Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware

Tony Z. Zhao [1]   Vikash Kumar [2]   Sergey Levine [3]   Chelsea Finn [1]

Abstract

Fine manipulation tasks, such as threading cable ties or slotting a battery, are notoriously difficult for robots because they require precision, careful coordination of contact forces, and closed-loop visual feedback. Performing these tasks typically requires high-end robots, accurate sensors, or careful calibration, which can be expensive and difficult to set up. *Can learning enable low-cost and imprecise hardware to perform these fine manipulation tasks?* We present a low-cost system that performs end-to-end imitation learning directly from real demonstrations, collected with a custom teleoperation interface. Imitation learning, however, presents its own challenges, particularly in high-precision domains: errors in the policy can compound over time, and human demonstrations can be non-stationary. To address these challenges, we develop a simple yet novel algorithm, *Action Chunking with Transformers (ACT)*, which learns a generative model over action sequences. ACT allows the robot to learn 6 difficult tasks in the real world, such as opening a translucent condiment cup and slotting a battery with 80-90% success, with only 10 minutes worth of demonstrations.

1. Introduction

Fine manipulation tasks involve precise, closed-loop feedback and require high degrees of hand-eye coordination to adjust in response to environment changes. Examples of such tasks include opening the lid of a condiment cup or slotting a battery, which involve delicate operations such as pinching, prying, and tearing rather than broad-stroke motions such as picking and placing. Take opening the lid of a condiment cup in Figure 1 as an example, where the cup is initialized upright on the table: the right gripper

needs to first tip it over, and nudge it into the opened left gripper. Then the left gripper closes gently and lifts the cup off the table. Next, one of the right fingers approaches the cup from below and pries the lid open. Each of these steps requires high precision, delicate hand-eye coordination, and rich contact. Millimeters of error would lead to task failure.

Existing systems for fine manipulation use expensive robots and high-end sensors for precise state estimation (Ke et al., 2021; Sundaresan et al., 2021; Kim et al., 2022; Paradis et al., 2020). In this work, we seek to develop a low-cost system for fine manipulation that is, in contrast, accessible and reproducible. However, low-cost hardware is inevitably less precise than high-end platforms, making the sensing and planning challenge more pronounced. One promising direction to resolve this is to incorporate learning into the system. Humans also do not have industrial-grade proprioception (Áron Horváth et al., 2022), and yet we are able to perform delicate tasks by learning from closed-loop visual feedback and actively compensating for errors. In our system, we therefore train an end-to-end policy that directly maps RGB images from commodity web cameras to the actions. This pixel-to-action formulation is particularly suitable for fine manipulation, because fine manipulation often involves objects with complex physical properties, such that learning the manipulation policy is much simpler than modeling the whole environment. Take the condiment cup example: modeling the contact when nudging the cup, and also the deformation when prying open the lid involves complex physics on a large number of degrees of freedom. Designing a model accurate enough for planning would require significant research and task specific engineering efforts. In contrast, the policy of nudging and opening the cup is much simpler, since a closed-loop policy can react to different positions of the cup and lid rather than precisely anticipating how it will move in advance.

Training an end-to-end policy, however, presents its own challenges. The performance of the policy depends heavily on the training data distribution, and in the case of fine manipulation, high-quality human demonstrations can provide tremendous value by allowing the system to learn from human dexterity. We thus build a low-cost yet dexterous teleoperation system for data collection, and a novel imitation learning algorithm that learns effectively from the demonstrations. We overview each component in the following

---

[1]Stanford University [2]Meta [3]UC Berkeley. Correspondence to: Tony Z. Zhao <tonyzhao@stanford.edu>.
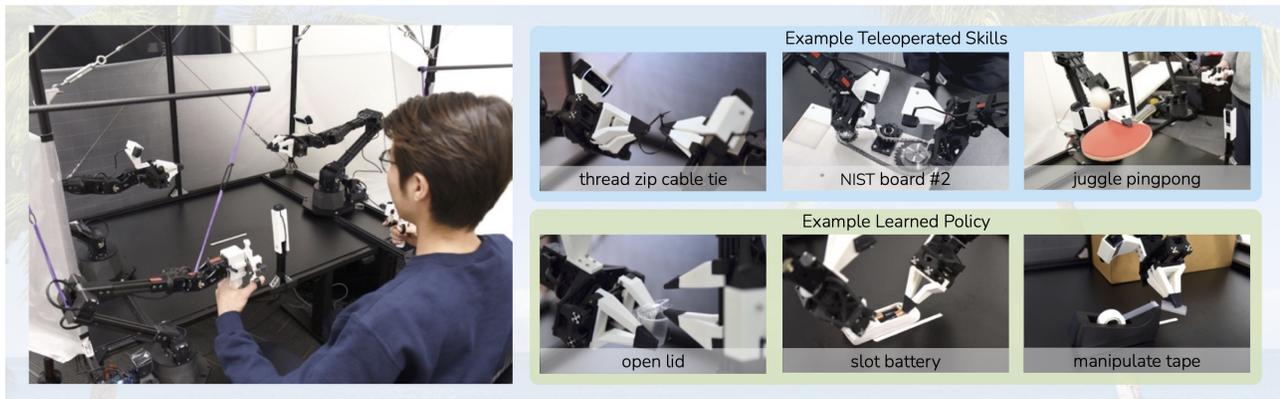
Figure 1: *ALOHA* 🍵 *: A Low-cost Open-source Hardware System for Bimanual Teleoperation*. The whole system costs <$20k with off-the-shelf robots and 3D printed components. *Left:* The user teleoperates by backdriving the leader robots, with the follower robots mirroring the motion. *Right: ALOHA* is capable of precise, contact-rich, and dynamic tasks. We show examples of both teleoperated and learned skills.

two paragraphs.

Teleoperation system. We devise a teleoperation setup with two sets of low-cost, off-the-shelf robot arms. They are approximately scaled versions of each other, and we use joint-space mapping for teleoperation. We augment this setup with 3D printed components for easier backdriving, leading to a highly capable teleoperation system within a $20k budget. We showcase its capabilities in Figure 1, including teleoperation of precise tasks such as threading a zip tie, dynamic tasks such as juggling a ping pong ball, and contact-rich tasks such as assembling the chain in the NIST board #2 (nis, 2022). For more details about *ALOHA*, please refer to the Appendix A.

Imitation learning algorithm. Tasks that require precision and visual feedback present a significant challenge for imitation learning, even with high-quality demonstrations. Small errors in the predicted action can incur large differences in the state, exacerbating the "compounding error" problem of imitation learning (Ross et al., 2010; Tu et al., 2021; Ke et al., 2021). To tackle this, we take inspiration from *action chunking*, a concept in psychology that describes how sequences of actions are grouped together as a chunk, and executed as one unit (Lai et al., 2022). In our case, the policy predicts the target joint positions for the next k timesteps, rather than just one step at a time. This reduces the effective horizon of the task by k-fold, mitigating compounding errors. Predicting action sequences also helps tackle temporally correlated confounders (Swamy et al., 2022), such as pauses in demonstrations that are hard to model with Markovian single-step policies. To further improve the smoothness of the policy, we propose *temporal ensembling*, which queries the policy more frequently and averages across the overlapping action chunks. We implement action chunking policy with Transformers (Vaswani et al., 2017), an architecture designed for sequence model-

ing, and train it as a conditional VAE (CVAE) (Sohn et al., 2015; Kingma and Welling, 2013) to capture the variability in human data. We name our method *Action Chunking with Transformers* (ACT), and find that it significantly outperforms previous imitation learning algorithms on a range of simulated and real-world fine manipulation tasks.

The key contribution of this paper is a low-cost system for learning fine manipulation, comprising a teleoperation system and a novel imitation learning algorithm. The teleoperation system, despite its low cost, enables tasks with high precision and rich contacts. The imitation learning algorithm, Action Chunking with Transformers (ACT), is capable of learning precise, close-loop behavior and drastically outperforms previous methods. The synergy between these two parts allows learning of 6 fine manipulation skills directly in the real-world, such as opening a translucent condiment cup and slotting a battery with 80-90% success, from only 10 minutes or 50 demonstration trajectories.

2. Related Work

Imitation learning for robotic manipulation. Imitation learning allows a robot to directly learn from experts. Behavioral cloning (BC) (Pomerleau, 1988) is one of the simplest imitation learning algorithms, casting imitation as supervised learning from observations to actions. Many works have then sought to improve BC, for example by incorporating history with various architectures (Mandlekar et al., 2021; Shafiullah et al., 2022; Jang et al., 2022; Brohan et al., 2022), using a different training objective (Florence et al., 2021; Pari et al., 2021), and including regularization (Rahmatizadeh et al., 2017). Other works emphasize the multi-task or few-shot aspect of imitation learning (Duan et al., 2017; James et al., 2018; Dasari and Gupta, 2020), leveraging language (Shridhar et al., 2021; 2022; Jang et al., 2022; Brohan et al., 2022), or exploiting the specific task

structure (Pastor et al., 2009; Zeng et al., 2020; Johns, 2021; Shridhar et al., 2022). Scaling these imitation learning algorithms with more data has led to impressive systems that can generalize to new objects, instructions, or scenes (Ebert et al., 2021; Jang et al., 2022; Brohan et al., 2022; Kim et al., 2022). In this work, we focus on building an imitation learning system that is low-cost yet capable of performing delicate, fine manipulation tasks. We tackle this from both hardware and software, by building a high-performance teleoperation system, and a novel imitation learning algorithm that drastically improves previous methods on fine manipulation tasks.

Addressing compounding errors. A major shortcoming of BC is compounding errors, where errors from previous timesteps accumulate and cause the robot to drift off of its training distribution, leading to hard-to-recover states (Ross et al., 2010; Tu et al., 2021). This problem is particularly prominent in the fine manipulation setting (Ke et al., 2021). One way to mitigate compounding errors is to allow additional on-policy interactions and expert corrections, such as DAgger (Ross et al., 2010) and its variants (Kelly et al., 2018; Menda et al., 2018; Hoque et al., 2021). However, expert annotation can be time-consuming and unnatural with a teleoperation interface (Ke et al., 2021). One could also inject noise at demonstration collection time to obtain datasets with corrective behavior (Laskey et al., 2017), but for fine manipulation, such noise injection can directly lead to task failure, reducing the dexterity of teleoperation system. To circumvent these issues, previous works generate synthetic correction data in an offline manner (Florence et al., 2019; Ke et al., 2021; Zhou et al., 2023). While they are limited to settings where low-dimensional states are available, or a specific type of task like grasping. Due to these limitations, we need to address the compounding error problem from a different angle, compatible with high-dimensional visual observations. We propose to reduce the effective horizon of tasks through action chunking, i.e., predicting an action sequence instead of a single action, and then ensemble across overlapping action chunks to produce trajectories that are both accurate and smooth.

3. Action Chunking with Transformers

As we will see in Section 4, existing imitation learning algorithms perform poorly on fine-grained tasks that require high-frequency control and closed-loop feedback. We therefore develop a novel algorithm, *Action Chunking with Transformers (ACT)*, to leverage the data collected by *ALOHA*. We first summarize the pipeline of training ACT, then dive into each of the design choices.

To train ACT on a new task, we first collect human demonstrations using *ALOHA*. We record the joint positions of the leader robots (i.e. input from the human operator) and use

them as actions. It is important to use the leader joint positions instead of the follower's, because the amount of force applied is implicitly defined by the difference between them, through the low-level PID controller. The observations are composed of the current joint positions of follower robots and the image feed from 4 cameras. Next, we train ACT to predict the *sequence of future actions* given the current observations. An action here corresponds to the target joint positions for both arms in the next time step. Intuitively, ACT tries to imitate what a human operator would do in the following time steps given current observations. These target joint positions are then tracked by the low-level, high-frequency PID controller inside Dynamixel motors. At test time, we load the policy that achieves the lowest validation loss and roll it out in the environment. The main challenge that arises is compounding errors, where errors from previous actions lead to states that are outside of training distribution.

3.1. Action Chunking and Temporal Ensemble

To combat the compounding errors of imitation learning in a way that is compatible with pixel-to-action policies (Figure 2), we seek to reduce the effective horizon of long trajectories collected at high frequency. We are inspired by *action chunking*, a neuroscience concept where individual actions are grouped together and executed as one unit, making them more efficient to store and execute (Lai et al., 2022). Intuitively, a chunk of actions could correspond to grasping a corner of the candy wrapper or inserting a battery into the slot. In our implementation, we fix the chunk size to be k: every k steps, the agent receives an observation, generates the next k actions, and executes the actions in sequence (Figure 3). This implies a k-fold reduction in the effective horizon of the task. Concretely, the policy models $\pi_\theta(a_{t:t+k}|s_t)$ instead of $\pi_\theta(a_t|s_t)$. Chunking can also help model non-Markovian behavior in human demonstrations. Specifically, a single-step policy would struggle with temporally correlated confounders, such as pauses in the middle of a demonstration (Swamy et al., 2022), since the behavior not only depends on the state, but also the timestep. Action chunking can mitigate this issue when the confounder is within a chunk, without introducing the causal confusion issue for history-conditioned policies (de Haan et al., 2019).

A naïve implementation of action chunking can be suboptimal: a new environment observation is incorporated abruptly every k steps and can result in jerky robot motion. To improve smoothness and avoid discrete switching between executing and observing, we query the policy at every timestep. This makes different action chunks overlap with each other, and at a given timestep there will be more than one predicted action. We illustrate this in Figure 3 and propose a *temporal ensemble* to combine these predictions. Our *temporal ensemble* performs a weighted average over
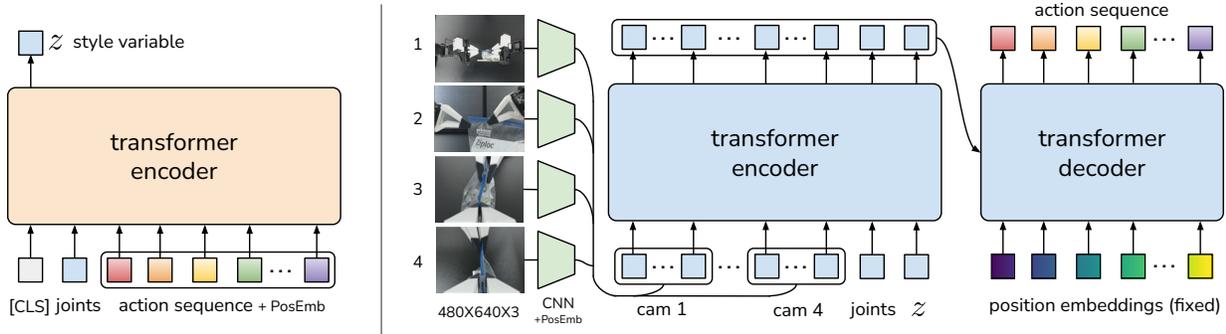
Figure 2: *Architecture of Action Chunking with Transformers (ACT)*. We train ACT as a Conditional VAE (CVAE), which has an encoder and a decoder. *Left:* The encoder of the CVAE compresses action sequence and joint observation into z, the style variable. The encoder is discarded at test time. *Right:* The decoder or policy of ACT synthesizes images from multiple viewpoints, joint positions, and z with a transformer encoder, and predicts a sequence of actions with a transformer decoder. z is simply set to zero at test time.
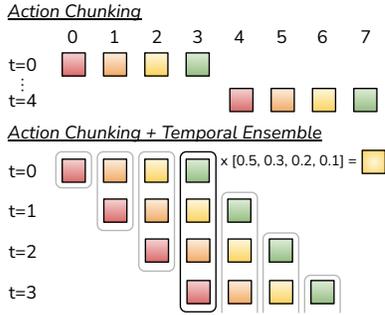
Figure 3: We employ both Action Chunking and Temporal Ensembling when applying actions, instead of interleaving observing and executing.

these predictions with an exponential weighting scheme $w_i = \exp(-m * i)$, where w_0 is the weight for the oldest action. The speed for incorporating new observation is governed by m, where a smaller m means faster incorporation. We note that unlike typical smoothing, where the current action is aggregated with actions in adjacent timesteps, which leads to bias, we aggregate actions predicted for the *same* timestep. This procedure also incurs no additional training cost, only extra inference-time computation. In practice, we find both action chunking and temporal ensembling to be important for the success of ACT, which produces precise and smooth motion. We discuss these components in more detail in the ablation studies in Subsection 5.1.

3.2. Modeling human data

Another challenge that arises is learning from noisy human demonstrations. Given the same observation, a human can use different trajectories to solve the task. Humans will also be more stochastic in regions where precision matters less (Li, 2006). Thus, it is important for the policy to focus on regions where high precision matters. We tackle this problem by training our action chunking policy as a generative model. Specifically, we train the policy as a conditional variational autoencoder (CVAE) (Sohn et al., 2015),

---

Algorithm 1 ACT Training

1: Given: Demo dataset $\mathcal{D}$, chunk size k, weight β.
2: Let a_t, o_t represent action and observation at timestep t, $\bar{o}_t$ represent o_t without image observations.
3: Initialize encoder $q_\phi(z|a_{t:t+k}, \bar{o}_t)$
4: Initialize decoder $\pi_\theta(\hat{a}_{t:t+k}|o_t, z)$
5: **for** iteration $n = 1, 2, ...$ **do**
6:     Sample o_t, $a_{t:t+k}$ from $\mathcal{D}$
7:     Sample z from $q_\phi(z|a_{t:t+k}, \bar{o}_t)$
8:     Predict $\hat{a}_{t:t+k}$ from $\pi_\theta(\hat{a}_{t:t+k}|o_t, z)$
9:     $\mathcal{L}_{reconst} = MSE(\hat{a}_{t:t+k}, a_{t:t+k})$
10:     $\mathcal{L}_{reg} = D_{KL}(q_\phi(z|a_{t:t+k}, \bar{o}_t) \parallel \mathcal{N}(0, I))$
11:     Update θ, ϕ with ADAM and $\mathcal{L} = \mathcal{L}_{reconst} + \beta\mathcal{L}_{reg}$
12: **end for**

Algorithm 2 ACT Inference

1: Given: trained π_θ, episode length T, weight m.
2: Initialize FIFO buffers $\mathcal{B}[0 : T]$, where $\mathcal{B}[t]$ stores actions predicted *for* timestep t.
3: **for** timestep $t = 1, 2, ...T$ **do**
4:     Predict $\hat{a}_{t:t+k}$ with $\pi_\theta(\hat{a}_{t:t+k}|o_t, z)$ where $z = 0$
5:     Add $\hat{a}_{t:t+k}$ to buffers $\mathcal{B}[t : t + k]$ respectively
6:     Obtain current step actions $A_t = \mathcal{B}[t]$
7:     Apply $a_t = \sum_i w_i A_t[i] / \sum_i w_i$, with $w_i = \exp(-m * i)$
8: **end for**

---

to generate an action sequence conditioned on current observations. The CVAE has two components: a CVAE encoder and a CVAE decoder, illustrated on the left and right side of Figure 2 respectively. The CVAE encoder only serves to train the CVAE decoder (the policy) and is discarded at test time. Specifically, the CVAE encoder predicts the mean and variance of the style variable z's distribution, which is parameterized as a diagonal Gaussian, given the current observation and action sequence as inputs. For faster training in practice, we leave out the image observations and only condition on the proprioceptive observation and the action sequence. The CVAE decoder, i.e. the policy, conditions on both z and the current observations (images + joint positions) to predict the action sequence. At test time, we set z to be the mean of the prior distribution i.e. zero to deter-

ministically decode. The whole model is trained to maximize the log-likelihood of demonstration action chunks, i.e. $\min_\theta - \sum_{s_t, a_{t:t+k} \in D} \log \pi_\theta(a_{t:t+k}|s_t)$, with the standard VAE objective which has two terms: a reconstruction loss and a term that regularizes the encoder to a Gaussian prior. Following (Higgins et al., 2016), we weight the second term with a hyperparameter β. Intuitively, higher β will result in less information transmitted in z (Tishby and Zaslavsky, 2015). Overall, we found the CVAE objective to be essential in learning precise tasks from human demonstrations. We include a more detailed discussion in Subsection 5.2.

3.3. Implementing ACT

We implement the CVAE encoder and decoder with transformers, as transformers are designed for both synthesizing information across a sequence and generating new sequences. The CVAE encoder is implemented with a BERT-like transformer encoder (Devlin et al., 2019). The inputs to the encoder are the current joint positions and the target action sequence of length k from the demonstration dataset, prepended by a learned "[CLS]" token similar to BERT. This forms a $k + 2$ length input (Figure 2 left). After passing through the transformer, the feature corresponding to "[CLS]" is used to predict the mean and variance of the "style variable" z, which is then used as input to the decoder. The CVAE decoder (i.e. the policy) takes the current observations and z as the input, and predicts the next k actions (Figure 2 right). We use ResNet image encoders, a transformer encoder, and a transformer decoder to implement the CVAE decoder. Intuitively, the transformer encoder synthesizes information from different camera viewpoints, the joint positions, and the style variable, and the transformer decoder generates a coherent action sequence. The observation includes 4 RGB images, each at 480×640 resolution, and joint positions for two robot arms (7+7=14 DoF in total). The action space is the absolute joint positions for two robots, a 14-dimensional vector. Thus with action chunking, the policy outputs a $k \times 14$ tensor given the current observation. The policy first process the images with ResNet18 backbones (He et al., 2015), which convert $480 \times 640 \times 3$ RGB images into $15 \times 20 \times 512$ feature maps. We then flatten along the spatial dimension to obtain a sequence of 300×512. To preserve the spatial information, we add a 2D sinusoidal position embedding to the feature sequence (Carion et al., 2020). Repeating this for all 4 images gives a feature sequence of 1200×512 in dimension. We then append two more features: the current joint positions and the "style variable" z. They are projected from their original dimensions to 512 through linear layers respectively. Thus, the input to the transformer encoder is 1202×512. The transformer decoder conditions on the encoder output through cross-attention, where the input sequence is a fixed position embedding, with dimensions $k \times 512$, and the keys

and values are coming from the encoder. This gives the transformer decoder an output dimension of $k \times 512$, which is then down-projected with an MLP into $k \times 14$, corresponding to the predicted target joint positions for the next k steps. We use L1 loss for reconstruction instead of the more common L2 loss: we noted that L1 loss leads to more precise modeling of the action sequence. We also noted degraded performance when using delta joint positions as actions instead of target joint positions. We include a detailed architecture diagram in Appendix C.

We summarize the training and inference of ACT in Algorithms 1 and 2. The model has around 80M parameters, and we train from scratch for each task. The training takes around 5 hours on a single 11G RTX 2080 Ti GPU, and the inference time is around 0.01 seconds on the same machine.

4. Experiments

We present experiments to evaluate ACT's performance on fine manipulation tasks. For ease of reproducibility, we build two simulated fine manipulation tasks in MuJoCo (Todorov et al., 2012), in addition to 6 real-world tasks with *ALOHA*.

4.1. Tasks

All 8 tasks require fine-grained, bimanual manipulation, and are illustrated in Figure 5. For ***Slide Ziploc***, the right gripper needs to accurately grasp the slider of the ziploc bag and open it, with the left gripper securing the body of the bag. For ***Slot Battery***, the right gripper needs to first place the battery into the slot of the remote controller, then using the tip of fingers to delicately push in the edge of the battery, until it is fully inserted. Because the spring inside the battery slot causes the remote controller to move in the opposite direction during insertion, the left gripper pushes down on the remote to keep it in place. For ***Open Cup***, the goal is to open the lid of a small condiment cup. Because of the cup's small size, the grippers cannot grasp the body of the cup by just approaching it from the side. Therefore we leverage both grippers: the right fingers first lightly tap near the edge of the cup to tip it over, and then nudge it into the open left gripper. This nudging step requires high precision and closing the loop on visual perception. The left gripper then closes gently and lifts the cup off the table, followed by the right finger prying open the lid, which also requires precision to not miss the lid or damage the cup. The goal of ***Thread Velcro*** is to insert one end of a velcro cable tie into the small loop attached to other end. The left gripper needs to first pick up the velcro tie from the table, followed by the right gripper pinching the tail of the tie in mid-air. Then, both arms coordinate to insert one end of the velcro tie into the other in mid-air. The loop measures 3mm x 25mm, while the velcro tie measures 2mm x 10-25mm depending on the

	Cube Transfer (sim)			**Bimanual Insertion** (sim)			**Slide Ziploc** (real)			**Slot Battery** (real)		
	Touched	Lifted	Transfer	Grasp	Contact	Insert	Grasp	Pinch	Open	Grasp	Place	Insert
BC-ConvMLP	34 ∣ 3	17 ∣ 1	1 ∣ 0	5 ∣ 0	1 ∣ 0	1 ∣ 0	0	0	0	0	0	0
BeT	60 ∣ 16	51 ∣ 13	27 ∣ 1	21 ∣ 0	4 ∣ 0	3 ∣ 0	8	0	0	4	0	0
RT-1	44 ∣ 4	33 ∣ 2	2 ∣ 0	2 ∣ 0	0 ∣ 0	1 ∣ 0	4	0	0	4	0	0
VINN	13 ∣ 17	9 ∣ 11	3 ∣ 0	6 ∣ 0	1 ∣ 0	1 ∣ 0	28	0	0	20	0	0
ACT (Ours)	**97 ∣ 82**	**90 ∣ 60**	**86 ∣ 50**	**93 ∣ 76**	**90 ∣ 66**	**32 ∣ 20**	**92**	**96**	**88**	**100**	**100**	**96**

Table 1: Success rate (%) for 2 simulated and 2 real-world tasks, comparing our method with 4 baselines. For the two simulated tasks, we report [training with scripted data ∣ training with human data], with 3 seeds and 50 policy evaluations each. For the real-world tasks, we report training with human data, with 1 seed and 25 evaluations. Overall, ACT significantly outperforms previous methods.

	Open Cup (real)			**Thread Velcro** (real)			**Prep Tape** (real)				**Put On Shoe** (real)			
	Tip Over	Grasp	Open Lid	Lift	Grasp	Insert	Grasp	Cut	Handover	Hang	Lift	Insert	Support	Secure
BeT	12	0	0	24	0	0	8	0	0	0	12	0	0	0
ACT (Ours)	**100**	**96**	**84**	**92**	**40**	**20**	**96**	**92**	**72**	**64**	**100**	**92**	**92**	**92**

Table 2: Success rate (%) for the remaining 3 real-world tasks. We only compare with the best performing baseline BeT.

position. For this task to be successful, the robot must use visual feedback to correct for perturbations with each grasp, as even a few millimeters of error during the first grasp will compound in the second grasp mid-air, giving more than a 10mm deviation in the insertion phase. For ***Prep Tape***, the goal is to hang a small segment of the tape on the edge of a cardboard box. The right gripper first grasps the tape and cuts it with the tape dispenser's blade, and then hands the tape segment to the left gripper mid-air. Next, both arms approach the box, the left arm gently lays the tape segment on the box surface, and the right fingers push down on the tape to prevent slipping, followed by the left arm opening its gripper to release the tape. Similar to *Thread Velcro*, this task requires multiple steps of delicate coordination between the two arms. For ***Put On Shoe***, the goal is to put the shoe on a fixed mannequin foot, and secure it with the shoe's velcro strap. The arms would first need to grasp the tongue and collar of the shoe respectively, lift it up and approach the foot. Putting the shoe on is challenging because of the tight fitting: the arms would need to coordinate carefully to nudge the foot in, and both grasps need to be robust enough to counteract the friction between the sock and shoe. Then, the left arm goes around to the bottom of the shoe to support it from dropping, followed by the right arm flipping the velcro strap and pressing it against the shoe to secure. The task is only considered successful if the shoe clings to the foot after both arms releases.

In addition to the delicate bimanual control required to solve these tasks, the objects we use also present a significant perception challenge. For example, the ziploc bag is largely transparent, with a thin blue sealing line. Both the wrinkles on the bag and the reflective candy wrappers inside can vary during the randomization, and distract the perception system. Other transparent or translucent objects include the tape and

both the lid and body of the condiment cup, making them hard to perceive precisely and ill-suited for depth cameras. The black table top also creates a low-contrast against many objects of interest, such as the black velcro cable tie and the black tape dispenser. Especially from the top view, it is challenging to localize the velcro tie because of the small projected area.

4.2. Data Collection

For all 6 real-world tasks, we collect demonstrations using *ALOHA* teleoperation. Each episode takes 8-14 seconds for the human operator to perform depending on the complexity of the task, which translates to 400-700 time steps given the control frequency of 50Hz. We record 50 demonstrations for each task, except for *Thread Velcro* which has 100. The total amount for demonstrations is thus around 10-20 minutes of data for each task, and 30-60 minutes in wall-clock time because of resets and teleoperator mistakes. For the two simulated tasks, we collect two types of demonstrations: one type with a scripted policy and one with human demonstrations. To teleoperate in simulation, we use the "leader robots" of *ALOHA* to control the simulated robot. In both cases, we record 50 successful demonstrations.

We emphasize that all human demonstrations are inherently stochastic, even though a single person collects all of the demonstrations. Take the mid-air hand handover of the tape segment as an example: the exact position of the handover is different across each episode. The human has no visual or haptic reference to perform it in the same position. Thus to successfully perform the task, the policy will need to learn that the two grippers should never collide with each other during the handover, and the left gripper should always move to a position that can grasp the tape, instead of trying to memorize where exactly the handover happens, which

can vary across demonstrations.

4.3. Experiment Results

We compare ACT with four prior imitation learning methods. **BC-ConvMLP** is the simplest yet most widely used baseline (Zhang et al., 2017; Jang et al., 2022), which processes the current image observations with a convolutional network, whose output features are concatenated with the joint positions to predict the action. **BeT** (Shafiullah et al., 2022) also leverages Transformers as the architecture, but with key differences: (1) no action chunking: the model predicts one action given the history of observations; and (2) the image observations are pre-processed by a separately trained frozen visual encoder. That is, the perception and control networks are not jointly optimized. **RT-1** (Brohan et al., 2022) is another Transformer-based architecture that predicts one action from a fixed-length history of past observations. Both *BeT* and *RT-1* discretize the action space: the output is a categorical distribution over discrete bins, but with an added continuous offset from the bin-center in the case of *BeT*. Our method, ACT, instead directly predicts continuous actions, motivated by the precision required in fine manipulation. Lastly, **VINN** (Pari et al., 2021) is a non-parametric method that assumes access to the demonstrations at test time. Given a new observation, it retrieves the k observations with the most similar visual features, and returns an action using weighted k-nearest-neighbors. The visual feature extractor is a pretrained ResNet finetuned on demonstration data with unsupervised learning. We carefully tune the hyperparameters of these four prior methods using cube transfer. Details of the hyperparameters are provided in Appendix D.

As a detailed comparison with prior methods, we report the average success rate in Table 1 for two simulated and two real tasks. For simulated tasks, we average performance across 3 random seeds with 50 trials each. We report the success rate on both scripted data (left of separation bar) and human data (right of separation bar). For real-world tasks, we run one seed and evaluate with 25 trials. ACT achieves the highest success rate compared to all prior methods, outperforming the second best algorithm by a large margin on each task. For the two simulated tasks with scripted or human data, ACT outperforms the best previous method in success rate by 59%, 49%, 29%, and 20%. While previous methods are able to make progress in the first two subtasks, the final success rate remains low, below 30%. For the two real-world tasks *Slide Ziploc* and *Slot Battery*, ACT achieves 88% and 96% final success rates respectively, with other methods making no progress past the first stage. We attribute the poor performance of prior methods to compounding errors and non-Markovian behavior in the data: the behavior degrades significantly towards the end of an episode, and the robot can pause indefinitely for certain

states. ACT mitigates both issues with action chunking. Our ablations in Subsection 5.1 also shows that chunking can significantly improve these prior methods when incorporated. In addition, we notice a drop in performance for all methods when switching from scripted data to human data in simulated tasks: the stochasticity and multi-modality of human demonstrations make imitation learning a lot harder.

We report the success rate of the 3 remaining real-world tasks in Table 2. For these tasks, we only compare with BeT, which has the highest task success rate so far. Our method ACT reaches 84% success for *Cup Open*, 20% for *Thread Velcro*, 64% for *Prep Tape* and 92% for *Put On Shoe*, again outperforming BeT, which achieve zero final success on these challenging tasks. We observe relatively low success of ACT in *Thread Velcro*, where the success rate decreased by roughly half at every stage, from 92% success at the first stage to 20% final success. The failure modes we observe are 1) at stage 2, the right arm closes its gripper too early and fails to grasp the tail of the cable tie mid-air, and 2) in stage 3, the insertion is not precise enough and misses the loop. In both cases, it is hard to determine the exact position of the cable tie from image observations: the contrast is low between the black cable tie and the background, and the cable tie only occupies a small fraction of the image.

5. Ablations

ACT employs action chunking and temporal ensembling to mitigate compounding errors and better handle non-Markovian demonstrations. It also trains the policy as a conditional VAE to model the noisy human demonstrations. In this section, we ablate each of these components, together with a user study that highlights the necessity of high-frequency control in *ALOHA*. We report results across a total of four settings: two simulated tasks with scripted or human demonstration.

5.1. Action Chunking and Temporal Ensembling

In Subsection 4.3, we observed that ACT significantly outperforms previous methods that only predict single-step actions, with the hypothesis that action chunking is the key design choice. Since k dictates how long the sequence in each "chunk" is, we can analyze this hypothesis by varying k. $k = 1$ corresponds to no action chunking, and $k = episode\_length$ corresponds to fully open-loop control, where the robot outputs the entire episode's action sequence based on the first observation. We disable temporal ensembling in these experiments to only measure the effect of chunking, and trained separate policies for each k. In Figure 4 (a), we plot the success rate averaged across 4 settings, corresponding to 2 simulated tasks with either human or scripted data, with the blue line representing ACT without the temporal ensemble. We observe that perfor-
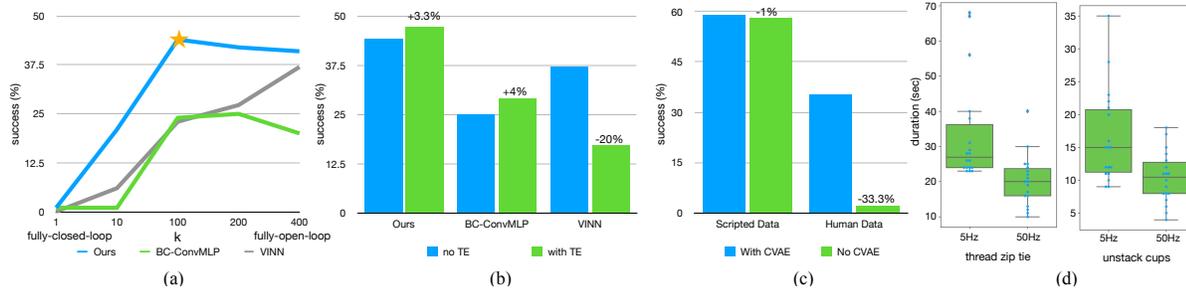
Figure 4: *(a)* We augment two baselines with action chunking, with different values of chunk size k on the x-axis, and success rate on the y-axis. Both methods significantly benefit from action chunking, suggesting that it is a generally useful technique. *(b)* Temporal Ensemble (TE) improves our method and *BC-ConvMLP*, while hurting *VINN*. *(c)* We compare with and without the CVAE training, showing that it is crucial when learning from human data. *(d)* We plot the distribution of task completion time in our user study, where we task participants to perform two tasks, at 5Hz or 50Hz teleoperation frequency. Lowering the frequency results in a 62% slowdown in completion time.

mance improves drastically from 1% at $k = 1$ to 44% at $k = 100$, then slightly tapers down with higher k. This illustrates that more chunking and a lower effective horizon generally improve performance. We attribute the slight dip at $k = 200, 400$ (i.e., close to open-loop control) to the lack of reactive behavior and the difficulty in modeling long action sequences. To further evaluate the effectiveness and generality of action chunking, we augment two baseline methods with action chunking. For *BC-ConvMLP*, we simply increase the output dimension to $k * action\_dim$, and for *VINN*, we retrieve the next k actions. We visualize their performance in Figure 4 (a) with different k, showing trends consistent with ACT, where more action chunking improves performance. While ACT still outperforms both augmented baselines with sizable gains, these results suggest that action chunking is generally beneficial for imitation learning in these settings.

We then ablate the temporal ensemble by comparing the highest success rate with or without it, again across the 4 aforementioned tasks and different k. We note that experiments with and without the temporal ensemble are separately tuned: hyperparameters that work best for no temporal ensemble may not be optimal with a temporal ensemble. In Figure 4 (b), we show that *BC-ConvMLP* benefits from temporal ensembling the most with a 4% gain, followed by a 3.3% gain for our method. We notice a performance drop for *VINN*, a non-parametric method. We hypothesize that a temporal ensemble mostly benefits parametric methods by smoothing out the modeling errors. In contrast, *VINN* retrieves ground-truth actions from the dataset and does not suffer from this issue.

5.2. Training with CVAE

We train ACT with CVAE objective to model human demonstrations, which can be noisy and contain multi-modal behavior. In this section, we compare with ACT without the CVAE objective, which simply predicts a sequence of actions given current observation, and trained with L1 loss. In

Figure 4 (c), we visualize the success rate aggregated across 2 simulated tasks, and separately plot training with scripted data and with human data. We can see that when training on scripted data, the removal of CVAE objective makes almost no difference in performance, because dataset is fully deterministic. While for human data, there is a significant drop from 35.3% to 2%. This illustrates that the CVAE objective is crucial when learning from human demonstrations.

5.3. Is High-Frequency Necessary?

Lastly, we conduct a user study to illustrate the necessity of high-frequency teleoperation for fine manipulation. With the same hardware setup, we lower the frequency from 50Hz to 5Hz, a control frequency that is similar to recent works that use high-capacity deep networks for imitation learning (Brohan et al., 2022; Zhou et al., 2023). We pick two fine-grained tasks: threading a zip cable tie and un-stacking two plastic cups. Both require millimeter-level precision and closed-loop visual feedback. We perform the study with 6 participants who have varying levels of experience with teleoperation, though none had used *ALOHA* before. The participants were recruited from among computer science graduate students, with 4 men and 2 women aged 22-25 The order of tasks and frequencies are randomized for each participant, and each participant was provided with a 2 minutes practice period before each trial. We recorded the time it took to perform the task for 3 trials, and visualize the data in Figure 4 (d). On average, it took 33s for participants to thread the zip tie at 5Hz, which is lowered to 20s at 50Hz. For separating plastic cups, increasing the control frequency lowered the task duration from 16s to 10s. Overall, our setup (i.e. 50Hz) allows the participants to perform highly dexterous and precise tasks in a short amount of time. However, reducing the frequency from 50Hz to 5Hz results in a 62% increase in teleoperation time. We then use "Repeated Measures Designs", a statistical procedure, to formally verify that 50Hz teleoperation outperforms 5Hz with p-value <0.001. We include more details about the

study in Appendix E.

6. Limitations and Conclusion

We present a low-cost system for fine manipulation, comprising a teleoperation system *ALOHA* and a novel imitation learning algorithm *ACT*. The synergy between these two parts allows us to learn fine manipulation skills directly in the real-world, such as opening a translucent condiment cup and slotting a battery with a 80-90% success rate and around 10 min of demonstrations. While the system is quite capable, there exist tasks that are beyond the capability of *ALOHA* and ACT, such as buttoning up a dress shirt. We include a more detailed discussion about limitations in Appendix F. Overall, we hope that this low-cost open-source system represents an important step and accessible resource towards advancing fine-grained robotic manipulation.

References

Viperx 300 robot arm 6dof. URL https://www.trossenrobotics.com/viperx-300-robot-arm-6dof.aspx.

Widowx 250 robot arm 6dof. URL https://www.trossenrobotics.com/widowx-250-robot-arm-6dof.aspx.

Highly dexterous manipulation system - capabilities - part 1, Nov 2014. URL https://www.youtube.com/watch?v=TearcKVj0iY.

Assembly performance metrics and test methods, Apr 2022. URL https://www.nist.gov/el/intelligent-systems-division-73500/robotic-grasping-and-manipulation-assembly/assembly.

Teleoperated robots - shadow teleoperation system, Nov 2022. URL https://www.shadowrobot.com/teleoperation/.

Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. *arXiv preprint arXiv:2210.06463*, 2022.

Pierre Berthet-Rayne, Konrad Leibrandt, Gauthier Gras, Philippe Fraisse, André Crosnier, and Guang-Zhong Yang. Inverse kinematics control methods for redundant snakelike robot teleoperation during minimally invasive surgery. *IEEE Robotics and Automation Letters*, 3:2501–2508, 2018.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Anand Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Ho Vuong, F. Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. *ArXiv*, abs/2212.06817, 2022.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *ArXiv*, abs/2005.12872, 2020.

Sudeep Dasari and Abhinav Kumar Gupta. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, 2020.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Neural Information Processing Systems*, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805, 2019.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, P. Abbeel, and Wojciech Zaremba. One-shot imitation learning. *ArXiv*, abs/1703.07326, 2017.

Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *ArXiv*, abs/2109.13396, 2021.

Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5:492–499, 2019.

Peter R. Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian S. Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *ArXiv*, abs/2109.00137, 2021.

Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan D. Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of

dexterous robotic hand-arm system. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170, 2019.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

Ryan Hoque, Ashwin Balakrishna, Ellen R. Novoseller, Albert Wilcox, Daniel S. Brown, and Ken Goldberg. Thriftydagger: Budget-aware novelty and risk gating for interactive imitation learning. In *Conference on Robot Learning*, 2021.

Stephen James, Michael Bloesch, and Andrew J. Davison. Task-embedded control networks for few-shot imitation learning. *ArXiv*, abs/1810.03237, 2018.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, 2022.

R G Jenness and C D Wicker. Master–slave manipulators and remote maintenance at the oak ridge national laboratory, Jan 1975. URL https://www.osti.gov/biblio/4179544.

Edward Johns. Coarse-to-fine imitation learning: Robot manipulation from a single demonstration. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4613–4619, 2021.

Liyiming Ke, Jingqiang Wang, Tapomayukh Bhattacharjee, Byron Boots, and Siddhartha Srinivasa. Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation. In *International Conference on Robotics and Automation (ICRA)*, 2021.

Michael Kelly, Chelsea Sidrane, K. Driggs-Campbell, and Mykel J. Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8077–8083, 2018.

Heecheol Kim, Yoshiyuki Ohmura, and Yasuo Kuniyoshi. Robot peels banana with goal-conditioned dual-action deep imitation learning. *ArXiv*, abs/2203.09749, 2022.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Lucy Lai, Ann Z Huang, and Samuel J Gershman. Action chunking as policy compression, Sep 2022. URL psyarxiv.com/z8yrv.

Michael Laskey, Jonathan Lee, Roy Fox, Anca D. Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, 2017.

Shuang Li, Qiang Li, and Jianwei Zhang. Chapter 12 - a hand-arm teleoperation system for robotic dexterous manipulation. In Qiang Li, Shan Luo, Zhaopeng Chen, Chenguang Yang, and Jianwei Zhang, editors, *Tactile Sensing, Skill Learning, and Robotic Dexterous Manipulation*, pages 227–245. Academic Press, 2022. ISBN 978-0-323-90445-2. doi: https://doi.org/10.1016/B978-0-32-390445-2.00021-0. URL https://www.sciencedirect.com/science/article/pii/B9780323904452000210.

Weiwei Li. Optimal control for biological movement systems. 2006.

Ajay Mandlekar, Danfei Xu, J. Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Mart'in-Mart'in. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

Kunal Menda, K. Driggs-Campbell, and Mykel J. Kochenderfer. Ensembledagger: A bayesian approach to safe imitation learning. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5041–5048, 2018.

Samuel Paradis, Minho Hwang, Brijen Thananjeyan, Jeffrey Ichnowski, Daniel Seita, Danyal Fer, Thomas Low, Joseph Gonzalez, and Ken Goldberg. Intermittent visual servoing: Efficiently learning policies robust to instrument changes for high-precision surgical manipulation. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7166–7173, 2020.

Jyothish Pari, Nur Muhammad, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.

Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. *2009 IEEE International Conference on Robotics and Automation*, pages 763–768, 2009.

Luis F. Peñín, Kotaro Matsumoto, and . Teleoperation with time delay: A survey and its use in space robotics. 2002.

Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1988.

Yuzhe Qin, Hao Su, and Xiaolong Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. *IEEE Robotics and Automation Letters*, 7:10873–10881, 2022.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765, 2017.

S. Rezaei-Shoshtari, David Meger, and Inna Sharf. Cascaded gaussian processes for data-efficient robot dynamics learning. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6871–6877, 2019.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2010.

Nur Muhammad (Mahi) Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *ArXiv*, abs/2206.11251, 2022.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. *ArXiv*, abs/2109.12098, 2021.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiveractor: A multi-task transformer for robotic manipulation. *ArXiv*, abs/2209.05451, 2022.

Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *RSS*, 2022.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *NIPS*, 2015.

srcteam. Shadow teleoperation system plays jenga, Mar 2021. URL https://www.youtube.com/watch?v=7K9brH27jvM.

srcteam. How researchers are using shadow robot's technology, Jun 2022a. URL https://www.youtube.com/watch?v=p36fYIoTD8M.

srcteam. Shadow teleoperation system, Jun 2022b. URL https://www.youtube.com/watch?v=cx8eznfDUJA.

Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Jeffrey Ichnowski, Ellen R. Novoseller, Minho Hwang, Michael Laskey, Joseph Gonzalez, and Ken Goldberg. Untangling dense non-planar knots by learning manipulation features and recovery policies. *ArXiv*, abs/2107.08942, 2021.

Gokul Swamy, Sanjiban Choudhury, J. Andrew Bagnell, and Zhiwei Steven Wu. Causal imitation learning under temporally correlated noise. In *International Conference on Machine Learning*, 2022.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5, 2015.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

Stephen Tu, Alexander Robey, Tingnan Zhang, and N. Matni. On the sample complexity of stability constrained imitation learning. In *Conference on Learning for Dynamics & Control*, 2021.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

Solomon Wiznitzer, Luke Schmitt, and Matt Trossen. interbotix_ros_manipulators. URL https://github.com/Interbotix/interbotix_ros_manipulators.

Andy Zeng, Peter R. Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, 2020.

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2017.

Allan Zhou, Moo Jin Kim, Lirui Wang, Peter R. Florence, and Chelsea Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. *ArXiv*, abs/2301.08556, 2023.

Áron Horváth, Eszter Ferentzi, Kristóf Schwartz, Nina Jacobs, Pieter Meyns, and Ferenc Köteles. The measurement of proprioceptive accuracy: A systematic literature review. *Journal of Sport and Health Science*, 2022. ISSN

2095-2546. doi: https://doi.org/10.1016/j.jshs.2022.04. 001. URL https://www.sciencedirect.com/ science/article/pii/S2095254622000473.

A. ALOHA: <u>A</u> <u>L</u>ow-cost <u>O</u>pen-source <u>H</u>ardware System for Bimanual Teleoperation

We seek to develop an accessible and high-performance teleoperation system for fine manipulation. We summarize our design considerations into the following 5 principles.

1. **Low-cost**: The entire system should be within budget for most robotic labs, comparable to a single industrial arm.

2. **Versatile**: It can be applied to a wide range of fine manipulation tasks with real-world objects.

3. **User-friendly**: The system should be intuitive, reliable, and easy to use.

4. **Repairable**: The setup can be easily repaired by researchers, when it inevitably breaks.

5. **Easy-to-build**: It can be quickly assembled by researchers, with easy-to-source materials.

When choosing the robot to use, principles 1, 4, and 5 lead us to build a bimanual parallel-jaw grippers setup with two ViperX 6-DoF robot arms (vx; Wiznitzer et al.). We do not employ dexterous hands due to price and maintenance considerations. The ViperX arm used has a working payload of 750g and 1.5m span, with an accuracy of 5-8mm. The robot is modular and simple to repair: in the case of motor failure, the low-cost Dynamixel motors can be easily replaced. The robot can be purchased off-the-shelf for around $5600. The OEM fingers, however, are not versatile enough to handle fine manipulation tasks. We thus design our own 3D printed "see-through" fingers and fit it with gripping tape (Fig 7). This allows for good visibility when performing delicate operations, and robust grip even with thin plastic films.

We then seek to design a teleoperation system that is maximally user-friendly around the ViperX robot. Instead of mapping the hand pose captured by a VR controller or camera to the end-effector pose of the robot, i.e. task-space mapping, we use direct joint-space mapping from a smaller robot, WidowX, manufactured by the same company and costs $3300 (wx). The user teleoperates by backdriving the smaller WidowX ("the leader"), whose joints are synchronized with the larger ViperX ("the follower"). When developing the setup, we noticed a few benefits of using joint-space mapping compared to task-space. (1) Fine manipulation often requires operating near singularities of the robot, which in our case has 6 degrees of freedom and no redundancy. Off-the-shelf inverse kinematics (IK) fails frequently in this setting. Joint space mapping, on the other hand, guarantees high-bandwidth control within the joint limits, while also requiring less computation and reducing

latency. (2) The weight of the leader robot prevents the user from moving too fast, and also dampens small vibrations. We notice better performance on precise tasks with joint-space mapping rather than holding a VR controller. To further improve the teleoperation experience, we design a 3D-printed "handle and scissor" mechanism that can be retrofitted to the leader robot (Fig 7). It reduces the force required from the operator to backdrive the motor, and allows for continuous control of the gripper, instead of binary opening or closing. We also design a rubber band load balancing mechanism that partially counteracts the gravity on the leader side. It reduces the effort needed from the operator and makes longer teleoperation sessions (e.g. >30 minutes) possible.

The rest of the setup includes a robot cage with 20×20mm aluminum extrusions, reinforced by crossing steel cables. There is a total of four Logitech C922x webcams, each streaming 480×640 RGB images. Two of the webcams are mounted on the wrist of the follower robots, allowing for a close-up view of the grippers. The remaining two cameras are mounted on the front and at the top respectively (Fig 7). All robots and webcams are connected to a Linux workstation through USB ports.

Specifically, the robot is composed of a chain of Dynamixel actuators, with an integrated controller and driver. We set all actuators to position control mode, where the underlying proprietary PID controller tracks the position target at >1kHz. The joint targets are read from leader robots and sent to follower robots at 50Hz. It is worth noting that this is not a bilateral system, in which case a much higher frequency is needed to e.g. enable force reflection (Peñín et al., 2002). A teleoperation frequency of 50Hz is common among other non-bilateral systems (Berthet-Rayne et al., 2018; Rezaei-Shoshtari et al., 2019; Li et al., 2022) and in practice demonstrates sufficient performance. We include a quantitative evaluation later in this section.

With the design considerations above, we build the bimanual teleoperation setup *ALOHA* within a 20k USD budget, comparable to a single research arm such as Franka Emika Panda. *ALOHA* enables the teleoperation of:

- **Precise tasks** such as threading zip cable ties, picking credit cards out of wallets, and opening or closing ziploc bags.

- **Contact-rich tasks** such as inserting 288-pin RAM into a computer motherboard, turning pages of a book, and assembling the chains and belts in the NIST board #2 (nis, 2022)

- **Dynamic tasks** such as juggling a ping pong ball with a real ping pong paddle, balancing the ball without it falling off, and swinging open plastic bags in the air.

Real-World Task Definitions

Slide Ziploc: Open the ziploc bag that is standing upright on the table. The bag is randomized along the 15cm white line. It is dropped from ~5cm above the table to randomize the deformation, which affects the height and appearance of the bag. The left arm first grasps the bag body *(Subtask#1 Grasp)* followed by the right arm pinching the slider *(Subtask #2 Pinch)*. Then the right arm moves right to unzip the bag *(Subtask #3 Open)*.
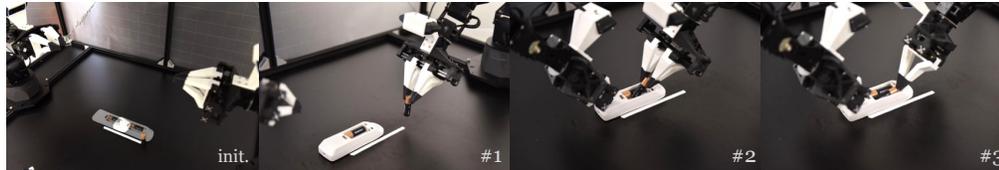
Slot Battery: Insert the battery into the remote controller. The controller is randomized along the 15cm white line. The battery is initialized in roughly the same position with different rotations. The right arm first grasps the battery *(Subtask#1 Grasp)* then places it into the slot *(Subtask#2 Place)*. The left arm presses onto the remote to prevent it from sliding, while the right arm pushes in the battery *(Subtask#3 Insert)*.

Open Cup: Pick up and open the lid of a translucent condiment cup. The cup is randomized along the 15cm white line. Both arms approach the cup, and the right gripper gently tips over the cup *(Subtask#1 Tip Over)* and pushes it into the gripper of the left arm. The left arm then gently closes its gripper and lifts the cup off the table *(Subtask#2 Grasp)*. Next, the right gripper approaches the cup lid from below and prys open the lid.
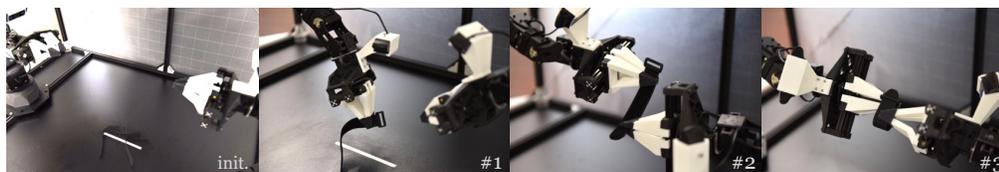
Thread Velcro: Pick up the velcro cable tie and insert one end into the small loop on the other end. The velcro tie is randomized along the 15cm white line. The left arm first picks up the velcro tie by pinching near the plastic loop *(Subtask#1 Lift)*. The right arm grasps the tail of the velcro tie mid-air *(Subtask#2 Grasp)*. Next, both arms coordinate to deform the velcro tie and insert one end of it into the plastic loop on the other end.

Prep Tape: Hang a short segment of tape on the edge of the box. The tape dispenser is randomized along the 15cm white line. First, the right gripper grasps the tape from the side *(Subtask#1 Grasp)*. It then lifts the tape and pulls to unroll it, followed by cutting it with the dispenser blade *(Subtask#2 Cut)*. Next, the right gripper hands the tape segment to the left gripper in mid-air *(Subtask#3 Handover)*, and both arms move toward the corner of the stationery cardboard box. The left arm then lays the tape segment flat on the surface of the box while the right gripper pushes down on the tape to prevent slipping. The left arm then opens its gripper to release the tape *(Subtask#4 Hang)*.

Put On Shoe: Put a velcro-strap shoe on a fixed manniquin foot. The shoe pose is randomized along the 15cm white line. First, both left and right grippers pick up the shoe *(Subtask#1 Lift)*. Then both arms coordinate to put it on, with the heel touching the heel counter *(Subtask#2 Insert)*. Next, the left arm moves to support the shoe *(Subtask#3 Support)*, followed by the right arm securing the velcro strap *(Subtask#4 Secure)*.

Figure 5: *Real-World Task Definitions.* For each of the 6 real-world tasks, we illustrate the initializations and the subtasks.
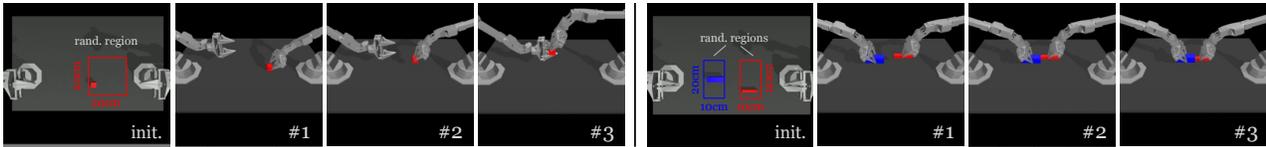
Left: Cube Transfer. Transfer the red cube to the other arm. The right arm touches *(#1)* and grasps *(#2)* the red cube, then hands it to the left arm.
Right: Bimanual Insertion. Insert the red peg into the blue socket. Both arms grasp *(#1)*, let socket and peg make contact *(#2)* and insertion.

Figure 6: *Simulated Task Definitions.* For each of the 2 simulated tasks, we illustrate the initializations and the subtasks.
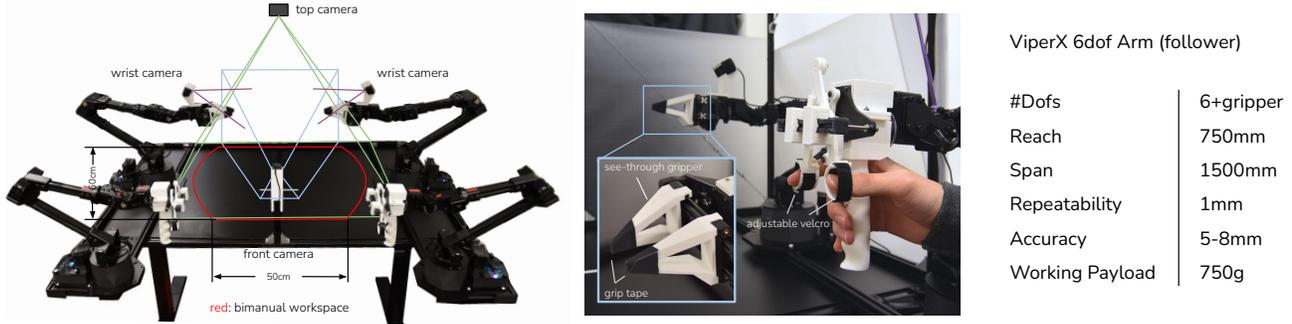
Figure 7: *Left:* Camera viewpoints of the front, top, and two wrist cameras, together with an illustration of the bimanual workspace of *ALOHA*. *Middle:* Detailed view of the "handle and scissor" mechanism and custom grippers. *Right:* Technical spec of the ViperX 6dof robot (vx).

Precise tasks			Contact-rich tasks			Dynamic tasks		
Thread zip tie (in 5s)	Pick credit card (in 5s)	Open ziploc (in 5s)	RAM insertion (in 30s)	Turn pages (in 30s)	NIST board belt (in 30s)	Juggle ping pong (for 5 times)	Balance ping pong (for 20s)	Swing open bags (in 5s)
10/10	10/10	10/10	9/10	10/10	8/10	4/10	9/10	10/10

Table 3: *Teleoperation success.* We report the number of successes out of 10 trials. The task is only successful if completed within the time limit specified in the table, excepting the two ping pong tasks where success is measured in duration. All objects used are off-the-shelf.
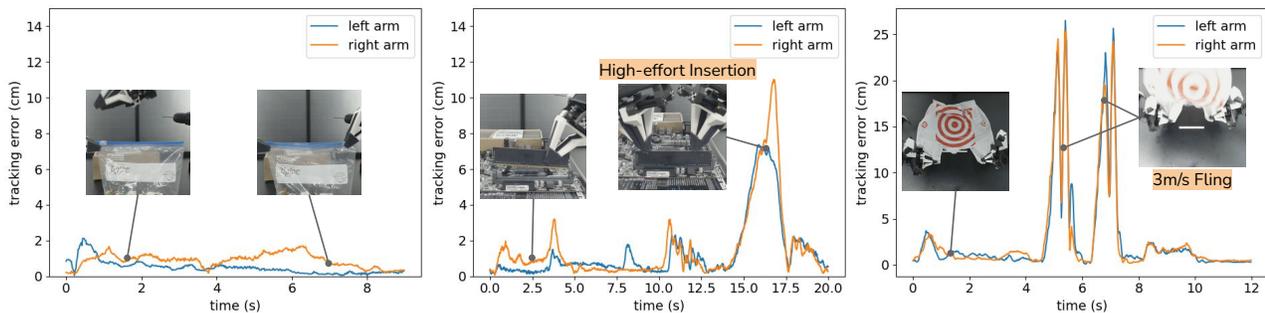
Figure 8: *Teleoperation accuracy.* We plot the position tracking error of the *ALOHA* teleoperation system for three representative tasks. (a) A relatively smooth task of sliding open a ziploc bag. The tracking error is consistently below 2cm with an average of 0.68cm, comparable to the accuracy of the robot arm. (b) The tracking error can spike when the follower arms are exerting high-amount of force, e.g. inserting a RAM stick into a tight-fitting slot. This is due to the non-bilateral nature of the system. (c) High tracking error can also occur during sudden acceleration, as in the case of swinging open a plastic bag. This is due to a combination of teleoperation delay and motor torque limitations.

Skills such as threading a zip tie, inserting RAM, and juggling ping pong ball, to our knowledge, are not available for existing teleoperation systems with 5-10x the budget (Handa et al., 2019; sha, 2022). We include quantitative human teleoperation success rates for each of the tasks in

Table 3. We also plot the tracking performance for one representative task in each of the three categories in Figure 8. For opening ziploc bags (precise task), the tracking error is consistently under 2cm, with an average of 0.68cm. The error is comparable to the robotic arm's accuracy of 5-8mm.

For RAM insertion (contact-rich task), the tracking error is generally under 2cm with the exception of insertion-time, where high-amount of force is needed to secure the RAM in the slot. This is expected as our setup is not bilateral, and the tracking error defines how much force is applied through the low-level PID controller. Lastly for swinging open a plastic bag (dynamic task), both hands grasp the opening and fling twice to let air fill in the bag. We measure a peak velocity of 3m/s, and the rapid change in speed results in spikes of tracking error at around 25cm. We attribute this to a combination of teleoperation delay and motor torque limitations. While despite these large spikes in tracking error for forceful and dynamic motions, the teleoperation success rate remains high. As shown in table 3, our operator is able to insert RAM in 30 seconds for 9 out of 10 times, and swing open the plastic bag in 5 seconds 10 out of 10 times. We include a more detailed price & capability comparison in Appendix A.1, as well as more skills that *ALOHA* is capable of in Figure 9. To make *ALOHA* more accessible, we open-source all software and hardware with a detailed tutorial covering 3D printing, assembling the frame to software installations.

A.1. Comparing *ALOHA* with Prior Teleoperation Setups

In Figure 9, we include more teleoperated tasks that *ALOHA* is capable of. We stress that all objects are taken directly from the real world without any modification, to demonstrate *ALOHA*'s generality in real life settings.

ALOHA exploits the kinesthetic similarity between leader and follower robots by using joint-space mapping for teleoperation. A leader-follower design choice dates back to at least as far as 1953, when Central Research Laboratories built teleoperation systems for handling hazardous material (Jenness and Wicker, 1975). More recently, companies like Sarcos and RE2 (you, 2014) also built highly dexterous teleoperation systems with joint-space mapping. *ALOHA* is similar to these previous systems, while benefiting significantly from recent advances of low-cost actuators and robot arms. It allows us to achieve similar levels of dexterity with much lower cost, and also without specialized hardware or expert assembly.

Next, we compare the cost of *ALOHA* to recent teleoperation systems. *DexPilot* (Handa et al., 2019) controls a dexterous hand using image streams of a human hand. It has 4 calibrated Intel Realsense to capture the point cloud of a human hand, and retarget the pose to an Allegro hand. The Allegro hand is then mounted to a KUKA LBR iiwa7 R800. DexPilot allows for impressive tasks such as extracting money from a wallet, opening a penut jar, and insertion tasks in NIST board #1. We estimate the system cost to be around $100k with one arm+hand. More recent works such

as Robotic Telekinesis (Sivakumar et al., 2022; Arunachalam et al., 2022; Qin et al., 2022) seek to reduce the cost of DexPilot by using a single RGB camera to detect hand pose, and retarget using learning techniques. While sensing cost is greatly reduced, the cost for robot hand and arm remains high: a dexterous hand has more degrees of freedom and is naturally pricier. Moving the hand around would also require an industrial arm with at least 2kg payload, increasing the price further. We estimate the cost of these systems to be around $18k with one arm+hand. Lastly, the *Shadow Teleoperation System* is a bimanual system for teleoperating two dexterous hands. Both hands are mounted to a UR10 robot, and the hand pose is obtained by either a tracking glove or a haptic glove. This system is the most capable among all aforementioned works, benefitted from its bimanual design. However, it also costs the most, at at least $400k. *ALOHA*, on the other hand, is a bimanual setup that costs $18k ($20k after adding optional add-ons such as cameras). Reducing dexterous hands to parallel jaw grippers allows us to use light-weight and low-cost robots, which can be more nimble and require less service.

Finally, we compare the capabilities of *ALOHA* with previous systems. We choose the most capable system as reference: the *Shadow Teleoperation System* (sha, 2022), which costs more than 10x of *ALOHA*. Specifically, we found three demonstration videos (srcteam, 2021; 2022a;b) that contain 15 example use cases of the *Shadow Teleoperation System*, and seek to recreate them using *ALOHA*. The tasks include playing "beer pong", "jenga," and a rubik's cube, using a dustpan and brush, twisting open a water bottle, pouring liquid out, untying velcro cable tie, picking up an egg and a light bulb, inserting and unplugging USB, RJ45, using a pipette, writing, twisting open an aluminum case, and in-hand rotation of Baoding balls. We are able to recreate 14 out of the 15 tasks with similar objects and comparable amount of time. We cannot recreate the Baoding ball in-hand rotation task, as our setup does not have a hand.

B. Example Image Observations

We include example image observations taken during policy execution time in Figure 10, for each of the 6 real tasks. From left to right, the 4 images are from top camera, front camera, left wrist, and right wrist respectively. The top and front cameras are static, while the wrist cameras move with the robots and give detailed views of the gripper. We also rotate the front camera by 90 degrees to capture more vertical space. For all cameras, the focal length is fixed with auto-exposure on to adjust for changing lighting conditions. All cameras steam at 480×640 and 30fps.
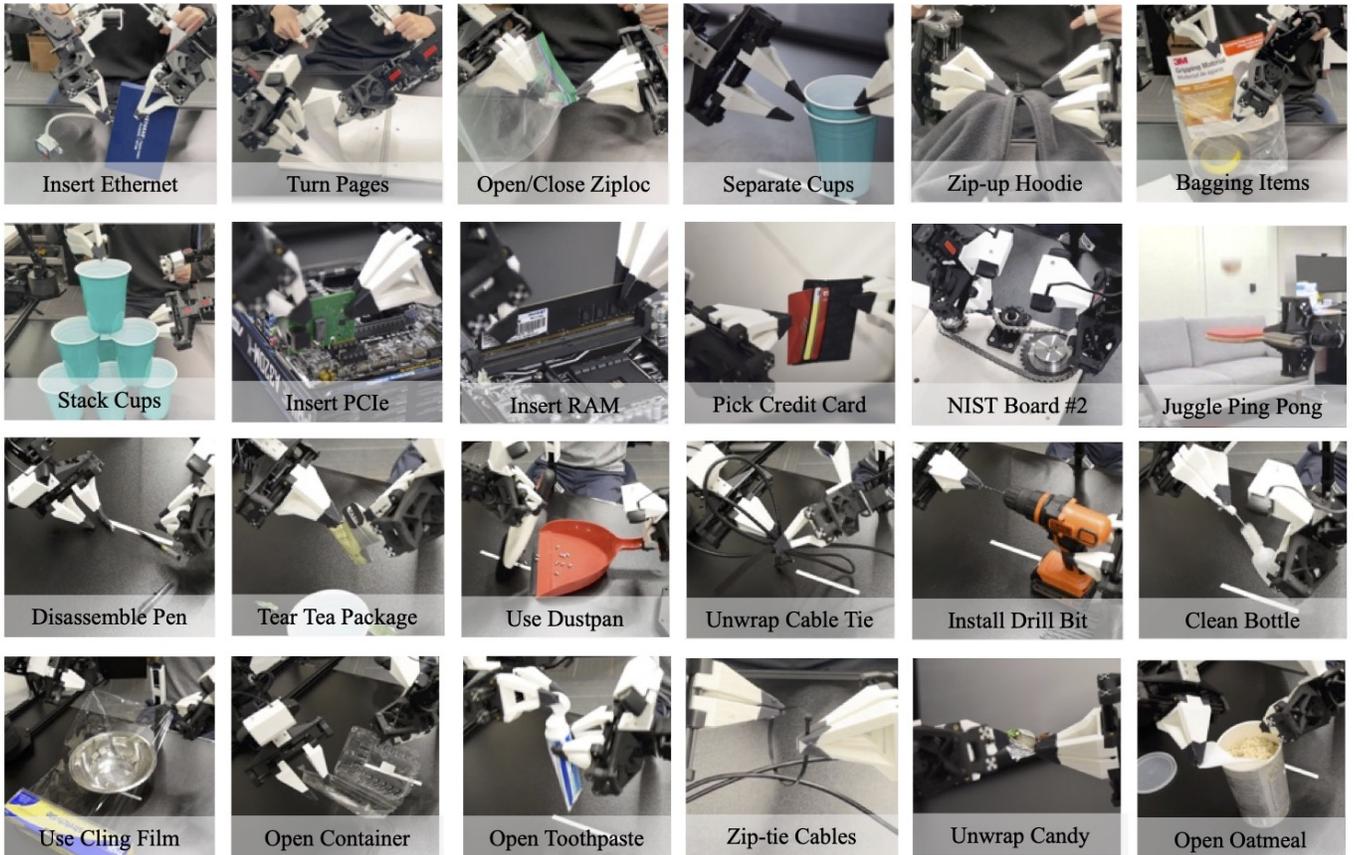
Figure 9: Teleoperation task examples with *ALOHA*.

C. Detailed Architecture Diagram

We include a more detailed architecture diagram in Figure 11. At training time, we first sample tuples of RGB images and joint positions, together with the corresponding action sequence as prediction target (Step 1: sample data). We then infer style variable z using CVAE encoder shown in yellow (Step 2: infer z). The input to the encoder are 1) the [CLS] token, which consists of learned weights that are randomly initialized, 2) embedded joint positions, which are joint positions projected to the embedding dimension using a linear layer, 3) embedded action sequence, which is the action sequence projected to the embedding dimension using another linear layer. These inputs form a sequence of $(k + 2) \times embedding\_dimension$, and is processed with the transformer encoder. We only take the first output, which corresponds to the [CLS] token, and use another linear network to predict the mean and variance of z's distribution, parameterizing it as a diagonal Gaussian. A sample of z is obtained using reparameterization, a standard way to allow back-propagating through the sampling process so the encoder and decoder can be jointly optimized (Kingma and Welling, 2013).

Next, we try to obtain the predicted action from CVAE decoder i.e. the policy (Step 3: predict action sequence). For each of the image observations, it is first processed by a ResNet18 to obtain a feature map, and then flattened to get a sequence of features. These features are projected to the embedding dimension with a linear layer, and we add a 2D sinusoidal position embedding to perserve the spatial information. The feature sequence from each camera is then concatenated to be used as input to the transformer encoder. Two additional inputs are joint positions and z, which are also projected to the embedding dimension with two linear layers respectively. The output of the transformer encoder are then used as both "keys" and "values" in cross attention layers of the transformer decoder, which predicts action sequence given encoder output. The "queries" are fixed sinusoidal embeddings for the first layer.

At test time, the CVAE encoder (shown in yellow) is discarded and the CVAE decoder is used as the policy. The incoming observations (images and joints) are fed into the model in the same way as during training. The only difference is in z, which represents the "style" of the action sequence we want to elicit from the policy. We simply set z to a zero vector, which is the mean of the unit Gaussian prior used during training. Thus given an observation, the output of the policy is always deterministic, benefiting policy
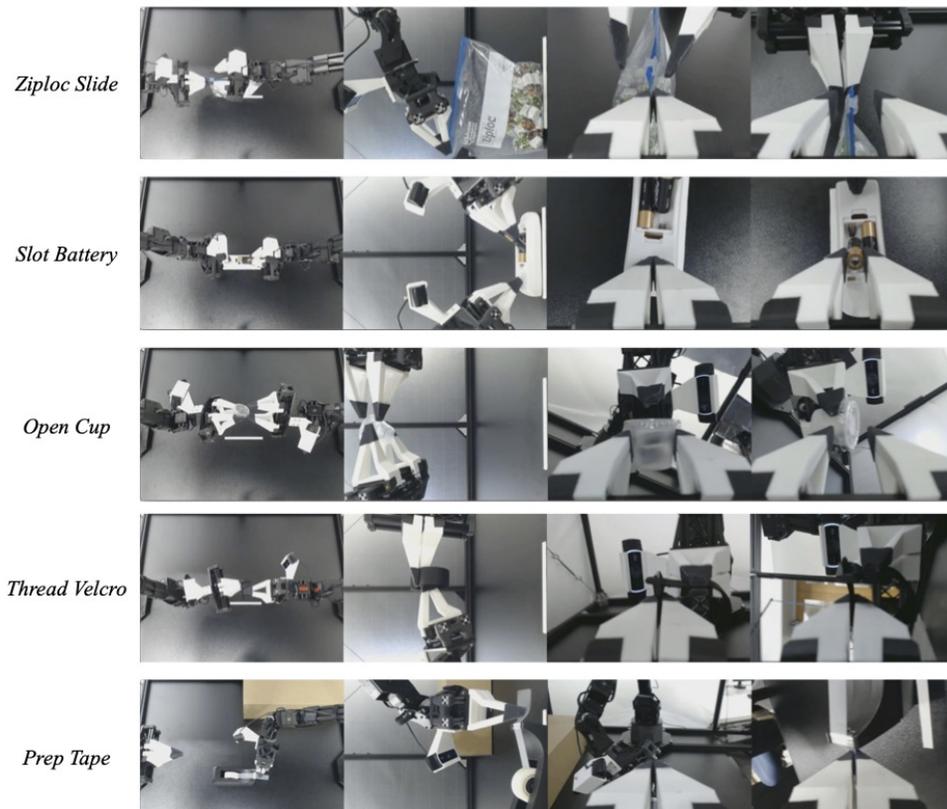
Figure 10: Image observation examples for 5 real-world tasks. The 4 columns are [top camera, front camera, left wrist camera, right wrist camera] respectively. We rotate the front camera by 90 degree to capture more vertical space.

evaluation.

D. Experiment Details and Hyperparameters

We carefully tune the baselines and include the hyperparameters used in Table 4, 5, 6, 7, 8. For BeT, we found that increasing history length from 10 (as in original paper) to 100 greatly improves the performance. Large hidden dimension also generally helps. For VINN, the k used when retrieving nearest neighbor is adaptively chosen with the lowest validation loss, same as the original paper. We also found that using joint position differences in addition to visual feature similarity improves performance when there is no action chunking, in which case we have state weight = 10 when retrieving actions. However, we found this to hurt performance with action chunking and thus set state weight to 0 for action chunking experiments.

E. User Study Details

We conduct the user study with 6 participants, recruited from computer science graduate students, with 4 men and 2 women aged 22-25. 3 of the participants had experience teleoperating robots with a VR controller, and the other 3 has no prior experience teleoperating. None of the participants used *ALOHA* before. To implement the 5Hz version of *ALOHA*, we read from the leader robot at 5Hz, interpolate in the joint space, and send the interpolated positions to the robot at 50Hz. We choose tasks that emphasizes high-precision and close-loop visual feedback. We include images of the objects used in Figure 12. For threading zip cable tie, the hole measures 4mm x 1.5mm, and the cable tie measures 0.8mm x 3.5mm with a pointy tip. It is initially lying flat on the table, and the operator needs to pick it up with one gripper, grasp the other end mid-air, then coordinate both hands to insert one end of the cable tie into the hole on the other end. For unstacking cup, we use two single-use plastic cups that has 2.5mm clearance between them when stacked. The teleoperator need to grasp the edge of upper cup, then either shake to separate or use the help from the other gripper. During the user study, we randomize the order in which operators attempt each task, and whether they use 50Hz or 5Hz controller first. We also randomize the initial position of the object randomly around the table center. For each setting, the operator has 2 minutes to adapt, followed by 3 consecutive attempts of the task with duration recorded.
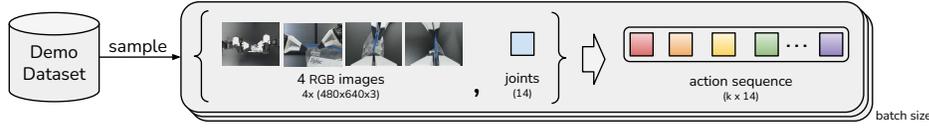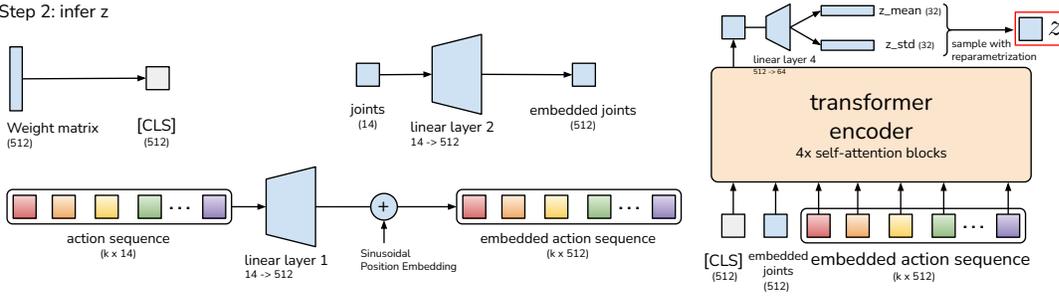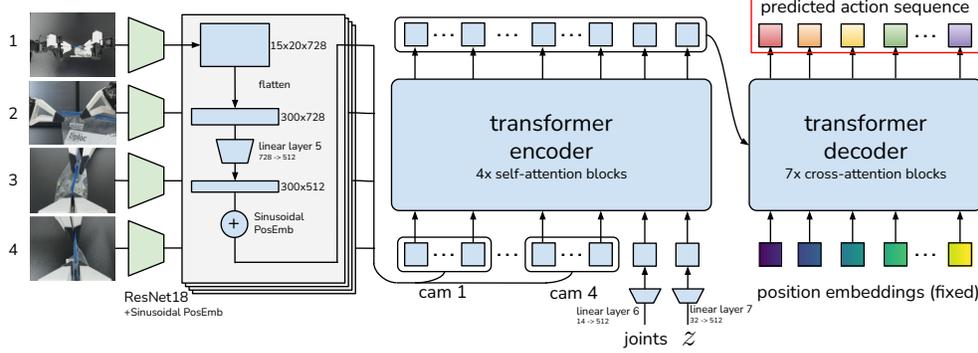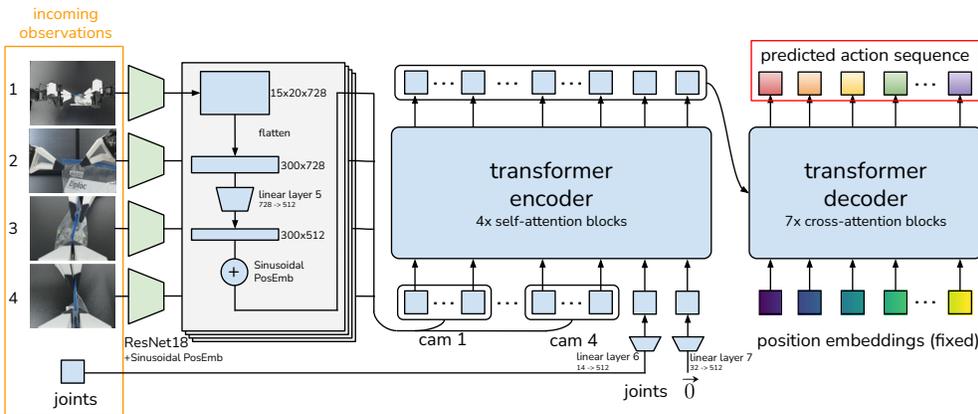
Figure 11: Detail architecture of Action Chunking with Transformers (ACT).

F. Limitations

We now discuss limitations of the ALOHA hardware and the policy learning with ACT.

Figure 12: The cable tie and cups for user study.

Hardware Limitations. On the hardware front, *ALOHA* struggles with tasks that require multiple fingers from both hands, for example opening child-proof pill bottles with a push tab. To open the bottle, one hand needs to hold the bottle and pushes down on the push tab, with the other hand twisting the lid open. *ALOHA* also struggles with tasks that require high amount of forces, for example lifting heavy objects, twisting open a sealed bottle of water, or opening markers caps that are tightly pressed together. This is because the low-cost motors cannot generate enough torque to support these manipulations. Tasks that requires finger nails are also difficult for *ALOHA*, even though we design the grippers to be thin on the edge. For example, we are not able to lift the edge of packing tape when it is taped onto itself, or opening aluminum soda cans.

Policy Learning Limitations. On the software front, we report all 2 tasks that we attempted where ACT failed to learn the behavior. The first one is unwrapping candies. The steps involves picking up the candy from the table, pull on both ends of it, and pry open the wrapper to expose the candy. We collected 50 demonstrations to train the ACT policy. In our preliminary evaluation with 10 trials, the policy picks up the candy 10/10, pulls on both ends 8/10, while unwraps the candy 0/10. We attribute the failure to the difficulty of perception and lack of data. Specifically, after pulling the candy on both sides, the seam for prying open the candy wrapper could appear anywhere around the candy. During demonstration collection, it is difficult even for human to discern. The operator needs to judge by looking at the graphics printed on the wrapper and find the discontinuity. We constantly observe the policy trying to peel at places where the seam does not exist. To better track the progress, we attempted another evaluation where we give 10 trials for each candy, and repeat this for 5 candies. For this protocol, our policy successfully unwraps 3/5 candies.

Another task that ACT struggles with is opening a small ziploc bag laying flat on the table. The right gripper needs to first pick it up, adjust it so that the left gripper can grasp firmly on the pulling region, followed by the right hand grasping the other side of the pulling region, and pull it open. Our policy trained with 50 demonstrations can consistently pick up the bag, while having difficulties performing the following 3 mid-air manipulation steps. We hypothesize that the bag is hard to perceive, and in addition, small

differences in the pick up position can affect how the bag deforms, and result in large differences in where the pulling region ends up. We believe that pretraining, more data, and better perception are promising directions to tackle these extremely difficult tasks.

learning rate	1e-5
batch size	8
# encoder layers	4
# decoder layers	7
feedforward dimension	3200
hidden dimension	512
# heads	8
chunk size	100
beta	10
dropout	0.1

Table 4: Hyperparameters of ACT.

learning rate	3e-4
batch size	128
epochs	100
momentum	0.9
weight decay	1.5e-6

Table 5: Hyperparameters of BYOL, the feature extractor for VINN and BeT.

learning rate	1e-4
batch size	64
# layers	6
# heads	6
hidden dimension	768
history length	100
weight decay	0.1
offset loss scale	1000
focal loss gamma	2
dropout	0.1
discretizer #bins	64

Table 6: Hyperparameters of BeT.

k (nearest neighbour)	adaptive
state weight	0 or 10

Table 7: Hyperparameters of VINN.

learning rate	1e-5
batch size	2
ViT dim head	32
ViT window size	7
ViT mbconv expansion rate	4
ViT mbconv shrinkage rate	0.25
ViT dropout	0.1
RT-1 depth	6
RT-1 heads	8
RT-1 dim head	64
RT-1 action bins	256
RT-1 cond drop prob	0.2
RT-1 token learner num output tokens	8
weight decay	0
history length	6

Table 8: Hyperparameters of RT-1.