2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Switchable Neural Teleoperation

Jianglong Ye*, Changwei Jing*, Kezhou Chen, Keyi Wang, Sha Yi, Xueyan Zou, Xiaolong Wang
UC San Diego
https://jianglongye.com/neural-teleop

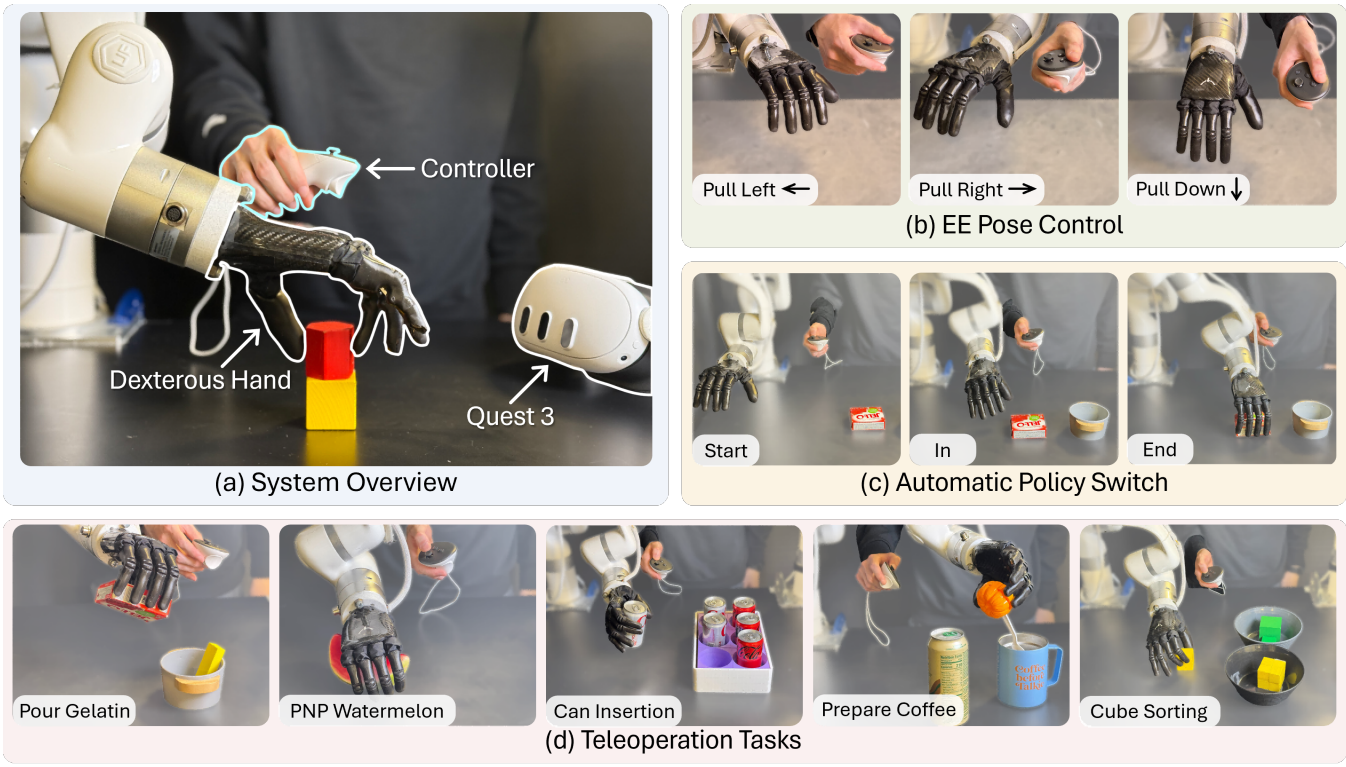

Fig. 1: **Neural Teleoperation System.** (a) Overview of our shared-autonomy teleoperation setup. (b) End-effector pose is controlled by relative wrist motion. (c) Our switcher detects to transfer from the teleoperator to the autonomous grasping policy. (d) Real-world teleoperation tasks.

***Abstract*— Collecting demonstrations through human teleoperation is an effective approach for learning complex manipulation skills. However, challenges such as morphology gaps, control latency, and limited feedback make high-quality data collection costly and inefficient. In this paper, we introduce Neural Teleoperation, a shared-autonomy system that integrates human guidance with a robust grasping policy using a learning-based policy switcher. This hybrid framework allows users to focus on high-level planning while delegating fine-grained control to an autonomous policy when needed. Our system supports both immersive VR devices and lightweight 6-DoF controllers, making dexterous hand teleoperation more accessible. Real-world experiments across six manipulation tasks show that Neural Teleop increases success rates and reduces demonstration collection time compared to state-of-the-art baselines.**

I. INTRODUCTION

Collecting demonstrations via human teleoperation and then distilling them into autonomous policies has been a widely adopted paradigm for learning complex manipulation skills [1]–[3]. Yet in practice the quality–efficiency trade-off of demonstration data remains a persistent bottleneck, owing to *morphology gaps* between human hands and robotic effectors, network or computation–induced *control latency*, and the inherently *limited perceptual feedback* available to the operator. These factors make each minute of high-fidelity data painfully expensive to acquire.

Between the two extremes of pure teleoperation and fully autonomous control lies a fertile middle ground: a **shared-autonomy** regime in which a partially trained policy already masters low-level dexterous motions, while the human still provides sparse, high-level guidance. Leveraging this complementarity, we adopt a *hybrid data-collection loop*: the operator quickly navigates coarse, long-horizon portions of a task, then hands control to a neural policy to execute precise finger motions. Conversely, the policy can relinquish control to the human whenever it encounters an unfamiliar

* Equal contribution.

state. Compared with either component alone, this division of labor accelerates data collection and maximizes the useful information density of each demonstration.

To decide *when* to delegate control, we introduce a **learning-based switcher**. Trained on proprioceptive and visual observations, the switcher predicts in real time whether the robot should remain in teleoperation mode or invoke the autonomous policy. This adaptive gating eliminates manual mode-toggling, reduces cognitive load on the operator, and ensures smooth transitions that preserve task success. The overview of our system is presented in Fig. 1.

While immersive hand-tracking devices (e.g., Vision Pro) provide rich inputs, they are still bulky and costly. We therefore extend our framework to *commodity 6-DoF motion controllers*. When only the end-effector pose is available, our grasping policy infers the missing finger configuration, allowing the operator to teleoperate complex dexterous tasks with nothing more than a lightweight controller, providing better user experience and greater collection speed.

We benchmark three settings: *Pure Teleoperation*, *Neural Policy via Hand-Retargeting*, and *Neural Policy via Controller*, on six challenging real-world manipulation tasks. The proposed **Neural Teleop** system improves success rates by up to 33% ($60 \rightarrow 93\%$ on *PNP Watermelon*) and reduces the demonstration time by more than 70 % (16.0 min $\rightarrow$ 4.6 min) compared with a state-of-the-art Vision-Pro teleoperation baseline. We claim the following contributions:

1) **Neural Teleoperation**: a shared-autonomy framework that interleaves human operation with dexterous neural policy via a learned switcher for fast, high-quality demonstration collection.
2) **Versatile Interfaces**: a single system that supports full hand-pose VR devices and low-cost 6-DoF motion controllers, broadening accessibility without sacrificing dexterity.
3) **Comprehensive Evaluation**: real-robot experiments on six tasks demonstrate higher success rates and up to $3\times$ faster data collection than prior teleoperation systems.

II. Related Work

Dexterous Manipulation. Dexterous manipulation aims for precise, multi-finger control in grasping and in-hand adjustment. Early control-based methods established caging and grasp principles [4], [5] and analyzed under-actuated hand manipulability [6]. Later work pruned the grasp search space and refined optimization [7]–[10], producing accurate yet task-specific controllers. To enhance generality, learning-based approaches emerged, directly regressing hand poses [11]–[15], incorporating geometric cues [16]–[18], or using contact maps [19], [20]. Our work employs a CVAE to generate diverse grasp poses conditioned on objects, and can further produce hand poses conditioned on object points, facilitating seamless teleoperation integration across objects and approach directions.

Shared Autonomy. Our work also intersects with shared-autonomy studies [21]–[25], which use external guidance to shape robot actions. Many approaches embed teleoperation data into the observation space during training [26]–[28]; we do not rely on human inputs at that stage. Other methods frame assistance as inferring a small set of task-specific goals [21], [29], but this coarse abstraction limits fine-finger dexterity. Our system treats the teleoperator's hand pose as a reference. When switching to the grasping policy, it samples multiple grasping poses and selects the one closest to the teleoperator's current hand pose as the deployed hand pose.

Dexterous Teleoperation. Dexterous hand teleoperation is inherently challenging because of their high degrees of freedom and intricate kinematic structure. Current approaches generally fall into two main categories, with one relying on motion capture gloves, which can accurately capture finger movements [30], [31], but they tend to be expensive and often restricted to specific hand sizes. Another vision-based method like AnyTeleop [32] and BunnyVisionPro [3] have enabled dexterous hand-arm teleoperation using camera systems [33], [34] or VR headsets [35]–[38]. However, capturing hand joint values (qpos) through VR/camera remains limited by the accuracy of vision-based retargeting and visual occlusions, making it difficult to perform highly precise teleoperation tasks.

III. Method

A. Overview

Our Neural Teleoperation system has three main parts: (1) a teleoperation system, (2) an autonomous grasping policy, and (3) a learning-based switcher. These components work together to combine human control and autonomous actions in a shared-autonomy setting.

When the teleoperator navigates the task, the teleoperation interface provides real-time control of a hand-arm robot using either VR devices which capture full wrist and finger poses, or simpler controllers that provide only end-effector poses. Meanwhile, a learning-based switcher monitors the robot's proprioception and visual observations to determine when to activate the autonomous grasping policy. Once the switcher decides to switch, the grasping policy takes over to execute the grasp. After the grasp is complete, control returns to the teleoperator to finish the manipulation task. The overview of our method is presented in Fig. 2.

B. Teleoperation System

Arm Control. For both VR and controller input devices, we estimate the human wrist pose and convert it into a target end-effector pose for the robot arm. Specifically, we initialize teleoperation while the robot arm remains in its predefined initial pose. When the human wrist orientation closely aligns with that of the robot end-effector, we record the current wrist pose as the initialized wrist pose. The relative transformation between the live wrist pose and this reference pose is then used to compute the desired relative pose of the robot end-effector with respect to its initialized configuration. We solve the inverse kinematics problem as a quadratic program using the Pink library [39], minimizing the following objective:

$$\min_{v} \ \|J_e(q)v + \alpha e(q)\|^2 + \lambda \|v\|^2, \tag{1}$$

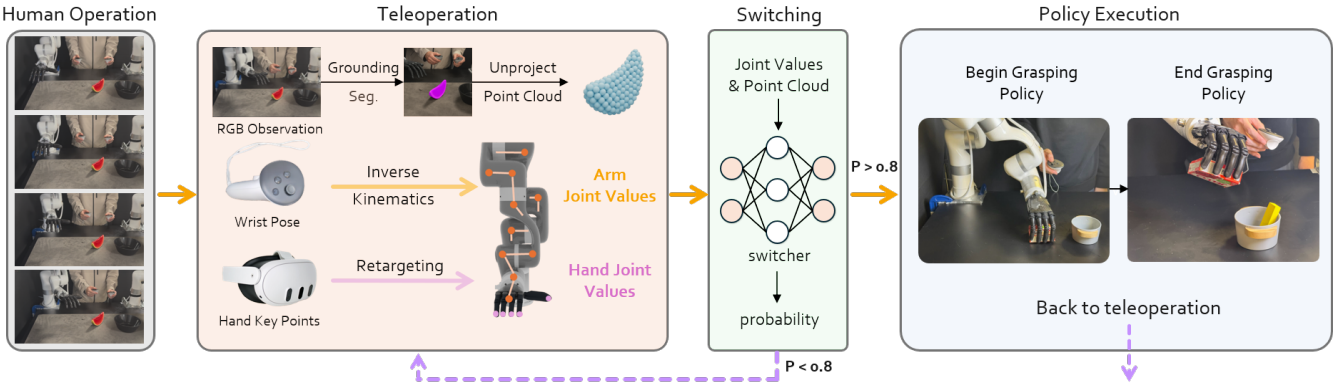

Fig. 2: **Overview.** Our Neural Teleoperation system combines human teleoperation, a learning-based switcher, and an autonomous grasping policy in a shared-autonomy loop. The switcher monitors visual and proprioceptive inputs to trigger autonomous grasping, then hands control back to the human to complete the task.

where

$$e(q) = \log(T_{\text{bt}}), \quad \text{with } T_{\text{bt}} = T_{\text{b}}^{-1} T_{\text{t}}, \tag{2}$$

and T_{b} and T_{t} denote the current and target end-effector poses, respectively. The Jacobian $J_e(q)$ is computed using Pinocchio [40]. Here, α is a tracking gain, and λ is a damping coefficient that regularizes motion for improved stability. The resulting joint velocity v is integrated over time to obtain the target joint positions.

Hand Control. When using the Apple Vision Pro as the input device, our system supports controlling the robot hand through retargeting. Specifically, human hand keypoints are translated into robot joint angles using dex-retargeting [32], a fast hand retargeting library. Our method employs vector optimizers that formulate retargeting as an optimization problem, minimizing the distance between user-defined vectors computed from both the human and robot hands [32], [34]:

$$\min_{q_t} \sum_{i=0}^{N} \|\alpha v_t^i - f^i(q_t)\|^2 + \beta \|q_t - q_{t-1}\|^2. \tag{3}$$

Here, v_t^i is the i-th keypoint vector in the human hand's wrist coordinate frame, and q_t denotes the robot's joint angles at time t. The function $f^i(q_t)$ computes the corresponding keypoint vector on the robot hand via forward kinematics. The parameter α adjusts for the scale mismatch between the human and robot hands, and β is a regularization weight that encourages temporal smoothness.

Note that hand retargeting is not performed during policy execution; it is only applied before or after the policy and is primarily used to achieve a better pre-grasp pose. After the grasping policy completes, the robot hand maintains the grasping configuration while the arm follows the human's movement until release. After the object is released at the target location, the teleoperator resumes control to continue the task.

C. Neural Switcher

The goal of the learning-based switcher is to determine the optimal moment to transition from human teleoperation to the autonomous grasping policy. A naive approach based solely on the distance between the end-effector and the target object is insufficient, as task success also depends on achieving a **good pre-grasp configuration** [41], [42]. We illustrate effective switching timing with a suitable pre-grasp pose in Fig. 3. Therefore, the switcher must be conditioned on both the current hand pose and the object geometry to make reliable decisions.

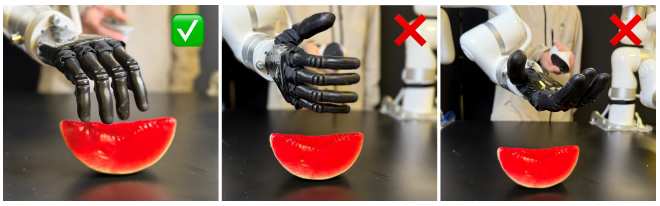

Fig. 3: **Motivation of Neural Switcher.** The switcher must determine the right moment to activate the grasping policy. Only the left pose provides a suitable pre-grasp configuration — the other two do not.

We take as input an RGBD image and text prompts specifying target objects (e.g., *coffee*, *stir*). To localize the target, we first apply Grounding DINO [43] to detect the object in the RGB image based on the provided text prompt. The resulting bounding box is then refined and tracked in real time using SAM2 [44], enabling consistent segmentation across frames. Using the aligned depth map, we unproject the segmented region into 3D space to obtain a point cloud $P \in \mathbb{R}^{N\times 3}$, which captures the geometry of the grounded object. We employ PointNet [45] to encode the point cloud into a global feature vector $f_{\texttt{obj}} \in \mathbb{R}^d$:

$$f_{\texttt{obj}} = \textbf{PointNet}(P). \tag{4}$$

The object geometry feature $f_{\texttt{obj}}$, along with the robot's proprioceptive state (i.e., joint positions) q, is passed into a multi-layer perceptron (MLP). The final output is a frame-wise switching probability:

$$p = \sigma\left(\textbf{MLP}_{\texttt{head}}([f_{\texttt{obj}}, q])\right), \tag{5}$$

where $\sigma(\cdot)$ denotes the sigmoid activation, and $p \in [0,1]$ indicates the predicted likelihood of switching from teleoperation to the autonomous grasping policy at the current frame. The

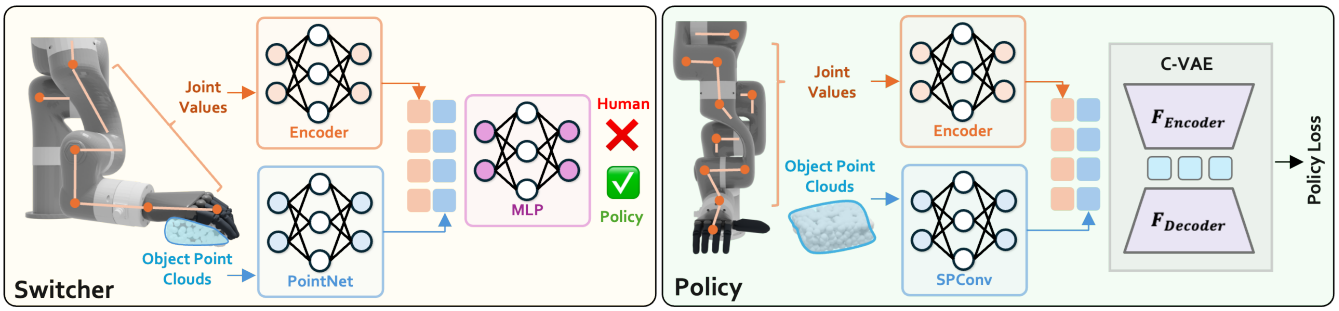

Fig. 4: **Network Architecture.** The switcher (left) predicts when to transition from teleoperation to the autonomous policy using joint states and object point clouds as inputs. The grasping policy (right) uses a CVAE conditioned on hand pose and object geometry to sample grasp poses.

architecture of the neural switcher is illustrated on the left side of Fig. 4.

To provide effective supervision, we assign a probability of 1.0 to the ground-truth switching frame. This probability is linearly decayed to 0.5 for frames within ± 10 frames of the ground truth, and further to 0.0 for frames beyond ± 25 frames. This soft labeling strategy encourages the model to learn a temporally smooth confidence distribution centered around the ground-truth switch point. Since our teleoperation sequences are recorded at approximately 30 frames per second, this decay window corresponds to a $\sim$1.7-second interval. During inference, we select the first frame with a predicted probability greater than 0.9 as the switching point. The policy is trained using the standard binary cross-entropy (BCE) loss between the predicted switching probability p and the soft labels described above. We note that a high inference threshold (0.9) reduces false positives where the switcher might activate while the operator is only hovering near the object.

D. Grasping Policy

Our grasping policy is designed to be (1) robust to diverse object geometries, (2) efficient for real-time switching in shared-autonomy settings, and (3) tolerant of varying initial hand poses during transition from human control. To meet these goals, we adopt a lightweight conditional variational autoencoder (CVAE) model with a point cloud vision encoder. The CVAE is efficient enough for real-time inference, enabling rapid switching from teleoperation to autonomous control. Moreover, as a generative model, it supports sampling multiple grasp candidates conditioned on the current hand pose and visual input, allowing for flexible and adaptive grasp execution. We collect a large-scale set of object assets from multiple datasets [46] and generate approximately 300 million demonstrations using grasp optimization [46], motion planning, and simulation [47], [48]. The diversity of object geometries and hand-object interactions in this dataset helps the policy generalize well to novel environments. The policy is trained on these simulated demonstrations and directly deployed in real-world settings.

We use a segmented depth map to generate a point cloud $P \in \mathbb{R}^{N\times 3}$ as the visual input. We employ SPConv [49] to encode the point cloud into local feature vectors $f_p \in \mathbb{R}^d$ for each object point $p \in \mathbb{R}^3$:

$$\{f_p\}_{p\in P} = \textbf{SPConv}(P).$$

The CVAE uses a multi-layer perceptron (MLP) to encode the hand pose q into the mean and standard deviation vectors of a latent distribution. A sample from this distribution is passed through an MLP decoder to reconstruct the original hand pose. We concatenate the point cloud feature vector f_{p} with both the encoder and decoder inputs to condition the CVAE on visual observations:

$$\begin{aligned} \mu, \sigma &= \textbf{Enc}(q, f_{\mathrm{p}}), \quad z = \mu + \sigma \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \\ \hat{q} &= \textbf{Dec}(z, f_{\mathrm{p}}). \end{aligned} \tag{6}$$

The CVAE training is supervised by a weighted sum of the standard reconstruction loss $\mathscr{L}_{\mathrm{R}} = \|q - \hat{q}\|_2^2$ and the KL divergence loss $\mathscr{L}_{\mathrm{KL}} = D_{\mathrm{KL}}\Big(\mathcal{N}(\mu, \sigma^2) \Big\| \mathcal{N}(0, I)\Big)$. The architecture of the grasping policy is illustrated on the right side of Fig. 4.

IV. Experiments

We evaluate our Neural Teleoperation system through real-robot teleoperation experiments, comparing it against the advanced teleoperation system BunnyVisionPro [3] across six dexterous hand tasks. The main evaluation results are reported in Sec. IV-A. We further evaluate the individual learning-based modules in our system — the switcher and the grasping policy — and present their results in Sec.IV-B and Sec.IV-C, respectively. System profiling analysis is provided in Sec. IV-E. Please refer to the supplementary materials for the imitation learning results.

A. Teleoperation Results

Real-world Robot Setup. Our teleoperation system consists of a UFactory xArm-7 robotic arm equipped with a 6-DoF Ability Hand. A RealSense L515 camera is positioned in front of the robot's workspace for visual observation collection. The camera pose is calibrated using hand-eye calibration. For input devices, we use the Apple Vision Pro to capture both wrist and finger poses, and the Meta Quest 3 controller to capture 6-DoF end-effector poses.

Baselines. We mainly compare our method against BunnyVisionPro [3], a recent real-time dexterous teleoperation system that leverages the Apple Vision Pro for accurate hand

Task	Description	System	Input Device	Policy Switch	Success (↑)	Time (↓)	EpLen (↓)
			Short-horizon Tasks				
PNP Watermelon	Pick watermelon and place in box.	Bunny-VisionPro	VR	-	9 / 15	15.98	479±68
		NeuralTeleop	VR	Manual	14 / 15	5.50	165±22
		NeuralTeleop	VR	Neural	15 / 15	**4.61**	**138±11**
		NeuralTeleop	Controller	Neural	**15 / 15**	7.57	227±13
Stack Prism	Stack a hexagonal prism on top of a cube.	Bunny-VisionPro	VR	-	10 / 15	9.93	297±58
		NeuralTeleop	VR	Manual	12 / 15	**6.09**	**183±29**
		NeuralTeleop	VR	Neural	13 / 15	7.03	211±15
		NeuralTeleop	Controller	Neural	**14 / 15**	8.72	261±33
Pour Gelatin	Grasp the gelatin then pour it into the bowl	Bunny-VisionPro	VR	-	11 / 15	9.08	272±59
		NeuralTeleop	VR	Manual	11 / 15	**6.14**	**184±31**
		NeuralTeleop	VR	Neural	12 / 15	8.11	243±38
		NeuralTeleop	Controller	Neural	**12 / 15**	9.67	290±39
			Long-horizon Tasks				
Prepare Coffee	Pour the coffee into the cup, place it down, and stir with a spoon.	Bunny-VisionPro	VR	-	11 / 15	43.05	1291±358
		NeuralTeleop	VR	Manual	**12 / 15**	**26.89**	**806±120**
		NeuralTeleop	VR	Neural	11 / 15	28.82	864±134
		NeuralTeleop	Controller	Neural	11 / 15	27.79	834±125
Cube Sorting	Pick up the cubes and toss them into the corresponding cases.	Bunny-VisionPro	VR	-	14 / 15	57.26	1717±299
		NeuralTeleop	VR	Manual	15 / 15	26.47	794±59
		NeuralTeleop	VR	Neural	15 / 15	27.02	811±65
		NeuralTeleop	Controller	Neural	**15 / 15**	**26.09**	**783±54**
Can Insertion	Pick up the cans and insert them into the container.	Bunny-VisionPro	VR	-	15 / 15	99.95	2998±458
		NeuralTeleop	VR	Manual	15 / 15	73.24	2209±315
		NeuralTeleop	VR	Neural	15 / 15	78.23	2360±344
		NeuralTeleop	Controller	Neural	**15 / 15**	**71.01**	**2129±294**

TABLE I: **Teleoperation System Evaluation.** We report the collection success rate (Success), episode length (EpLen), and average collection time in seconds (Time) across all settings. Neural Teleop consistently outperforms Bunny-VisionPro across all tasks, demonstrating the benefits of integrating the grasping policy for both precise grasping and improved efficiency. We further show that Neural Teleop can operate without manual switching (using our neural switcher) and without finger tracking (via a controller-based interface).

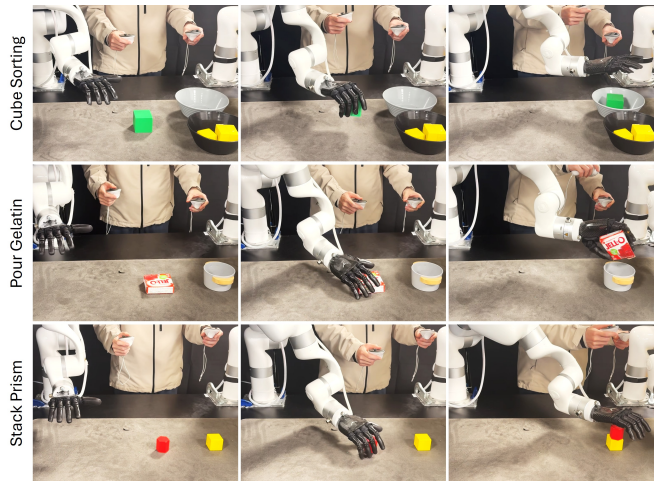

Fig. 5: Visualization of Teleoperation Episodes.

tracking. Its design emphasizes both safety and responsiveness, making it a strong baseline for real-world dexterous teleoperation and imitation learning tasks. For simplicity, we omit the low-cost haptic feedback devices introduced in their paper.

Main Results. Tab. I summarizes the real-world teleoperation performance across six manipulation tasks. We report three metrics: collection success rate (*Success*), average collection time in seconds (*Time*), and episode length (*EpLen*), where lower time and episode length indicate more efficient teleoperation. Both collection time and episode length are calculated over successful episodes only. We indicate *Input Device* in Tab. I: "VR" refers to using the Apple Vision Pro for capturing both wrist and finger poses, while "Controller" refers to using the Meta Quest 3 controller for capturing 6-DoF poses. While both Bunny-VisionPro and our system support Apple Vision Pro, Bunny-VisionPro cannot use the Quest 3 controller because it requires finger positions to retarget to robot hand poses. In contrast, our system flexibly supports both input devices by utilizing a learned policy to predict hand poses, allowing it to operate even without finger poses tracking.

We evaluate our proposed system under three different settings: (1) *VR + Manual Policy Switch*, where the operator manually switches between teleoperation and policy control; (2) *VR + Learned Policy Switch*, where our learned switcher autonomously determines when to delegate control; and (3) *Controller + Learned Policy Switch*, where the end-effector

Fully Autonomous							
Method	Straw-berry	Mango	Rubik Cube	Sauce Can	Tooth-paste Box	Tennis Ball	Mean
DexSampler	3/5	4/5	3/5	**3/5**	2/5	5/5	66.67%
Ours	**5/5**	**4/5**	**4/5**	2/5	**3/5**	**5/5**	**76.67%**
Shared Autonomous (Neural Teleoperation - Pick & Place)							
Method	Straw-berry	Mango	Rubik Cube	Sauce Can	Tooth-paste Box	Tennis Ball	Mean
DexSampler	3/5	5/5	2/5	3/5	3/5	3/5	63.3%
Ours	**4/5**	**5/5**	**5/5**	**5/5**	**3/5**	**5/5**	**90%**

TABLE II: **Grasping Policy Evaluation.** Success rate out of 5 trials across different objects and autonomy settings.

Fig. 6: Grasping Objects.

Setting	Abs Frame Error (↓)	Abs Time Error (↓)
Default (Ours)	5.67	0.16
Cam Coord PC	7.56	0.27
Mean Point	16.44	0.64
EE Pose	16.11	0.58

TABLE III: **Neural Switcher Evaluation.** Absolute frame and time errors under various settings.

is controlled via the Quest 3 controller and switching is fully automatic. These settings allow us to evaluate the performance trade-offs between human involvement and policy automation across different device setups.

Across all tasks, Neural Teleop consistently outperforms Bunny-VisionPro. In short-horizon tasks like *PNP Watermelon* and *Stack Prism*, using manual switching with Neural Teleop leads to higher success rates and much faster collection, showing the benefit of adding a robust policy for precise grasping. This benefit is even clearer in long-horizon tasks: while success rates are similar due to easier grasps, Neural Teleop cuts collection time by over 40% in all three tasks. The results also demonstrate the benefits of learning-based policy switching and flexible input devices. Compared to manual switching, our neural switcher achieves comparable success rates and collection efficiency, showing that learned switching can reliably replace manual control. Additionally, our controller-based interface delivers competitive performance despite lacking finger tracking, making teleoperation more accessible. We also provide visualizations of successful episodes in Fig. 5.

B. Neural Switcher Evaluation

The neural switcher predicts whether to activate the grasping policy or stay in teleoperation mode, based on the robot's joint positions and the segmented point cloud of the target object. We train it using 180 teleoperation episodes with manual switching by a human operator, using 80% of the data for training and 20% for evaluation. For evaluation, we identify the frame with the highest predicted switching probability and compare it to the ground-truth switching frame. The absolute frame/time difference serves as our metric. With recordings at around 30 fps, each frame corresponds to about 30 ms. Smaller differences indicate more accurate switch timing, which is crucial for smooth transitions.

We ablate different design choices for the switcher and report the results in Tab. III. The default model, which uses both the robot's joint positions and the object's point cloud (transformed into the robot coordinate frame), achieves the best accuracy. Using point clouds in the camera coordinate frame (Cam Coord PC) leads to slightly worse performance, indicating that transforming the point cloud into the robot's frame helps the model reason about spatial relationships more effectively. Replacing the full point cloud with only its centroid (Mean Point) significantly degrades performance, confirming that detailed object geometry is necessary for accurate switching decisions. Lastly, removing joint positions and providing only the end-effector wrist pose (EE Pose) also results in large errors, suggesting that the full robot state, including finger states, is critical for determining whether the hand is in the correct pre-grasp pose.

C. Grasping Policy Evaluation

The grasping policy is a key component of our system. We benchmark it in both fully autonomous and shared-autonomy settings. We compare our policy against DexDiffuser [50], a recent method for dexterous grasping using diffusion models. For each trial, we sample 128 candidate poses, select a valid IK solution, and apply motion planning to execute the grasp. The evaluation consists of 5 trials per object across 6 unseen objects (see Fig. 6). Both methods take partial point clouds as input and utilize motion planning to execute grasp poses. Since DexDiffuser was originally trained on the Allegro Hand, we retrain it using our dataset and platform. Specifically, we retrain only the DexSampler module and omit the DexEvaluator module used in the original paper. Both methods are trained entirely in simulation and directly transferred to the real world.

We report results in Tab. II. In the fully autonomous setting, the robot executes grasps without human input. Compared to DexDiffuser [50], our CVAE-based policy achieves higher success rates (76.67% vs. 66.67%) and runs in real time without the overhead of diffusion sampling. In the shared-autonomy setting (Pick & Place task), the human guides coarse motion while the policy executes the grasp. Our method not only outperforms DexDiffuser but

Task	Description	System	Input Device	Policy Switch	Success (↑)	Time (↓)
		Short-horizon Tasks				
PNP Watermelon	Pick watermelon and place in box.	Bunny-VisionPro	VR	–	9 / 10	9.10
		NeuralTeleop	VR	Neural	**9 / 10**	**3.95**
Stack Prism	Stack a hexagonal prism on top of a cube.	Bunny-VisionPro	VR	–	7 / 10	8.01
		NeuralTeleop	VR	Neural	**10 / 10**	**5.40**
Pour Gelatin	Grasp the gelatin then pour it into the bowl	Bunny-VisionPro	VR	–	**7 / 10**	8.95
		NeuralTeleop	VR	Neural	6 / 10	**5.45**
		Long-horizon Tasks				
Cube Sorting	Pick up the cubes and toss them into the corresponding cases.	Mix	–	–	7 / 10	25.89
Can Insertion	Pick up the cans and insert them into the container.	Mix	–	–	7 / 10	50.65

TABLE IV: **Imitation Learning Results.** We compare ACT performance on three short-horizon manipulation tasks (top) and two long-horizon tasks (bottom).

also surpasses the fully-autonomous counterparts, thanks to better pre-grasp poses provided by the teleoperator.

D. Imitation Learning Results

To evaluate the quality of demonstrations collected by our system from an imitation learning perspective, we train ACT [1] on demonstrations collected by BunnyVisionPro [3] and by our system, and report the results in Table IV. Unlike the original ACT implementation, we use a single camera with an RGB-D 4-channel input. We expand the input channels of the vision encoder to support this RGB-D data.

Table IV shows that for short-horizon tasks, the imitation-learning performance trained on data collected by our method and by BunnyVisionPro is comparable; however, our method is much faster than BunnyVisionPro, owing to the shorter episode length during data collection.

Since we collect only 15 episodes per task, ACT performs well on short-horizon tasks but not on long-horizon tasks. Therefore, we mix episodes from different teleoperation systems and report results obtained by training on the merged data for long-horizon tasks. Even so, we cannot achieve a single successful "Prepare Coffee" episode: although the method can grasp the bottle, pour, and place it, it fails to grasp the spoon.

E. System Profiling

While the latency of the teleoperation system is crucial, we profile the runtime of different modules described in Sec. III and report the results in Tab. V. Note that retargeting and inverse kinematics (IK) are executed on the CPU, while segmentation, the switching policy, and the grasping policy are executed on the GPU. In our implementation, CPU and GPU components run in separate processes.

Module	Time (ms)
Retargeting	1.41
IK	0.21
Segmentation	52.03
Neural Switcher	4.09
Grasping Policy	13.03

TABLE V: Profiling Results.

V. Conclusion

We present **Neural Teleoperation**, a shared-autonomy framework that seamlessly integrates human teleoperation with learned autonomous grasping via a policy switcher. Our system supports both immersive hand tracking and lightweight 6-DoF controllers, enabling flexible and efficient demonstration collection for dexterous manipulation tasks. Through extensive real-world experiments, we show that Neural Teleop not only improves task success rates but also significantly reduces collection time across both short- and long-horizon scenarios. Our results demonstrate that learned policy switching can reliably replace manual toggling, while controller-based interfaces can deliver competitive performance without fine-grained hand tracking. These results highlight the potential of combining neural policies with flexible teleoperation interfaces for scalable and efficient demonstration collection. We are committed to releasing the code upon acceptance.

VI. Limitations

Our system has several limitations. First, performance depends heavily on the grasping policy. The current CVAE-based policy is efficient and robust across diverse objects, but it is open-loop and does not use feedback. A closed-loop alternative may improve reliability. Second, although grasping is a core action, many teleoperation tasks require a broader action set, such as placing, opening, and closing. Expanding beyond grasping is important for wider use. Third, while our learned switcher decides when to trigger grasping effectively, extending policy switching across multiple policies for complex long-horizon tasks remains future work. Finally, performance also depends on the teleoperator's skill level. More experienced operators can reach good pre-grasp poses faster and more consistently, which directly helps the neural switcher trigger at the right time and improves overall efficiency. Understanding how operator expertise affects system performance and demonstration quality is an important direction for future study.

VII. Acknowledgment

This project was supported, in part, by NSF CAREER Award IIS-2240014, NSF CCF-2112665 (TILOS), gifts from Amazon, Meta, and Qualcomm.

References

[1] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *RSS*, 2023.

[2] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid, "Aloha unleashed: A simple recipe for robot dexterity," in *CoRL*, 2024.

[3] 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*, 2024.

[4] A. Rodriguez, M. T. Mason, and S. Ferry, "From caging to grasping," *IJRR*, 2012.

[5] C. Rosales, R. Suárez, M. Gabiccini, and A. Bicchi, "On the synthesis of feasible and prehensile robotic grasps," in *ICRA*, 2012.

[6] D. Prattichizzo, M. Malvezzi, M. Gabiccini, and A. Bicchi, "On the manipulability ellipsoids of underactuated robotic hands with compliance," *RAS*, 2012.

[7] J. Ponce, S. Sullivan, J.-D. Boissonnat, and J.-P. Merlet, "On characterizing and computing three-and four-finger force-closure grasps of polyhedral objects," in *ICRA*, 1993.

[8] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet, "On computing four-finger equilibrium and force-closure grasps of polyhedral objects," *IJRR*, 1997.

[9] Y. Zheng and C.-M. Chew, "Distance between a point and a convex cone in *n*-dimensional space: Computation and applications," *T-RO*, 2009.

[10] H. Dai, A. Majumdar, and R. Tedrake, "Synthesis and optimization of force closure grasps via sequential semidefinite programming," *ISRR*, 2018.

[11] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *IJRR*, 2020.

[12] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *CoRL*, 2020.

[13] H. Jiang, S. Liu, J. Wang, and X. Wang, "Hand-object contact consistency reasoning for human grasps generation," in *ICCV*, 2021.

[14] E. Corona, A. Pumarola, G. Alenya, F. Moreno-Noguer, and G. Rogez, "Ganhand: Predicting human grasp affordances in multi-object scenes," in *CVPR*, 2020.

[15] J. Ye, J. Wang, B. Huang, Y. Qin, and X. Wang, "Learning continuous grasping function with a dexterous hand from human demonstrations," *RA-L*, 2023.

[16] L. Shao, F. Ferreira, M. Jorda, V. Nambiar, J. Luo, E. Solowjow, J. A. Ojea, O. Khatib, and J. Bohg, "Unigrasp: Learning a unified model to grasp with multifingered robotic hands," *RA-L*, 2020.

[17] A. Wu, M. Guo, and C. K. Liu, "Learning diverse and physically feasible dexterous grasps with generative model and bilevel optimization," *CoRL*, 2022.

[18] J. Ye, K. Wang, C. Yuan, R. Yang, Y. Li, J. Zhu, Y. Qin, X. Zou, and X. Wang, "Dex1b: Learning with 1b demonstrations for dexterous manipulation," in *RSS*, 2025.

[19] S. Brahmbhatt, A. Handa, J. Hays, and D. Fox, "Contactgrasp: Functional multi-finger grasp synthesis from contact," in *IROS*, 2019.

[20] D. Turpin, L. Wang, E. Heiden, Y.-C. Chen, M. Macklin, S. Tsogkas, S. Dickinson, and A. Garg, "Grasp'd: Differentiable contact-rich grasp synthesis for multi-fingered hands," in *ECCV*, 2022.

[21] D. Aarno, S. Ekvall, and D. Kragic, "Adaptive virtual fixtures for machine-assisted teleoperation tasks," in *ICRA*, 2005.

[22] H. K. Kim, J. Biggs, W. Schloerb, M. Carmena, M. A. Lebedev, M. A. Nicolelis, and M. A. Srinivasan, "Continuous shared control for stabilizing reaching and grasping with brain-machine interfaces," *TBME*, 2006.

[23] T. Carlson and Y. Demiris, "Human-wheelchair collaboration through prediction of intention and adaptive assistance," in *ICRA*, 2008.

[24] S. Schröer, I. Killmann, B. Frank, M. Völker, L. Fiederer, T. Ball, and W. Burgard, "An autonomous robotic assistant for drinking," in *ICRA*, 2015.

[25] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *ICRA*, 2017.

[26] S. Reddy, A. D. Dragan, and S. Levine, "Shared autonomy via deep reinforcement learning," *RSS*, 2018.

[27] Y. Du, S. Tiomkin, E. Kiciman, D. Polani, P. Abbeel, and A. Dragan, "Ave: Assistance via empowerment," *NeurIPS*, 2020.

[28] S. Reddy, S. Levine, and A. Dragan, "First contact: Unsupervised human-machine co-adaptation via mutual information maximization," *NeurIPS*, 2022.

[29] S. Javdani, S. S. Srinivasa, and J. A. Bagnell, "Shared autonomy via hindsight optimization," *RSS*, 2015.

[30] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu, "Dexcap: Scalable and portable mocap data collection system for dexterous manipulation," *RSS*, 2024.

[31] H. Liu, Z. Zhang, X. Xie, Y. Zhu, Y. Liu, Y. Wang, and S.-C. Zhu, "High-fidelity grasping in virtual reality using a glove-based system," in *ICRA*, 2019.

[32] 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 *RSS*, 2023.

[33] A. Sivakumar, K. Shaw, and D. Pathak, "Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube," *arXiv*, 2022.

[34] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox, "Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system," in *ICRA*, 2020.

[35] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto, "Holo-dex: Teaching dexterity with immersive mixed reality," in *ICRA*, 2023.

[36] L. S. Yim, Q. T. Vo, C.-I. Huang, C.-R. Wang, W. McQueary, H.-C. Wang, H. Huang, and L.-F. Yu, "Wfh-vr: Teleoperating a robot arm to set a dining table across the globe via virtual reality," in *IROS*, 2022.

[37] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," *CoRL*, 2024.

[38] J. Ye, L. Wei, G. Jiang, C. Jing, X. Zou, and X. Wang, "From power to precision: Learning fine-grained dexterity for multi-fingered robotic hands," *arXiv*, 2025.

[39] S. Caron, Y. De Mont-Marin, R. Budhiraja, S. H. Bang, I. Domrachev, and S. Nedelchev, "Pink: Python inverse kinematics based on Pinocchio," 2025.

[40] J. Carpentier, F. Valenza, N. Mansard, *et al.*, "Pinocchio: fast forward and inverse dynamics for poly-articulated systems," 2015.

[41] S. Dasari, A. Gupta, and V. Kumar, "Learning dexterous manipulation from exemplar object trajectories and pre-grasps," in *ICRA*, 2023.

[42] T. Wu, Y. Gan, M. Wu, J. Cheng, Y. Yang, Y. Zhu, and H. Dong, "Dexterous functional pre-grasp manipulation with diffusion policy," *arXiv*, 2024.

[43] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *ECCV*, 2024.

[44] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, *et al.*, "Sam 2: Segment anything in images and videos," *arXiv*, 2024.

[45] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[46] R. Wang, J. Zhang, J. Chen, Y. Xu, P. Li, T. Liu, and H. Wang, "Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation," in *ICRA*, 2023.

[47] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang, *et al.*, "Sapien: A simulated part-based interactive environment," in *CVPR*, 2020.

[48] S. Tao, F. Xiang, A. Shukla, Y. Qin, X. Hinrichsen, X. Yuan, C. Bao, X. Lin, Y. Liu, T. kai Chan, Y. Gao, X. Li, T. Mu, N. Xiao, A. Gurha, Z. Huang, R. Calandra, R. Chen, S. Luo, and H. Su, "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," *arXiv*, 2024.

[49] S. Contributors, "Spconv: Spatially sparse convolution library," https://github.com/traveller59/spconv, 2022.

[50] Z. Weng, H. Lu, D. Kragic, and J. Lundell, "Dexdiffuser: Generating dexterous grasps with diffusion models," *RA-L*, 2024.