Toward Autonomous Dexterous Manipulation using Diffusion Policies with a Humanoid Robot

Toby Buckley, Kevin Lynch, J. Edward Colgate

***Abstract*— An autonomous machine learning agent was trained using demonstration data to perform a dexterous manipulation task using the Dexterity Nexus (DexNex) upper-limb robot testbed. Denoising Diffusion Probabilistic Models were used to clone the behavior of the teleoperator. The diffusion model was able to learn and perform the task, but its performance was worse than the teleoperation data it was trained from. The drop in performance is likely a combination of lack of demonstration data, limitations of the model, and slow trajectory execution. More demonstration data and more advanced trajectory execution methods are needed to realize the full potential of this technology.**

This is the first demonstration of an autonomous agent controlling the DexNex hardware setup, with a task output space of 21 joint positions. This work is the start of the HAND Engineering Research Center's effort to develop autonomous capabilities for controlling high-degree-of-freedom manipulators. Characterizing its limitations will inform future design decisions for developing dexterous systems.

I. Introduction

This abstract presents our progress in developing an autonomous manipulation controller for an anthropomorphic upper-limb robotic system. The system, DexNex (Figure 1), was developed to be a testbed for experimenting with advanced hardware, software, controls, and algorithms for dexterity for the US National Science Foundation HAND Engineering Research Center [1].

Prior DexNex work has focused on maximizing teleoperation performance on several benchmarking tasks. One such task is the Box and Blocks task, where a single hand is used to grasp and move small cubes from one box to another, one at a time [2]. The task's score is how many blocks one can transfer in a single minute. DexNex teleoperation performance on Box and Blocks is greatly lacking compared to direct manipulation by a human, however; an experienced teleoperator achieves a 15x lower score than typical humans manipulating directly [1]. Other work by Kuling et al. has shown a 13.3x reduction or as little as a 4.6x reduction for the Tactile Telerobot [3, 4], meaning that DexNex performed worse compared to other similar teleoperation systems.

Research at the HAND Center is exploring two paths: (1) improving teleoperation performance via improved robot hands, advanced haptic interfaces, and shared autonomy; and (2) training autonomous dexterous manipulation policies based on teleoperation. This abstract describes our initial work along the latter path.

In recent years much progress has been made in developing autonomous controllers for robotic systems. Action Chunking with Transformers uses a novel transformer-based algorithm to generate robot action sequences for controlling low-cost manipulators [5]. Google DeepMind's RT-2 casts actions as natural language in a vision-language-action model to leverage existing internet datasets to generate robot trajectories [6].

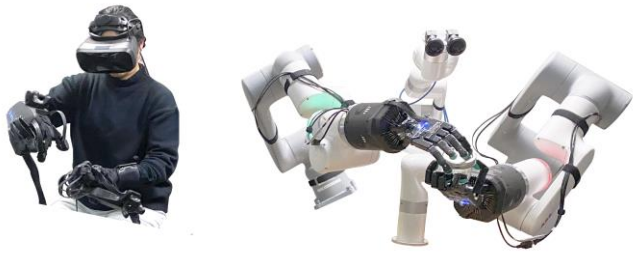

Figure 1. Dexterity Nexus (DexNex) testbed. Left: Operator station. Right: Avatar station. Only the upper limbs are tracked and mirrored. Feedback is provided to the user visually and haptically.

Denoising Diffusion Probabilistic Models (DDPMs, or diffusion models) are generative models which are best known for generating high fidelity realistic images from input text prompts [7]. The DDPM technique was applied to generate robot trajectories by Chi et al. with great success [8]. They used diffusion models to autonomously control Franka Panda robots to perform 15 different in-simulation and in-real-world manipulation tasks (e.g., Multimodal Block Pushing, Push-T, Mug Flipping, and others). In their work, a different agent was trained for each task, using teleoperation demonstrations of the task. Their approach outperformed alternative state-of-the-art autonomous algorithms.

Work by Ze et al. has advanced diffusion models for dexterous manipulation by incorporating 3d representations of objects using sparse point clouds [9].

The work described in this abstract adopts the diffusion model for behavior cloning of teleoperation demonstrations of the Box and Blocks task. Compared to many other tasks, the Box and Blocks task is visually complicated because at any given time, there are more than 40 similar blocks from which the model must target a viable grasping candidate.

One reason diffusion models were used for this work was to see if the successes experienced by Chi et al. transfer to multi-fingered hands and more complex manipulation tasks [8]. In the future, this abstract's results can be compared to alternative methods.

II. System Overview

DexNex is a teleoperation system with an Operator and Avatar. The Operator is a human teleoperator outfitted with two HaptX DK2 gloves and a Varjo Aero virtual reality (VR) headset.

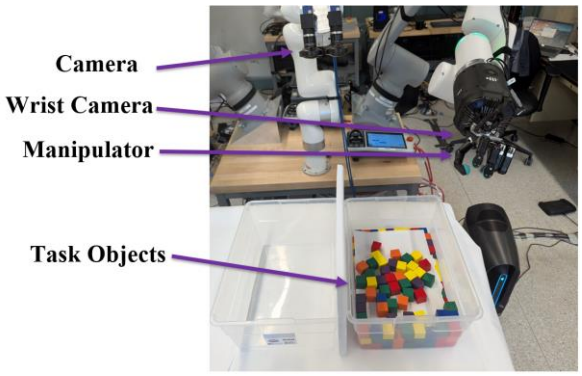

Figure 3. The Box and Blocks task setup. The overhead and wrist cameras provide the images. The manipulator is controlled by either a teleoperator during demonstrations or the diffusion model during evaluation. The Box and Blocks task has the participant transferring as many blocks, one at a time, from one bin to another in 60 seconds.

The Avatar is an upper-limb humanoid robot. It consists of two robot arms (ABB GoFa), two robot hands (Shadow Dexterous Hand), ten robot fingertips (SynTouch BioTac SP) one robot neck (UFACTORY xArm6), and two 4k RGB cameras for vision (FLIR Blackfly). There are 66 joints and 58 degrees of freedom (DoF), as each non-thumb digit of each Shadow hand has an unactuated distal joint.

The Operator's head, hand, and finger motions are tracked in Cartesian space and the Avatar's corresponding robot arms track the Operator's motion. The Avatar generates visual and tactile feedback which are displayed on the Operator's VR headset and HaptX Gloves, respectively. For this work, however, the VR headset was not used.

A. *DexNex v0.1*

Prior teleoperation results used DexNex version 0.1, which focused on mimicking the Operator's motions as closely as possible with the Avatar [1]. No assistive features were provided.

B. *DexNex v0.2*

The work in this abstract used DexNex version 0.2 which provides some low-level assistive features designed to overcome some of the difficulties of teleoperating with HaptX gloves. For example, HaptX glove thimbles make it difficult for the teleoperator to press their fingers together, so one assistive feature reduces the distance between the fingertips of the robot hands relative to those of the HaptX gloves.

III. Approach

A. *Teleoperation*

Demonstrations were collected by teleoperating DexNex (Figure 2). To restrict the number of Avatar degrees of freedom and the number of sensory channels, the Avatar neck was locked into place and its left "eye" camera was used to provide an overhead view of the workspace. Additionally, a wrist camera was added to the forearm of the left Shadow Hand. The teleoperator had a direct view of the working area, along with a view of a monitor playing the video feed from the two cameras. The teleoperator was an experienced user of the system with over 50 hours of training.

The Operator controlled the Avatar's left arm and parts of the left hand. To reduce potential collisions, only the wrist, thumb, first finger, and middle finger were controlled while the ring finger and pinky were locked in a curled position.

From this, the number of output joint position commands that the teleoperation software produced was 21 (six GoFa, two wrist, five thumb, four first finger, four middle finger).

Over the course of collecting demonstrations, the Operator's performance improved by adopting a better strategy for block separation and pickup. Depending on the distribution of blocks in the box, the teleoperator executed either a "scooping" strategy, which served to separate adjacent blocks as well as trap a single block between the robot fingers, or a "pecking" strategy, which was used to grasp already separated blocks in the shortest time.

B. *Data Processing*

Demonstrations were collected as ROS2 bags and post-processed into Zarr datasets consisting of the joint positions, fingertip cartesian positions, one tactile value per fingertip, the left "eye" camera, the wrist camera, and joint position commands. The datapoints were time-synchronized, and saved at 100 ms intervals, resulting in 10 Hz data.

Three datasets were produced through the demonstrations. The first dataset was obtained from a simpler block-transfer task where just one block was picked from one plate and placed on another. From this, 54,390 data-points were generated.

The second dataset was obtained from the Box and Blocks task using only the overhead camera video, as it was unknown at the time whether a wrist camera was needed to improve performance. 101,846 data-points were generated.

Once it was clear that performance would improve with an additional perspective from a wrist camera, one was added, and further Box and Blocks demonstrations were performed. The third dataset produced 160,487 datapoints.

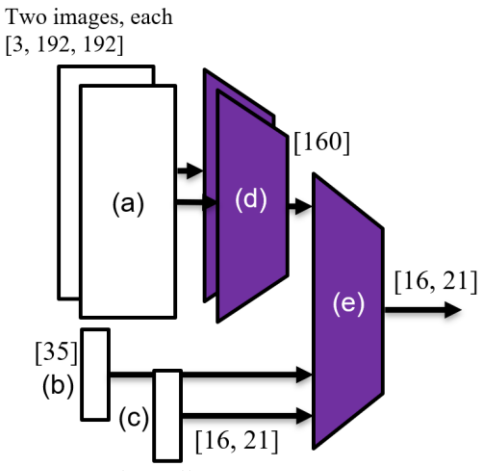

Figure 2. An outline of the diffusion model. (a) Two images are input to the visual encoder. (b) The state (joint positions, haptics, and fingertip positions), are input to the UNet. (c) actions (waypoints of joint positions) are input to the UNet. (d) The images are passed through separate ResNet-18 visual encoders. (e) A UNet outputs the predicted noise.

Table 1. Model and training parameters for the diffusion model. For a full explanation of parameters, refer to [8]. D-Params and V-Params are listed in millions. V-Features: number of output values from the visual encoder.

Model & Training Parameters	Parameter Name										
	Overhead Camera Resolution	*Wrist Camera Resolution*	*State Input Length*	*Horizon*	*Nb Obs Steps*	*D-Params*	*V-Params*	*V-Features*	*Batch Size*	*D-Iters Train*	*D-Iters Eval*
Value	3x192x192	3x192x192	35	16	1	276	22	160	64	100	100

C. Machine Learning Policy

Chi et al. open-sourced their software on GitHub which provides several diffusion models ready to be used [8]. Our work uses their hybrid architecture, consisting of both state and images as inputs (Figure 3). The input images are fed through a visual encoder, ResNet-18, to produce image features [10]. A noise level from one to 100 is then randomly generated. Noisy actions are calculated from the input actions and noise level. The image features, along with the state, noisy actions, and noise level are fed through a 1d CNN-based UNet [11] which outputs the predicted noise level present in the noisy actions.

The image inputs were a single RGB overhead camera view, cropped and resized to 192x192 pixels, and the RGB wrist camera, resized to 192x192 pixels.

Most default settings for the diffusion model were kept, with some customization to the training parameters. Details of the training parameters are provided in Table 1. No pre-trained network weights were used.

When using the policy during real-time control, if the execution speed differs from the demonstration speed, there is a risk of experiencing "distributional shift". That is, the input data may differ from what the diffusion model was trained on which may lead to suboptimal output trajectories. To avoid this issue, the policy is restricted to planning given the current state and image only (nb_obs_steps: 1).

D. Co-Training

Co-training is when a policy is trained on a combination of data relevant to the problem at hand along with other data. Prior work has shown that co-training improves the performance of agents when problem-specific data is limited, as is often the case when doing real-world manipulation tasks [12].

Co-training was implemented by assembling each training batch with samples from the three available datasets. Each batch consisted of 25% dataset one, 25% dataset two, and 50% dataset three.

Table 2. Time per block-transfer in the Box and Blocks task for direct human manipulation (natural body), an expert teleoperator, and the diffusion policy. The diffusion policy is far slower due to several failure modes the policy exhibits, as well as slowed down trajectory execution. Success rate was determined by whether at-least 1 block was transferred before a trial ended.

Box and Blocks	Method		
	Natural Body	*Expert Teleoperator*	*Diffusion Policy*
Time Per Block (s)	0.9 (SD=0.13)	6.7 (SD=0.75)	60.2 (SD=32.6)
Success Rate	100%	100%	80%

E. Trajectory Execution

During evaluation, the diffusion policy outputs a 16-waypoint-long trajectory. Each waypoint of the trajectory contains a list of joint positional commands, absent of timestamps. A trajectory timer then uses a timing parameter (dt) and the distance problem formula ($d = rt$) to determine the average speed of each trajectory segment. If the speed is greater than a max velocity, then that segment is slowed down accordingly. The resultant dt is used to calculate each waypoint's timestamp. By changing the dt value, trajectory execution can be sped up or slowed down.

The final trajectory is sent to a ROS2-Control Joint Trajectory Controller which ingests the trajectory and outputs joint commands at the appropriate time. The joint trajectory controller also ensures that all joints reach the same waypoint at the same time, enforcing synchronization of joint positions between arm and hand.

The joint trajectory controller is set to position-only mode which does a simple time-based linear interpolation between waypoints. Position mode produces continuous positions but discontinuous velocities (and further derivatives).

The output joint positional commands are passed through a low-pass-filter per-joint to smooth the discontinuous velocities experienced when transitioning between line segments in the position-mode joint trajectory controller.

The final joint position commands are passed to the respective hardware drivers for each component.

IV. Results

Table 2 shows our preliminary results. The natural body results were computed from 8 human participants, each performing just one trial. The expert teleoperator did 10 trials of the standard Box and Blocks task from which the average time per block transfer and standard deviation were computed.

The diffusion policy was co-trained for 195 epochs, where each epoch consisted of 1000 gradient descent steps, and checkpoints of the policy weights were saved every 15 epochs. Because policy behavior can vary throughout training, three checkpoints, from epochs 75, 105, and 195, were tested and qualitatively evaluated. The best performing checkpoint was then tested over 10 trials with a slightly modified task end-condition: when the policy failed to grasp a block three times in a row, the trial was ended.

Success was defined as any trial which ended with at-least one block transferred. The diffusion policy's success

rate was 80%, while for natural body and expert teleoperator it was 100%.

From only the successful trials, the diffusion policy's average time per block transfer and standard deviation were computed.

On average, the diffusion policy transfers one block every 60.2 seconds. This is 9x slower than the expert teleoperator that the policy was trained to imitate.

Qualitatively, observations were made about the policy's performance and how to improve it. Behavior varied significantly depending on which checkpoint was evaluated.

Initial checkpoints were dangerous to execute as they produced non-smooth trajectories.

Apart from that, earlier checkpoints showed the ability to pick high potential blocks but were worse at generating smooth trajectories and worse at successfully finishing grasps of blocks.

Later checkpoints exhibited very smooth trajectories but were poor at the overall strategy of picking blocks. For each grasp attempt, it would target a location nearer to the center of the box (i.e., the average of all pick locations) and would attempt to pick non-existent blocks at that spot repeatedly.

This is indicative of overfitting to the data, since the policy had low training loss but could not generalize to unseen states (i.e., states seen during evaluation).

V. Discussion

This work presented the development of a first attempt at an autonomous diffusion policy tasked with replicating teleoperation behavior to accomplish the Box and Blocks task. The diffusion policy has a decent success rate and block-transfer rate, but further work is needed to improve policy performance. The authors believe performance comparable and even surpassing teleoperation is possible. To realize this, advancements must be made to the system.

One reason for the expert teleoperator vs. diffusion policy performance gap is due to the teleoperator's demonstration instructions and endurance. During demonstrations, they were instructed to minimize mistakes by moving slowly and deliberately. In addition, they could easily move at full speed during their trials, but when tasked with collecting demonstrations for upwards of 60 minutes per session, they quickly became tired. So, the average movement speed in the training data is far slower than the expert teleoperator's trials.

Teleoperator fatigue may be addressed by training several users and swapping between them during demonstration data collection sessions.

To improve policy intelligence, first, more varied demonstration data spanning a larger space of the policy's distribution must be collected from which to learn desirable behavior and recovery behaviors.

Second, different and larger neural net architectures can be experimented with. For instance, the default architecture does not have a persistent memory, only an optional history of previous observations. Incorporating a form of memory like a recurrent neural network may yield more intelligent behavior. For tasks with high visual complexity like Box and Blocks, more advanced visual encoders which preserve more spatial information may also be beneficial.

Third, co-fine-tuning with open-source robotic datasets, such as the Open X-Embodiment, should be incorporated as it has been shown to improve task performance [6, 13].

Lastly, since performance is directly correlated with average hardware velocity, if we assume quasi-static dynamics then simply running faster will lead to better scores. A more advanced online trajectory generator (OTG) is needed for quicker trajectory execution. During run-time, the robot is non-stationary when it receives a new trajectory. To maximize velocity, a smooth transition is needed from old to new trajectories. This problem can be cast as a series of two-point boundary-value-problem (BVP) where we specify the start and end position and velocity. Potential solutions are cubic polynomial fitting, time-optimal-trajectory-generation, or using a BVP solver like MATLAB's *bvp4c* [14].

As the HAND Engineering Research Center evolves, we are hopeful that this work and additional research will reveal insights into which hardware and software are most effective at autonomous dexterous manipulation.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 2330040. The authors thank Russ Tedrake, Adam Wei, and Davin Landry for help and fruitful discussions.

References

[1] Toby Buckley, J. Edward Colgate. "Dexterous Manipulation with a Bi-Manual Anthropomorphic Teleoperation Robot", 3rd Workshop Toward Robot Avatars, IROS 2024

[2] Mathiowetz V, Volland G, Kashman N, Weber K. Adult norms for the Box and Block Test of manual dexterity. The American journal of occupational therapy. 1985 Jun 1;39(6):386-91.

[3] Irene A. Kuling, Kaj Gijsbertse, Bouke N. Krom, Kees J. van Teeffelen, and Jan B. F. van Erp. "Haptic feedback in a teleoperated box & blocks task," in Ilana Nisky, Jess Hartcher-O'Brien, Micha¨el Wiertlewski, and Jeroen Smeets, editors, Haptics: Science, Technology, Applications, pages 96–104, Cham, 2020. Springer International Publishing

[4] Jeremy A. Fishel, Toni Oliver, Michael Eichermueller, Giuseppe Barbieri, Ethan Fowler, Toivo Hartikainen, Luke Moss, and Rich Walker. "Tactile telerobots for dull, dirty, dangerous, and inaccessible tasks," in 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 11305–11310, 2020.

[5] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bi-manual manipulation with low-cost hardware. RSS, 2023

[6] Brohan A, Brown N, Carbajal J, et al. RT-2: vision-language-action models transfer web knowledge to robotic control. In: Proceedings of The 7th Conference on Robot Learning. 2023. p. 2165–2183

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In Advances in Neural Information Processing Systems, Vol. 33. 6840–6851.

[8] Chi C, Xu Z, Feng S, et al. Diffusion policy: Visuomotor policy learning via action diffusion. The International Journal of Robotics Research. 2024;0(0). doi:10.1177/02783649241273668

[9] Ze, Yanjie & Zhang, Gu & Zhang, Kangning & Hu, Chenyuan & Wang, Muhan & Xu, Huazhe. (2024). 3D Diffusion Policy: Generalizable Visuomotor Policy Learning via Simple 3D Representations. 10.15607/RSS.2024.XX.067.

[10] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.

[11] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science(), vol 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

[12] Wei, Adam, et al. "Empirical Analysis of Sim-and-Real Cotraining Of Diffusion Policies For Planar Pushing from Pixels." arXiv preprint arXiv:2503.22634 (2025).

[13] A. O'Neill et al., "Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration0," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 6892-6903, doi: 10.1109/ICRA57147.2024.10611477.

[14] Kunz, Tobias, and Mike Stilman. "Time-optimal trajectory generation for path following with bounded acceleration and velocity." Robotics: Science and Systems VIII (2012): 1-8.