of the whole robot system

References

[1] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3651–3657, IEEE, 2019.

[2] C. Wang, X. Luo, K. Ross, and D. Li, "Vrl3: A data-driven framework for visual deep reinforcement learning," in *Conference on Neural Information Processing Systems*, 2022.

[3] Z. Jiang, Y. Xie, K. Lin, Z. Xu, W. Wan, A. Mandlekar, L. Fan, and Y. Zhu, "Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning," 2024.

[4] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[5] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, p. 02783649241273668, 2023.

[6] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, "Rdt-1b: a diffusion foundation model for bimanual manipulation," *arXiv preprint arXiv:2410.07864*, 2024.

[7] Y. Su, X. Zhan, H. Fang, Y.-L. Li, C. Lu, and L. Yang, "Motion before action: Diffusing object motion as manipulation condition," *arXiv preprint arXiv:2411.09658*, 2024.

[8] Y. Mu, T. Chen, S. Peng, Z. Chen, Z. Gao, Y. Zou, L. Lin, Z. Xie, and P. Luo, "Robotwin: Dual-arm robot benchmark with generative digital twins (early version)," *arXiv preprint arXiv:2409.02920*, 2024.

[9] R. Ding, Y. Qin, J. Zhu, C. Jia, S. Yang, R. Yang, X. Qi, and X. Wang, "Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning," *arXiv preprint arXiv:2407.03162*, 2024.

[10] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*, 2024.

[11] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3d diffuser actor: Policy diffusion with 3d scene representations," *Arxiv*, 2024.

[12] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox, "Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system," in *Robotics: Science and Systems*, 2023.

[13] D. Rakita, B. Mutlu, and M. Gleicher, "A motion retargeting method for effective mimicry-based teleoperation of robot arms," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 361–370, 2017.

[14] F. Emika, "franka_ros," 2024. Accessed: 2024-12-16.

[15] MoveIt!, "Moveit!," 2024. Accessed: 2024-12-16.

[16] N. R. Learning, "Allegro-hand-controller-dime," 2024. Accessed: 2024-12-16.

[17] L2S-lab, “natnet_ros_cpp,” 2024. Accessed: 2024-12-16.

[18] I. RealSense, “realsense_ros,” 2024. Accessed: 2024-12-16.

[19] “Optitrack v120:trio.” Accessed December 21, 2024.