# Multi-Stage Cable Routing through Hierarchical Imitation Learning

Jianlan Luo<sup>\*,†</sup>, *Member, IEEE*, Charles Xu<sup>\*</sup>, Xinyang Geng, Gilbert Feng, Kuan Fang, *Member, IEEE*, Liam Tan, Stefan Schaal, *Fellow, IEEE*, and Sergey Levine, *Member, IEEE* 

Abstract—We study the problem of learning to perform multistage robotic manipulation tasks, with applications to cable routing, where the robot must route a cable through a series of clips. This setting presents challenges representative of complex multi-stage robotic manipulation scenarios: handling deformable objects, closing the loop on visual perception, and handling extended behaviors consisting of multiple steps that must be executed successfully to complete the entire task. In such settings, learning individual primitives for each stage that succeed with a high enough rate to perform a complete temporally extended task is impractical: if each stage must be completed successfully and has a non-negligible probability of failure, the likelihood of successful completion of the entire task becomes negligible. Therefore, successful controllers for such multi-stage tasks must be able to recover from failure and compensate for imperfections in low-level controllers by smartly choosing which controllers to trigger at any given time, retrying, or taking corrective action as needed. To this end, we describe an imitation learning system that uses vision-based policies trained from demonstrations at both the lower (motor control) and the upper (sequencing) level, present a system for instantiating this method to learn the cable routing task, and perform evaluations showing great performance in generalizing to very challenging clip placement variations. Supplementary videos, datasets, and code can be found at https://sites.google.com/view/cablerouting.

#### I. INTRODUCTION

Complex, multi-stage robotic manipulation tasks often arise in robotic manipulation applications of practical interest: from home robots that need to prepare a meal to industrial robots that need to assemble a complex device. Many of the tasks we might want to automate in these settings consist of complex low-level behaviors and also require these behaviors to be sequenced appropriately to perform the overall task. This presents a major challenge: when a robot naïvely executes a sequence of primitive behaviors to perform a complex task, the probability of failing at the task grows multiplicatively with each stage. Advances in perception, control, and robotic learning can enable each stage of a task to be more performant, but as long as sequencing stages naïvely leads to such difficulties, elaborate multi-stage tasks that consist of a sequence of individually difficult primitives will remain out of reach. In this paper, we examine how hierarchical imitation learning

\*equal contribution



Fig. 1: Overview of our robotic cable routing system. The robot needs to route the cable through three clips separately by executing the sequence of primitives displayed on the right.

with levels of learned primitives and high-level sequencing can address this issue, in the context of a difficult multi-stage cable routing task. Our focus is on providing for robustness at both levels of the hierarchy, not by trying to construct perfect robotic skills that never fail, but by endowing both levels of the hierarchy with the ability to correct and recover from mistakes. We study the problem of multi-stage cable routing, where a robot routes a cable through a series of clips (see Figure 1). This task is representative of commonly occurring scenarios in manufacturing and maintenance, where a robot might need to route cables or hoses, and provides an interesting challenge for robotic manipulation: each stage of the cable routing task requires handling the deformable cable, reasoning about complex contact patterns between the cable and the clip, accounting for deformations, and also observing that the cable has been clipped into each clip successfully. At the same time, the higher-level sequencing of primitives might require retrying a clipping motion, securing the cable by pulling it taut, and dynamically deciding when to advance to the next clip. This task is, therefore, both practically relevant for industrial applications, and captures many of the essential characteristics of multi-stage manipulation discussed in the preceding paragraph, with challenges and ambiguity at both the lower and upper level necessitating intelligent controllers that combine perception and control, and recovery from failure.

Learning provides a powerful tool for handling complex, hard-to-model scenarios such as the manipulation of deformable cables, particularly when the robot controller needs to tightly integrate perception and control. Methods based on imitation learning (IL) [48, 7] and reinforcement learning

<sup>&</sup>lt;sup>†</sup>Correspondence to jianlanluo@eecs.berkeley.edu

Jianlan Luo, Charles Xu, Xinyang Geng, Gilbert Feng, Kuan Fang, Liam Tan, and Sergey Levine are with the Berkeley AI Research Lab (BAIR), Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA 94720 USA.

Stefan Schaal is with Intrinsic Innovation LLC, Mountain View, CA 94043 USA.



Fig. 2: Overview of the hierarchical cable routing policy. The **high-level primitive selection policy** takes the robot wrist and side camera observations, as well as the history of executed primitives as input, and outputs a categorical distribution to select the next primitive. The **low-level single clip cable routing policy** only uses the wrist camera observations and the robot state to output a Gaussian distribution of robot actions. This decomposition of our system into high-level and low-level policies allows us to collect data and train policies with large flexibility, thus enabling the robot to master sophisticated cable routing tasks while reducing the overall complexity of our system.



Fig. 3: Sequence of primitives chained together by a successful high-level policy rollout. The primitives in the sequence are color-coded following Figure 2. After successfully routing the cable through clip 3, the policy triggers go\_next a third and final time, signaling the end of the trajectory.

(RL) [37, 38] provide for the ability to learn vision-based controllers end-to-end from data and experience, but such methods are difficult to apply directly to complex multi-stage tasks. As we show in our experiments in Section VI, naïve end-to-end learning for tasks such as the one in Figure 1 fails even when provided with idealized demonstration data. However, learning methods can provide an excellent tool for learning basic primitives, such as inserting the cable into a single clip at a single position. Such primitives may not succeed every time, but they are relatively easy to obtain with minimal engineering effort (e.g., by using demonstration data) and can operate directly on raw image observations without any hand-designed perception system. Thus, if we can combine such primitives with an intelligent higher-level controller, we can make solving multi-stage tasks much easier. However, as mentioned previously, simply sequencing the primitives naïvely (e.g., inserting into each clip in turn) leads

to exceedingly low success rates when the individual primitives are imperfect, as the probability of failing on the entire task is the product of failure probabilities at each stage. We therefore propose to employ a learned policy for higher-level primitive sequencing that can compensate for imperfections in the individual skills, resulting in a robotic learning system that is more robust than the individual parts: the higher-level can decide to automatically trigger a second attempt if the clipping failed, tighten the cable to reduce slack and select intelligently when to move on to the next clip.

Our complete system for learning-based cable routing incorporates the idea of learned controllers for recovering from failure at both levels of the hierarchy. At the lower-level, we show how we can train a clipping controller from demonstrations that can insert a cable into a clip at various orientations, respecting translational and rotational invariances, while at the same time automatically recovering from small failures through an appropriate choice of demonstration collection strategy. At the higher-level, our high-level policy selects primitives to trigger at each stage, dynamically choosing when to move on to the next clip or reduce slack on the cable. This policy is also trained with demonstrations, which are significantly easier to gather as they require the human operator to simply manually trigger one of a discrete set of primitives at each stage. We find that the process of gathering these demonstrations can be further simplified by employing an interactive imitation learning strategy, similar to DAgger [68]. We also explore design decisions for making this system robust and practical, including the sharing of convolutional network representations between the lower and upper layer, encoding translational and rotational invariances into the lower-level controller with an appropriate choice of view-invariant representations, and using a learned word embedding layer to encode primitive execution history.

Our primary contribution is a system for hierarchical imitation learning applied to the cable routing task. We show that learned and scripted lower-level primitives can be combined via a learned higher-level policy into a multi-stage controller that can perform temporally extended cable routing tasks, compensating for failures in the lower-level by triggering appropriate primitives at each stage. We describe the design decisions behind our system and compare our approach to methods that employ a flat end-to-end policy, as well as a naïve scripted sequencing strategy, showing that our method not only outperforms them in terms of absolute performance but can also adapt to entirely novel situations using our finetuning mechanism with ten additional demonstrations.

#### II. RELATED WORK

Our hierarchical imitation framework combines concepts from imitation learning and hierarchical policies to address the challenges in multi-stage cable routing. We therefore survey prior work on learning-based visuomotor control, composition of skills, and deformable object manipulation.

Learning-based robotic control. Learning-based methods have been proposed for a range of robotic control problems, such as manipulation [67, 57, 48, 24, 47, 54, 95], navigation [71, 98, 72], and locomotion [46, 51]. Much of the progress has been made possible by reinforcement learning (RL) [36, 78, 66] and imitation learning (IL) [31] techniques, which enable robots to solve various tasks by learning from trial and error processes or human demonstrations. To solve long-horizon tasks, hierarchical RL and IL algorithms have been proposed to reuse primitive skills, reduce sample complexity, and facilitate generalization [5, 6, 64]. In these approaches, the hierarchy of the task can be discovered by learning high-level and low-level policies from online interactions [13, 16, 14, 3, 4, 20, 25] or offline dataset [44, 90, 43, 19, 41, 73, 52, 89]. Similar to several prior works on hierarchical imitation [90, 19], our method uses a fixed set of primitives and trains a high-level model to select skill indices. However, we describe a set of design decisions that make it feasible to extend hierarchical imitation to a complex multistage cable routing task, including techniques for maximizing the generalization and invariance of a learned low-level clip insertion primitive, and strategies for data collection. We also show that our high-level policy can be fine-tuned rapidly for out-of-distribution scenarios through an interactive data collection scheme.

Skill composition for multi-stage tasks. A large number of works investigate how to solve long-horizon tasks by composing primitive skills. With fixed control flows, behavior trees [9, 59] are widely used in robotics and control systems to switch between a finite set of tasks in a modular fashion. Built on pre-defined action modes and the symbolic states of the environment, Task and Motion Planning (TAMP) [22, 35, 15, 21] uses task planners to select feasible task plans and motion planners to generate trajectories for solving longhorizon tasks. An increasing number of works use learned state estimators [60, 18, 11], planners [12, 84, 17, 65], and skill trees [42] to apply planning and sequencing to realworld environments where the ground truth states and models are not provided. Such methods' performance usually relies on the accuracy of the state estimators and model of the environment. It's very challenging to build these components for the considered task; for instance, it's not obvious how to build an effective state estimator for deformable cables, particularly in the presence of occlusions that tend to occur with clips and with the arm. In contrast to these works, our method learns both the high-level and low-level policy to solve multi-stage cable routing tasks. Recent works also use natural language to compose hand-designed or learned skills using pre-trained large language models to leverage prior knowledge about the tasks [2, 76, 33]. Our work does not utilize any additional sources of data for pre-training and performs the task only based on visual inputs.

Deformable object manipulation. The manipulation of deformable materials, such as cloth, rope, and liquid, presents a major challenge for robots [28, 93]. Several works have designed motion planners and controllers based on models of deformable objects for specific domains [77, 82, 27]. Most of these methods rely on extensive domain expertise, and their design is typically specific to a particular type of object, material, and task. Recently, an increasing number of data-driven methods have been proposed to train robots to manipulate deformable objects through physical interactions. [92, 53, 30, 50, 23] learn to estimate the configuration and model the dynamics of cloth and ropes based on visual inputs. In [49, 56, 74, 79, 87], multi-modal sensory inputs, such as haptics and audio, are utilized to enable a robot to perform challenging deformable object manipulation tasks. These works have enabled robots to perform a wide range of tasks such as folding clothes [58, 45, 86, 91], rope reshaping [63, 97, 92, 34], knot tying [85, 62], rope untangling [81, 80], dynamics rope manipulation [8, 94], etc. In this paper, we are interested in the application of cable routing with potential applications in industrial tasks. Prior work [74, 83] has studied similar problems using hand-designed controllers and planners. Many previous works on rope manipulation such as Yan et al. [92] and Schulman et al. [69] model ropes as linear objects and then estimate the best matching cable shape from perceptual inputs, which assumes full visibility of the

cable in the scene and can be problematic in our setting due to occlusions introduced by clips. As far as we know, ours is one of the first works that use learning-based methods to enable robots to route cables across multiple clips in unseen scenarios based on visual inputs.

## III. SYSTEM AND TASK SETUP

Figure 4 provides an illustration of our robotic manipulation setup for cable routing. Our system consists of one Franka Panda robot, two Basler RGB cameras mounted on the robot's end-effector, and two additional Intel RealSense cameras mounted on the work cell, one providing a top-down view and the other providing a front-facing view. The endeffector cameras provide a close-in view of the cable and facilitate robust insertion of the cable into clips, while the static cameras provide an overview of the workspace to aid in routing the cable through multiple clips while observing the resulting deformations. We only use the RGB component of the Intel RealSense output in our implementation. A space mouse is also used for collecting human demonstrations.

While one end of the cable is fixed on the table, the task is for the robot to pick up some part of the cable and then route it through all the clips present on the table sequentially, as shown in Figure 1. Each instance of the task has a different placement of the clips in terms of their position and orientation.

Routing cables through clips can be difficult: since the cables are deformable, they can take an infinite variety of shapes, and their behavior is affected by the complex interaction between the gripper, the cable, and multiple other clips that the cable might already be inserted into. To correctly slot the cable into a new clip, the robot must keep track of the point on the cable that contacts the clip while grasping it at a different location, which necessarily requires closing the loop on visual perception. So it is crucial to learn a robust reactive policy that can not only perform reasonably routing a single clip but also employs recovery and retry mechanisms in terms of failure. This makes learning-based control particularly appealing, as it can enable closing the loop on vision with minimal manually encoded domain knowledge.

# IV. CABLE ROUTING VIA HIERARCHICAL IMITATION LEARNING

It is very difficult to learn a cable routing policy for the multi-stage task described in the previous section that never fails. Instead, we could design a hierarchical architecture where a higher-level policy compensates for the deficiency of lower level skills, sequencing appropriate primitives to retry, repositioning the cable, and dynamically deciding when to proceed to the next clip. Merely sequencing primitives naïvely is unlikely to lead to much benefit in this case, but a strategy that selects the primitives dynamically to compensate for mistakes could lead to significantly better performance.

In this section, we present our complete system of sequential multiple-clip cable routing. Our policy is structured in a hierarchical fashion: at the low level, we have several primitive policies that individually perform some part of the cable routing task. These primitives comprise a variety of simple



4

Side view (Intel Realsense) Top view (Intel Realsense) Wrist view 1 (Basler) Wrist view 2 (Basler)

Fig. 4: **Top left**: Picture of the entire system setup depicting the robot arm mounted to a frame along with a metal board where three clips are fixed. (a): White 1/4" thick poly cord with colored segments. (b): Clip with 1cm wide opening. (c): Eye-in-hand camera mounted to the wrist of the robot. **Bottom first from left**: Side view of the workspace from the RealSense camera. **Bottom second from left**: Top view of the workspace from RealSense camera. **Bottom second from right**: Wrist view 1 from Basler camera. **Bottom first from right**: Wrist view 2 from Basler camera.

scripted behaviors for picking up and moving the cable, as well as a learned clip insertion primitive that is trained to put a grasped cable into a single clip. This single clip policy, which addresses by far the most challenging of the low-level skills, is designed so as to make it maximally invariant to clip position and orientation. At the higher level, a primitive selection policy is trained to integrate these primitives together to perform the entire multi-stage routing task, using visual observations to select which primitive to trigger. This higher level policy uses a more global view of the scene to route the cable through multiple clips. Particularly unfamiliar or unusual clip configurations that differ significantly from those seen in the training data can still cause the higher level policy to fail, even when the invariant low-level skills generalize. We therefore additionally explore how the higher level policy can be finetuned efficiently via an interactive data collection procedure to adapt it to especially difficult configurations for which zero-shot generalization fails.

## A. Low-Level Clip Routing Policy

We will first describe our low-level single clip routing policy. The higher-level policy in our system can make use of a number of primitives, the rest of which we will summarize in Section IV-B, but the most important of these is the policy that attempts to insert the cable near the current grasp point into the closest clip. This primitive performs by far the most complex low-level task, which requires carefully manipulating the cable into the clip while holding it at a different location. It must use visual feedback since proprioceptive readings provide the endeffector position but not the cable configuration. It must also generalize to a variety of clip positions and orientations, and handle a variety of cable configurations (though the higherlevel policy can compensate somewhat by deciding to perturb the cable if it is in a particularly unfavorable shape). For these reasons, we use an end-to-end imitation learning approach to train this primitive. The neural network architecture for this policy is illustrated in Figure 2. In this section, we describe the training procedure and dataset for this policy, as well as techniques we employ to maximize the invariance and generalization of this policy with respect to variation in the clip position and orientation, which are based on using wristmounted cameras and data augmentation.

End-to-end imitation learning of single clip policy. The single clip policy assumes that the cable has already been grasped, though it is trained to be robust to some variability in the grasp point. In the full system, the grasp will be performed by a separate primitive. The single clip policy needs to insert the cable through a single clip. We first collect a dataset by teleoperating the robot to perform the task in various locations and for clips with various orientations. The dataset consists of 1442 trajectories, but these trajectories are relatively quick to collect, since they do not require regrasping. Each trial is less than 10 seconds in length. At the beginning of each trial, we randomize the poses of the robot's end-effector and clip. The teleoperator records two types of demonstrations: "successful" and "recovery" trials. In about 800 of the trials, the demonstrator successfully inserts the cable into the clip. In about 600 of the trials, the demonstrator intentionally moves the end-effector into some failure state (e.g., a partial or failed insertion), and then demonstrates a recovery, as shown in Figure 5. These recovery trials are intended to ensure that the learned policy is more robust to small local mistakes. Both sets of demonstrations are combined into one dataset, which we denote  $\mathcal{D} = \{(\mathbf{o}, \mathbf{a})\}$ , where **o** is the sensor observation, which we will describe later in this section, and a is the robot's action. The goal of imitation learning is to find a parameterized policy  $\pi_{\theta}$  that can maximize the likelihood of the current dataset  $\mathcal{D}$ :

$$\theta = \arg \max_{\theta} \mathop{\mathbb{E}}_{\mathbf{o}, \mathbf{a} \sim \mathcal{D}} \left[ \log \pi_{\theta}(\mathbf{a} | \mathbf{o}) \right]$$
(1)

This corresponds to a standard behavioral cloning method and can be implemented with simple supervised learning methods for the architecture in Figure 2. We use Adam optimizer [40] with a learning rate of 3e - 4. More details can be found in Appendix C.

**Generalization via view-invariant representations.** The full multi-stage cable routing task requires being able to execute the clip insertion primitive (as well as other primitives) in a variety of locations depending on the placement of the clips. Therefore, it is critical to ensure that the clip insertion skill generalizes effectively over clip configurations. We found that we could significantly improve this by training the policy to



Fig. 5: A recovery trial. The leftmost frame shows an initial failure state, where the rope has missed the slot and is on the side of the clip. The rest of the demo, depicted in the remaining frames, shows a recovery resulting in successful insertion into the slot.

act in the frame of reference of the robot's gripper with wrist cameras. This effectively puts all of the low-level policy's sensory observations and control commands into the frame of the end-effector. We use two eye-in-hand cameras as shown in Fig. 4, as they are more robust to view shifts [55, 29, 1]. For the robot's proprioceptive information, such as end-effector pose, we reference them w.r.t. the randomized reset pose of each episode. The full sensor observation o therefore consists of images from two wrist cameras and the TCP pose, which we find sufficient to only take the z component since other spatial information can be inferred from the wrist cameras. The control command is a 4-D Cartesian twist with full translation components and the rotation component around the z-axis. We detail the exact procedure of calculating these quantities by doing frame transformation in the appendix A.

**Policy network architecture.** The single clip routing policy is represented by a deep neural network, as illustrated in Figure 2, which in the end produces a Gaussian distribution over the action at each step given the current observation o. The input images from the two wrist cameras are first fed into two convolution neural networks to obtain the corresponding feature embedding vectors. We use ResNet18 [26] with group normalization [88] instead of the original batch normalization [32] for simplicity. As mentioned in the previous section, we only take the z component of the end-effector pose, which we found practically sufficient in terms of additional spatial information. We embed the z-coordinate by passing it through an additional fully connected layer to get a higher-dimension embedding. We concatenate the 2 camera embedding vectors and the z-coordinate embedding vector and pass the result through a 3-layer multi-layer perceptron (MLP) network to obtain the final mean and log-standard deviation of the resulting Gaussian action distribution. This model can then be trained end-to-end via maximum likelihood on the demonstration dataset, as described previously.

**Data augmentation.** We employ data augmentation techniques [75] to facilitate the generalization capability of the learned policy to deal with image view shifts at test time such as variation in lighting conditions, small perturbations in camera pose, and variability in the objects in the scene. In order to balance the augmentation effectiveness and the complexity of tunable hyperparameters, we employ the RandAugment [10] technique, where a set of image transforms are chosen at random and applied sequentially each time to obtain the augmented image. Specifically, we apply two transforms sequentially and choose the strength of augmentation to be nine. We visualize the randomly augmented image in Figure 6. With image augmentation, the data becomes much more diverse, allowing us to train more robust skills with smaller datasets.



Wrist Views

Augmented Wrist Views after RandAugmentation

Fig. 6: Visualization of image augmentation. Left: original images from both wrist cameras when the rope is being grasped. Right: randomly sampled augmented images using RandAugmentation during training. Data augmentation significantly alters the images during training, enabling the policy to be more robust to distribution shift at runtime.

#### B. High-Level Primitive Selection Policy

Policy architecture overview. Similarly to the low-level cable routing policy, the high-level primitive selection policy is also fully parameterized by a deep neural network and trained endto-end, as illustrated on the left side of Figure 2. However, unlike the low-level policy, which directly commands lowlevel actions on the robot, the high-level policy performs planning from a global perspective, and therefore we make the following design choices to suit its purpose. First, in order to choose the right clip to route, the high-level policy must be able to see the entire cable and all clips. To this end, we feed the side camera image into an additional ResNet18 network to obtain a third embedding vector, along with the two embedding vectors from the robot wrist cameras. For sample efficiency, we reuse the pre-trained ResNet18 parameters from the low-level routing policy and freeze them for training highlevel policy. It is very important for the high-level policy to utilize a history of recent observations so that the primitive sequence for the current clip can be Markovian. For example, if a particular primitive fails, the policy might choose some corrective action, such as perturbing the cable, so that it does not fail in the exact way again. Even though the task is typically fully observed from the side camera perspective, we found that this addition improved the high-level policy's ability to correct mistakes. We, therefore, augment the input of the high-level policy with the history of at most 6 primitives chosen at prior high-level policy steps for the current clip. We concatenate the indices of past primitives into a sequence and feed it into a learned word embedding layer [61] to obtain a vector representation. We concatenate this vector with the embedding vectors of the three cameras and feed the result into a three-layer MLP to obtain the final logits for choosing the primitive.

Primitives. The full set of primitives available to the highlevel policy consists of the learned clip insertion primitive described previously, as well as three scripted primitives: Pickup, Perturb cable, and Go next. We found that these three additional primitives could be solved via existing robotic solutions and did not require learning, as they perform relatively simple tasks, and the high-level policy could reliably compensate for their imperfections with appropriate choices at the upper layer of the hierarchy. Note that our hierarchical system is designed to be modular, so any of these primitives can be replaced with other solutions as well.

The clip configuration can be fully represented by a list containing the estimated position of each clip in the desired order, which can be estimated with a standard computer vision detector. We use these estimated positions to parameterize primitives, though they are not visible to the routing policy itself. The high-level policy proceeds through each clip in the list in order. At each clip, it can trigger a variety of primitives, and select when to advance to the next clip (incrementing the current clip index). All of the primitives operate on the current clip that the high-level policy is handling, and the history of previously selected primitives is reset each time the high-level policy advances to the next clip.

Pickup: The Pickup primitive is used for picking up the cable at a particular position and holding it within the gripper. We designed this separate Pickup primitive instead of hardcoding it as a precursor to Route to allow re-trying the Route without having to drop and re-grasp the cable. As shown in Fig. 4, different segments of the cable are marked with different colors. We move the arm out of the way to avoid occlusion before using the top camera as shown in Fig. 4 to detect the colored marker corresponding to the current clip to route, and then choose the center point of the detected region to construct a pick-up pose in the camera frame. This pose is then converted back to the robot's base frame by using the top camera's extrinsic calibration information and a fixed z-position based on table height rather than the depth map from the RGBD camera because we find it to be inaccurate in practice for our thin cable. Note that this is the only camera that needs to be calibrated.



(b) Pickup

Fig. 7: The pickup primitive uses image segmentation to detect the grasping point illustrated by the green dot in 7a and picks up the rope.

Perturb cable: Repeated application of various primitives can cause the cable to end up in a shape from which other primitives will consistently fail. One of the benefits of our hierarchical design is that the high-level policy can detect this by leveraging the history input and camera observations. In this case, the Perturb\_cable primitive can be used to rearrange the cable into a new shape, which can get it out of a pathologically difficult configuration. We use the same detection method as described for the Pickup primitive to grasp the cable at specific locations, and then apply pre-defined

motions to change the shape of the cable before releasing it. Although picking the right perturbation motion w.r.t. any given cable shape is generally difficult, we found in practice it's sufficient to just stretch the cable along one direction so the amount of slackness in the cable can be reduced.



(a) Before Perturb\_cable (b) After Perturb\_cable

Fig. 8: Perturb\_cable applies a pre-defined motion to a specified point on the cable, changing the cable shape from 8a to 8b.

Go\_next: The Go\_next primitive advances the task to route the cable into the next clip. This primitive releases the cable and moves the robot's end-effector to a position that is close to that of the next clip in the sequence, applying a small random perturbation to offset the end-effector from the clip. The random offset is sampled from the same distribution that we use to determine initial states for training the single-clip policy, as described in Sec. IV-A. That way the end-effector pose ends up within the distribution of relative poses seen during training of the low-level single-clip skill. Note that this primitive moves the end-effector while the cable is grasped in hand.

**Training the high-level policy.** We now discuss how we train the high-level primitive selection policy. Denote images from three cameras (the two wrist cameras and the side camera) as  $I_1, I_2, I_3$ . We pass these images through the ResNet18 encoders and get three embeddings  $e_1, e_2, e_3$ . Let h be the list containing the indices of up to six primitives that have been executed so far for the current clip, with the number 0 as paddings to fill the empty slots if the length of the history is less than six. Each element is a categorical variant that takes on one of four values (the learned single-clip skill and the three scripted primitives). The high-level policy  $\pi_{\phi}$  takes these image embeddings and primitive history as inputs, together with the z component of the end-effector pose. It outputs a categorical distribution over primitives, represented as a fourway softmax. That is,

$$\pi_{\phi}(\cdot|e_1, e_2, e_3, h, z) \sim \text{Categorical}_4(\cdot) \tag{2}$$

To train this policy, we also collect a dataset of about 250 trajectories from a human demonstrator selecting the primitives to route a cable through the clips, using a keyboard. To make the learned high-level policy more robust to local variations when making a decision, we further augment the dataset by relabelling the nearby frames where the policy needs the select primitives. Specifically, we label the adjacent three to five frames with the same primitive selection choice from the human. This dataset can be significantly smaller than the one we use for the low-level policy, since the image encoders are reused, and the action space of the high-level policy is significantly simpler. This dataset is used to train the high-level policy via a standard maximum likelihood behavioral cloning loss (i.e., a cross-entropy loss, since the output is categorical). A sample rollout of the policy is visualized in Figure 3.

## C. Interactive Finetuning

While our system provides a robust framework for tackling the cable routing problem with a variable number of clips in different locations, it is still possible for these pre-trained policies to fail to generalize to a novel condition, such as an arrangement of clips that is outside the range of clip placements seen in the training data, or a new number of clips. Additionally, it's often very desirable to have a high success rate for such systems to be deployed in real industrial settings, and it could be entirely possible that our system can't meet these stringent requirements of reliability initially. In these scenarios, we can further finetune our policy for better performance with a small amount of interactive training for the high-level policy. Specifically, we adopt the HG-Dagger [39] method, where the high-level policy attempts to complete the task under human supervision. When the policy makes a mistake and selects the wrong primitive, the human operator will intervene by overriding the high-level policy with the correct primitive, repeating this process until the episode is completed successfully. This interactive training procedure is easy and less disruptive since a human can immediately tell if a policy's selection will make sense without having the robot actually execute it. After the policy outputs a primitive index, this integer number will be displayed on the computer screen waiting for human input; if the current selection will result in a non-recoverable state, the human will override the policy's selection; otherwise, we'll advance the policy's selection. For instance, if the policy decides to skip routing the current cable and go to the next one, the human operator can override this command right after seeing it from the computer screen. We illustrate such a procedure in Fig. 10.

In our interactive finetuning experiments, we use this procedure to collect up to ten additional trials with human corrections for each new scenario. Note that we don't use the previous dataset collected for the high-level policy when finetuning to a new configuration. Rather, we finetune the weights of the MLP solely on this newly collected dataset with a learning rate warm-up. Further details can be found in Appendix C.

#### V. DATA COLLECTION AND OPEN-SOURCE RELEASE

In this section, we detail our data collection procedure for both the low-level policy and high-level policy as well as the now publicly-available dataset resulting from this research.

#### A. Data Collection

For all of the experiments, we fixed one end of a cable to the table and fix the clips in the randomization areas defined in Fig. 9.

**Single clip cable routing data collection.** For each of the three clips, we vary its position within the constraints described in Fig. 9 and collect a total of 1442 demonstration

Fig. 9: Each of the three clips is placed randomly within a 12.5 cm by 15 cm rectangular region, with the regions spaced 7.5 cm apart. Additionally, each clip can be rotated between 0 and 45 degrees.

trajectories via a human expert teleoperating the robot at 5Hz. As mentioned in Sec. IV-A, of the 1442 demonstrations, about 800 start within a region in space, measuring 10cm by 10cm by 2cm in the x, y, and z direction, centered 10cm above and 5cm in front of the clip. The other 600 or so demonstrations start on the lower end of the table, where the starting position was not precisely controlled to demonstrate recovery.

**High-level data collection.** After training the single clip routing policy and the other primitives, we collect demonstrations for the high-level policy by having a human expert trigger primitives in sequence to perform the complete multi-stage cable routing task. We ask humans to use combinations of primitives to route cables when there are one clip, two clips, or three clips on the table. In each demonstration, the cable starts lying flat on the table in an arbitrary shape, and the expert inputs the next primitive for the robot to execute until the task is complete. We record the sensory information of entire trajectories when executing low-level primitives. We also augment the dataset by labeling the adjacent states of the actual state where the human makes a selection with the same executed primitive index.

#### B. Open-Source Dataset

To facilitate the reproducibility of our work, we release the datasets used to train the low-level and high-level policies hosted on our website: https://sites.google.com/view/ cablerouting. The dataset used to train the low-level policy consists of human teleoperated robot cable routing trajectories. Each trajectory contains around 20 time steps, and each time step contains a tuple of four robot camera images, robot configuration state vector, and the human teleoperator's commanded action. The entire low-level policy dataset contains 1442 such trajectories, each trajectory is around 3-5 seconds long. The high-level policy data consists of high-level trajectories of robot observations between primitive executions, where one timestep in a trajectory corresponds to one observation and the index human selected primitive. This dataset contains 257 such trajectories, where one full such trajectory is roughly 1 minute. Furthermore, to ensure that our robot setup can be reproduced, we also release the CAD file of the plastic clips we used, which can be easily produced by common 3D printers.

# VI. EXPERIMENTAL EVALUATION

In this section, we describe the experiments we conducted to evaluate our hierarchical imitation learning system for cable



Fig. 10: After the low-level primitives are acquired, data for the high-level policy can be collected by a human selecting the appropriate primitive to execute after the previous has autonomously finished until the cable is successfully routed through all the clips.

routing. One central premise of our method is the necessity of adopting hierarchical structures. To examine this claim, one natural question to ask is: can such long-horizon behavior be acquired via "flat" imitation learning approaches? Hierarchical methods will be much more convincing choices if they can outperform recent state-of-the-art "flat" methods [70, 96]. Another goal of the experiment is to validate the effectiveness of the main design choices in our system. Dissecting these design choices will facilitate the improvement and adoption of such a learning-based system in dealing with deformable objects in a much more general sense. Finally, to validate the capability of our system, it's crucial to examine its performance in terms of generalization, both in terms of zero-shot performance and in terms of fine-tuning with a small amount of human guidance. With these considerations in mind, our experiments study the following questions:

- How effective is our low-level clip insertion policy compared to baseline approaches, and how important are the specific design choices we made in the training process for this policy?
- How effective is our method compared to other imitation learning methods that compose low-level skills in different ways?
- How well does our high-level policy generalize to novel clip arrangements?
- How efficient is our finetuning scheme in quickly handling out-of-distribution clip arrangements as well as improving the system to achieve desirable performance?

### A. Experiment Setup

We evaluate the low-level and high-level policies under several different scenarios. For the trained low-level policy, we test its performance in the case of single-clip routing with different clip placement variations. For that, we ask the robot to repeatedly execute the trained neural network control policy autonomously, starting from various initial arm configurations. The high-level policy is evaluated on multiple runs of full cable routing tasks, and each such trial is only marked as successful if the cable is routed through all of the clips correctly.

# B. Evaluating the Low-Level Single Clip Routing Policy

We evaluate the low-level routing policies by first sampling five clip positions from the same distribution as used for training. For each of the clip positions, we identify two different cable shapes: one that curves toward the clip opening and one that curves away from it. From experience, the former is much easier than the latter for both learned policies and human experts. In the hard configurations, the cable tends to make a small radius curve of close to 90° near the grasping point perpendicular to the direction of the narrow straight opening on the clips. Without another arm to manipulate the curvature of the cable, this configuration makes the cable much harder to be routed through for both learned policies and human experts. An example of the different shapes is shown in 11. For each combination of clip position and cable shape, we roll out each policy five times, for a total of 50 attempts per policy. Before the start of each rollout, we position the end-effector approximately 5 centimeters in front of the clip with the cable grasped in the fingers. We roll out the policy for a total of 50 timesteps and consider the trials successful if a section of the cable is completely in the clip.



(a) Curving toward clip

(b) Curving away from clip

Fig. 11: Wrist view of the two different cable shapes for the same clip. 11a is an easier configuration to route than 11b because the cable makes a smaller angle with the opening of the clip and is easier to align.

Our learned low-level policy successfully inserts the cable into the clip on 18 out of 25 trials with the easier cable shape, and 5 out of 25 trials with the harder cable shape across different clip positions, for an overall success rate of 46%. This is a reasonable number as compared to that of a human operator: during data collection, we filter out failed episodes, but roughly a human can achieve a 60% success rate by teleoperating the robot. While the learned policy is far from perfect, we will see below that it drastically outperforms a scripted alternative and various ablations, though it still requires an intelligent higher-level policy to compensate for its failures and mistakes.

**Comparison to a scripted policy.** In principle, a central benefit of training the low-level clip insertion policy end to

Method	Easy	Hard	Overall
Scripted	7 / 25	3 / 25	10 / 50
Ours	<b>18 / 25</b>	<b>5 / 25</b>	23 / 50

TABLE I: Low-level single-clip policy comparison against a scripted baseline. Our learned policy outperforms the hand-designed alternative.

end is that it can close the loop on visual perception and observe the cable as it is being inserted into the clip, in contrast to simple scripted strategies that blindly match end-effector offsets from the clip. To evaluate whether we improve on such simple baselines, we compare our low-level policy to a scripted strategy that uses the ground-truth clip position and orientation, which is not available in our vision system but does not attempt to perceive the cable itself. This scripted strategy follows a series of predefined waypoints relative to the clip to attempt to insert the cable and then wiggles the cable (using normally distributed noise added to target positions) to attempt to insert it. The results of this comparison are shown in Table I. Although the scripted policy even uses privileged information that is otherwise not available to our policy, we can see our policy has twice the success rate. This confirms that the single clip insertion task is challenging, and also suggests that the additional visual perception enabled by our end-to-end policy is quite helpful.

Ablation experiments. To examine the effectiveness of the design choices we made in Sec. IV, we conduct extensive ablation experiments on the low-level policy. The results are shown in Table II. We first note the drastic drop in performance when we omit image augmentation, highlighting its utility in facilitating robustness. Additionally, using separate ResNet18 encoders for the two eye-in-hand cameras in the low-level policy provides marginal benefit. This is reasonable and suggests that the network might have enough capacity to digest current inputs. Replacing the eye-in-hand cameras with the external side camera view yielded a drastic drop in performance, indicating the importance of view-invariant representations. Finally, excluding the correction demonstrations from the offline dataset caused a noticeable drop in performance, with many of the failures involving the cable missing the clip and the policy failing to lift up the cable to recover and retry routing within the 50 rollout timesteps.

Design Choice	Easy	Hard	Overall
No image augmentation No shared ResNet Side camera view No correction data	9 / 25 <b>18 / 25</b> 5 / 25 15 / 25	2 / 25 3 / 25 1 / 25 3 / 25	11 / 50 21 / 50 6 / 50 18 / 50
Our full method	18 / 25	5 / 25	23 / 50

TABLE II: Ablation experiments for the low-level policy, showing performance on single clip insertion after ablating each design choice. The variant with decoupled ResNet encoders performs somewhat similarly, but the other variants are significantly worse than our full design.

## C. Evaluating the High-Level Policy for Cable Routing

We first evaluate our full hierarchical system on one-, two-, and three-clip routing tasks. For a given number of clips, we sample each clip position from the same distribution as used for our training data. For each number of clips, we evaluate four randomly sampled configurations and six trials each, for a total of 24 trials. At the start of each trial, we place the cable flat on the table. Then we roll out the high-level policy by executing each primitive that it outputs until the policy outputs go\_next at the last clip. We consider the trial successful only if the cable has been routed through all of the clips that are currently on the table.

One Cli	ip Two Clips	Three Clips	Total
Success Rate   19 / 24	4 14 / 24	12 / 24	45/72

TABLE III: Hierarchical policy evaluation for different numbers of clips, evaluating in-distribution test scenarios.

The results are presented in Table III. Fig. 12 gives a visualization of how we conduct this randomization procedure. It's important to note that the performance of our system doesn't drop exponentially as the number of clips increases from one to three. Rather, it maintains a reasonable performance in the most challenging three-clip scenario compared to the one-clip case; though the task there is much harder. This resonates with our key motivation that the smart combinatorial use of primitives can compensate for the deficiency of each individual one; so that the overall performance of the resulting system will largely not be subject to compounding errors in longhorizon tasks in a mechanical way.



Fig. 12: The red boxes depict the randomization areas where each of the three clips could be placed independently from one another, creating large variations in the subsequent cable shape.

**Baseline comparison.** Table IV compares the experimental results of our model with the following baselines:

1) State machine: The primitives are composed together using a state machine that sequentially performs pickup, attempts to route through the current clip twice before prompting go\_next, and then executes perturb cable before the pickup for the next clip. This sequential composition was inspired by strategies that the authors found to work reasonably efficiently at successfully completing the routing task during demo collection. Note that this strategy is not as naïve as simply executing the low-level clip insertion policy repeatedly, and actually includes a scripted strategy for correcting mistakes. While it is indeed possible to always develop a better state machine, we found it increasingly complex in dealing with combinatorial variations; eventually became infeasible under practical constraints.

Model	Success Rate
State Machine	5 / 24
End-to-End BC	0 / 24
End-to-End BeT	0 / 24
End-to-End ACT	0 / 24
Hierarchical Imitation (ours)	12 / 24

TABLE IV: Comparisons on the full multi-stage routing task. Our full method significantly outperforms both end-to-end (non-hierarchical) baselines and a hierarchical method with a non-learned higher-level state machine.

- 2) Flat BC policy: To verify the necessity of hierarchical structures, we train a flat end-to-end BC policy on the long-horizon trajectories during the high-level policy data collection phase. We adopt the same policy neural network architecture as in Fig. 2, with an additional action dimension to control the gripper closure. We found it achieved 0% success rate out of 24 trials; and it never succeeded even in routing the first clip, which corresponded to failure modes such as not picking up the cable, directly moving the arm out of the board, missing the clip, prematurely dropping the cable before it's routed, and combinatorial of such. This indeed suggests the proposed task is a challenging one that poses difficulties in a compounded way that necessitates reasonable hierarchical approaches dealing with those difficulties modularly.
- 3) Behavior Transformer(BeT): One possible reason that flat BC policy performs poorly is the multimodal nature of human demonstrations. To address this point so that we can make concrete a conclusion in terms of hierarchical structure. We compare to a BeT policy [70], a promising approach specifically geared towards handling multi-modal inputs, trained on the same dataset we train the flat BC policy as mentioned in the last paragraph. We refer to Appx. C for implementation details of BeT. It is observed that BeT policy also ended up with 0% success rate as well as exhibiting similar failure modes as the flat BC policy, including not grasping the cable and missing the clip. This suggested although human demonstrations may be naturally multi-modal, the bottleneck of the proposed problem is largely orthogonal to that.
- 4) Action Chunking(ACT): Results from the aforementioned flat BC policies motivate the use of hierarchical approaches to address error-compounding issues in this long-horizon task. One natural intermediate step between a fully flat method to a fully hierarchical method such as ours with complex machinery is the semi-hierarchical approach. One instance of such an approach is Action Chunking(ACT) [96], which employs multiple-step action prediction and ensembles to alleviate issues of error accumulation. We detail the algorithm implementation in Appx. C. While we did find the executed action smoothness improved, the trained ACT policy was not able to succeed at the task at all. The failure modes are missing the clip and dropping the cable to pick up the next section when the current clip is not routed.

We can draw several conclusions from these results: (1) The poor performance of the flat BC and BeT methods is consistent with our hypothesis that hierarchically organized policies are important for recovering from compounding errors over the stages of the task. (2) Methods like ACT, which resemble a kind of implicit hierarchical approach by predicting temporally extended action sequences, still significantly underperform our explicit hierarchical policies. (3) However, designing the high-level state machine manually is also insufficient, and the recovery strategies acquired automatically by our learned higher-level policies lead to more than double the success rate of the hand-designed state machine.

Model	Success Rate
Hierarchical Imitation, no history	0 / 24
Hierarchical Imitation (ours)	12 / 24

TABLE V: History ablation for the high-level policy, showing that including the history of previously triggered primitives is essential for good performance.

Ablation experiments with the high-level policy. Table V compares the experimental results for our full method with a memoryless variant, demonstrating the importance of using history information. Specifically, withholding the primitive history embedding causes the policy to fail when switching between clips. After one clip is successfully routed, the memoryless policy continuously recognizes the completion of the routing through the first clip and repeatedly prompts the go\_next primitive, but fails to move on and begin routing the next clip (i.e., prompting a pickup). In practice, we found maintaining a reasonable length of context history helps the high-level policy make better decisions; for example, more recovering behavior will be attempted.

Qualitative analysis of learned behavior. Our high-level policy makes decisions by processing its sensory inputs. By training on a diverse dataset end-to-end, our policy in theory should also generate new emergent behavior that was not seen in the training dataset. Indeed, we observed a few interesting behaviors at test time. The robot expanded the usage of the Perturb\_cable primitive combining with other primitives to create novel recovery mechanisms. For example, as shown in Fig. 13, the robot applies Perturb\_cable. However, the cable resulted in the inner corner of the clip where it couldn't be picked up; then the robot applied again Perturb\_cable to adjust the shape of the rope to a level it could pick up the cable.

**Failure mode analysis.** We observe that over 95% of the failures in the system occur when the high-level policy would predict the go\_next primitive while the cable is not actually successfully routed, especially if the cable is just behind the clip, resulting in an irrecoverable state. We hypothesize this is because the visual feature of a clip being inside versus behind the clip is not very distinctive, and perhaps a different camera placement would alleviate the issue.

#### D. Interactive Fine-Tuning to Quickly Improve Performance

We additionally demonstrate the ability of our model can fine-tune its performance in both out-of-distribution(OOD) and in-distribution scenarios with a small amount of interactive training, as described in Section IV-C.



(a) Before Perturb (b) Perturbed Once

Once (c) Perturbed Twice

Fig. 13: This figure shows the Perturb\_cable primitive used twice in a row. 13a: The initial state is hard for the Route primitive to succeed. 13b: The red pickup point on the cable is next to the left clip and cannot be picked up. 13c: The cable is perturbed again so it can be picked up.



Fig. 14: Out-of-distribution clip configurations. **Top left**, **Top right**, **Bottom left**: OOD clip configurations in the 3-clip routing task, the drastically different clip orientation makes them particularly challenging due to the resulting slackness of the cable. **Bottom right**: 4-clip routing task with the additional clip not seen in the offline dataset.

Out-of-distribution finetuning. We first evaluate our system on the configurations presented in Figure 14. Three configurations (OOD 1, OOD 2, OOD 3) were specifically selected so that some clips were outside the 12.5cm by 15cm box used for selecting the corresponding clip configurations in the training set, or outside the 0 to 45 degree range of orientations used in training, which resulted in scenarios that present a particularly significant generalization challenge. The fourth configuration was selected to include an additional fourth clip that did not appear in any of the original training data for the high-level policy. For each of the four out-of-distribution clip configurations, we collected 10 interactive demonstrations which were used to fine-tune the policy. Figure 15 shows the overall generalization performance of the fined-tuned policies compared with the direct zero-shot transfer of our original policy while Table VI details the number of successes for each configuration. In the case of three clips, we find our policy was able to quickly improve its performance twofold with this handful amount of fine-tuning data. In the four-clip case, due to view shifts observed by the side camera, the original policy was not able to succeed at all. However, the fine-tuned policy was able to rapidly adapt to this new challenging OOD situation with only ten demonstrations.

In distribution finetuning. It is often of concern in industrial robotic applications that a task should be executed with a fairly high success rate to imply the potential deployment of such systems. Towards that end, we further study how our system can improve reliability for a particular case when the initial success rate is not sufficiently high. To study this setting, we finetuned our pre-trained model on four new randomly



Fig. 15: Evaluation of fine-tuning our high-level policy. Finetuning enables out-of-distribution generalization to clip configurations in Figure 14, including to a configuration where an additional fourth clip was added. This demonstrates the modularity of our system and the ability to extend our method to new clip configurations.

Configuration	Zero-shot (10 Trials)	<b>Finetuned</b> (10 Trials)
Three Clips In Distribution		
Configuration 1	5	9
Configuration 2	6	9
Configuration 3	4	8
Configuration 4	5	8
Subtotal	20/40	34/40
Three Clips Out of Distribution		
Configuration 1	1	3
Configuration 2	2	6
Configuration 3	3	6
Subtotal	6/30	15/30
Four Clips Out of Distribution		
Configuration 1	0	2
Subtotal	0/10	2/10
Total	26/80	51/80

TABLE VI: Detailed table comparing success rates of zeroshot and fine-tuned high-level policies on 4 in-distribution configurations with three clips, 3 out-of-distribution configurations with three clips, and 1 out-of-distribution configuration with four clips.

sampled clip configurations within its training distributions (separate from the configuration in Section VI.C). Presented in Table VI, our system was able to improve its performance from 50% to 85% with only ten additional demonstrations each measured over 10 trials per each of the four configurations for a total of 40 trials. This suggests our system not only enjoys broad generalization capability but also rapidly improves its reliability when a high success rate is desirable.

### VII. DISCUSSION AND FUTURE WORK

We presented a hierarchical imitation learning system applied to the task of cable routing. Our approach is based on the principle that temporally extended multi-stage tasks become more practical when each stage of the pipeline can compensate for and correct mistakes when they arise. In this way, performing a task that requires multiple stages (e.g., inserting the cable into a series of clips) does not lead to a success rate that drops exponentially with each step. The high-level policy in our system, which selects primitives at each stage, can trigger primitives to retry or correct mistakes, and the learned low-level clip insertion primitive can also correct small mistakes because of the corrections present in the demonstrations. In our experimental evaluation, we show that this approach enables the robot to route cables through a series of clips, and even for harder clip arrangements where the system does not succeed consistently, it can be fine-tuned with as few as 10 additional trials.

Our method still has a number of limitations. Although the success rate of our approach significantly exceeds that of baselines that do not employ learned policies at both levels of the hierarchy, the absolute success rate is still not perfect for industrially relevant applications. Of course, as with all learning systems, larger datasets are likely to lead to improvements in performance. However, it would also be interesting in future work to explore how the addition of more diverse primitives can further enhance the capability of the higher level to correct for mistakes and further reduce failure rates, or how autonomous improvement with RL can improve the method further.

#### APPENDIX

#### A. View-invariant coordinate system

We use a coordinate system attached to the robot's endeffector frame to express observation and actions. This way, the policy will not over-fit to any particular absolute positions; rather generalize to new clip placements if we can keep the same spatial relativity between clips and end-effector. This is also a convenient mechanism for the system to gain robustness by randomizing the end-effector's initial pose; from the policy's perspective, this is equivalent to *physically* moving the goal (clip) but requires no additional mechanical apparatus beyond the robot itself.

Firstly, we use the wrist camera views to train the policies, which are mounted in the end-effector so they directly enjoy the benefit of the mentioned view-invariant coordinate. We also only use the z-position, or the height, of the end-effector in the policy observation space, which is rotational invariant for our 4DoF action space (translation about the XYZ-axis and rotation about the Z-axis).

The policy outputs the 4D twist  $\mathcal{V}_t$  (all translation components, rotation around the z-axis) w.r.t. current end-effector frame  $T_t$ . For controlling the Franka robot with its control software, we convert it back to the robot's base frame as  $\mathcal{V}'_t$ using Adjoint mapping, which is a function of  $T_t$ . Note  $T_t$  is a homogeneous transformation matrix as:

$$T_t = \begin{bmatrix} R_t & p_t \\ 0_{1 \times 3} & 1 \end{bmatrix}$$

Then the Adjoint map to relate the two twists in the current

frame and base frame can be defined as:

$$\mathrm{Ad}_t = \begin{bmatrix} R_t & 0\\ [p_t] \cdot R_t & R_t \end{bmatrix},$$

where  $[p_t]$  is the skew-symmetric matrix constructed from  $p_t$ ; then we calculate  $\mathcal{V}'_t = [\mathrm{Ad}_t]\mathcal{V}_t$  which to be sent to the Franka controller.

#### B. Neural Network Details

For all neural networks, we use ResNet18 as backbones [26]. For low-level routing policy, we first linear project the z-value to a 128-dimensional vector, then concatenate with ResNet embeddings; then go through a 2-layer MLP with 256 nodes each, and finally output the mean and variance of a TanhGaussian policy. For the high-level primitive selection policy, we take the pre-trained ResNet18 from training the lowlevel policy and keep their parameters frozen while training high-level policy. We pass through three view images through the ResNets to get three embedding vectors, we then up-project the z-value to a 128-dimensional vector. For the primitive sequence embedding, we use a learned word embedding layer [61] to project it to an embedding matrix of size  $6 \times 4$ , we then flatten this matrix and up-project it to a 128-dimensional vector; we then concatenate all the vectors so far to pass through a single-layer MLP of size 256 with softmax activation in the end.

## C. Training Hyperparameters

Hyperparameters	Routing Policy
Optimizer	Adam
Base learning rate	3e-4
Weight decay	0.05
Optimizer momentum	$\beta_1 = 0.9,  \beta_2 = 0.99$
Batch size	512
Learning rate schedule	cosine decay

TABLE VII: Hyperparameters for training low-level routing policies

Hyperparameters	Primitive Selection Policy
Optimizer	Adam
Base learning rate	3e-4
Weight decay	0.05
Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.99$
Batch size	128
Learning rate schedule	cosine decay

TABLE VIII: Hyperparameters for training high-level primitive selection policies

Hyperparameters	Behavior Transformer
Optimizer	AdamW
Base learning rate	1e-5
Weight decay	2e-4
Optimizer momentum	$\beta_1 = 0.9,  \beta_2 = 0.99$
Batch size	16
Number of bins k	64
Attention Heads	8
Block Size	144
Decoder Layers	6
Output Embedding Size	256
Resnet Embedding Size	512

TABLE IX: Hyperparameters for training Behavior Transformer

Hyperparameters	Action Chunking
Optimizer	AdamW
Base learning rate	1e-3
Weight decay	3e-3
Batch size	256
Chunk Size	5
Optimizer momentum	$\beta_1 = 0.9, \beta_2 = 0.99$
Exponential moving average weight	0.01
Learning rate schedule	cosine decay

TABLE X: Hyperparameters for training with Action Chunking

#### ACKNOWLEDGMENTS

This work was partially supported by ONR N00014-20-1-2383, NSF IIS-2150826, AFOSR FA9550-22-1-0273, and Intrinsic Innovation LLC. We also thank the computing resources provided by the Berkeley Research Computing (BRC) program.

#### REFERENCES

- Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In 6th Annual Conference on Robot Learning, 2022.
- [2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Jayant Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego M Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding language in robotic affordances. *ArXiv*, abs/2204.01691, 2022.
- [3] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The optioncritic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.
- [4] A. Bagaria and G. Konidaris. Option discovery using deep skill chaining. In *International Conference on Learning Representations (ICLR)*, 2020.
- [5] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic* systems, 13(1-2):41–77, 2003.
- [6] Matthew Michael Botvinick. Hierarchical reinforcement learning and decision making. *Current opinion in neurobiology*, 22 (6):956–962, 2012.

- [7] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. arXiv preprint arXiv:2212.06817, 2022.
- [8] Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. arXiv preprint arXiv:2203.00663, 2022.
- [9] Michele Colledanchise and Petter Ögren. *Behavior trees in robotics and AI: An introduction.* CRC Press, 2018.
- [10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.
- [11] Aidan Curtis, Xiaolin Fang, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Caelan Reed Garrett. Long-horizon manipulation of unknown objects via task and motion planning with estimated affordances. In 2022 International Conference on Robotics and Automation (ICRA), pages 1940–1946, 2022. doi: 10.1109/ICRA46639.2022.9812057.
- [12] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Roberto Martín-Martín, Animesh Garg, Silvio Savarese, and Ken Goldberg. Mechanical search: Multi-step retrieval of a target object occluded by clutter. In 2019 International Conference on Robotics and Automation (ICRA), pages 1614–1621. IEEE, 2019.
- [13] P. Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In Advances in Neural Information Processing Systems, 1992.
- [14] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. Advances in neural information processing systems, 5, 1992.
- [15] Lavindra de Silva, Amit Kumar Pandey, Mamoun Gharbi, and Rachid Alami. Towards combining htn planning and geometric task planning. arXiv:1307.1482, 2013.
- [16] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.
- [17] Danny Driess, Jung-Su Ha, Russ Tedrake, and Marc Toussaint. Learning geometric reasoning and control for long-horizon tasks from visual input. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 14298–14305. IEEE, 2021.
- [18] Yilun Du, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Learning object-based state estimators for household robots. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 12558–12565, 2022. doi: 10.1109/IROS47612.2022.9981287.
- [19] Roy Fox, Richard Shin, William Paul, Yitian Zou, Dawn Song, Ken Goldberg, Pieter Abbeel, and Ion Stoica. Hierarchical variational imitation learning of control programs. arXiv preprint arXiv:1912.12612, 2019.
- [20] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.
- [21] Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings* of the International Conference on Automated Planning and Scheduling, volume 30, pages 440–448, 2020.
- [22] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1): 265–293, 2021.
- [23] Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeffrey Ichnowski, Ashwin Balakrishna, Minho Hwang, Vainavi

Viswanath, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. Untangling dense knots by learning task-relevant keypoints. 2020.

- [24] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In 2017 IEEE international conference on robotics and automation (ICRA), pages 3389–3396. IEEE, 2017.
- [25] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving longhorizon tasks via imitation and reinforcement learning. arXiv preprint arXiv:1910.11956, 2019.
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings* of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.
- [27] Shinichi Hirai, Tatsuhiko Tsuboi, and Takahiro Wada. Robust grasping manipulation of deformable objects. In Proceedings of the 2001 IEEE International Symposium on Assembly and Task Planning (ISATP2001). Assembly and Disassembly in the Twenty-first Century.(Cat. No. 01TH8560), pages 411–416. IEEE, 2001.
- [28] John E Hopcroft, Joseph K Kearney, and Dean B Krafft. A case study of flexible object manipulation. *The International Journal of Robotics Research*, 10(1):41–50, 1991.
- [29] Kyle Hsu, Moo Jin Kim, Rafael Rafailov, Jiajun Wu, and Chelsea Finn. Vision-based manipulators need to also see from their hands. In *International Conference on Learning Representations*, 2022.
- [30] Zixuan Huang, Xingyu Lin, and David Held. Self-supervised cloth reconstruction via action-conditioned cloth tracking. In *IEEE International Conference on Robotics and Automation* (ICRA), 2023, 2023.
- [31] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR), 50(2):1–35, 2017.
- [32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [33] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi (Jim) Fan. Vima: General robot manipulation with multimodal prompts. *ArXiv*, abs/2210.03094, 2022.
- [34] Shiyu Jin, Wenzhao Lian, Changhao Wang, Masayoshi Tomizuka, and Stefan Schaal. Robotic cable routing with spatial representation. *IEEE Robotics and Automation Letters*, 7(2): 5687–5694, 2022. doi: 10.1109/LRA.2022.3158377.
- [35] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical planning in the now. In Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.
- [36] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.
- [37] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. arXiv preprint arXiv:1806.10293, 2018.
- [38] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. arXiv preprint arXiv:2104.08212, 2021.
- [39] Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Hg-dagger: Interactive imitation learning with human experts. In 2019 International Conference on Robotics and Automation (ICRA), pages 8077–8083. IEEE, 2019.

- [40] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
- [41] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Alvaro Sanchez-Gonzalez, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compile: Compositional imitation learning and execution. In *International Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.
- [42] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31 (3):360–375, 2012.
- [43] Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on robot learning*, pages 418–437. PMLR, 2017.
- [44] Hoang Le, Nan Jiang, Alekh Agarwal, Miroslav Dudík, Yisong Yue, and Hal Daumé III. Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pages 2917–2926. PMLR, 2018.
- [45] Alex X Lee, Henry Lu, Abhishek Gupta, Sergey Levine, and Pieter Abbeel. Learning force-based manipulation of deformable objects from multiple demonstrations. In *Robotics* and Automation (ICRA), 2015 IEEE International Conference on, pages 177–184. IEEE, 2015.
- [46] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.
- [47] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal* of Machine Learning Research, 17(1):1334–1373, 2016.
- [48] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421– 436, 2018.
- [49] Hao Li, Yizhi Zhang, Junzhe Zhu, Shaoxiong Wang, Michelle A Lee, Huazhe Xu, Edward Adelson, Li Fei-Fei, Ruohan Gao, and Jiajun Wu. See, hear, and feel: Smart sensory fusion for robotic manipulation. arXiv preprint arXiv:2212.03858, 2022.
- [50] Yunzhu Li, Antonio Torralba, Anima Anandkumar, Dieter Fox, and Animesh Garg. Causal discovery in physical systems from videos. Advances in Neural Information Processing Systems, 33, 2020.
- [51] Antonio Loquercio, Ashish Kumar, and Jitendra Malik. Learning visual locomotion with cross-modal supervision. *arXiv preprint arXiv:2211.03785*, 2022.
- [52] Yuchen Lu, Yikang Shen, Siyuan Zhou, Aaron Courville, Joshua B Tenenbaum, and Chuang Gan. Learning task decomposition with ordered memory policy network. 2021.
- [53] Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, and Alice M. Agogino. Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2062–2069, 2018. doi: 10.1109/ IROS.2018.8594353.
- [54] Jianlan Luo, Eugen Solowjow, Chengtao Wen, Juan Aparicio Ojea, Alice M Agogino, Aviv Tamar, and Pieter Abbeel. Reinforcement learning on variable impedance controller for highprecision robotic assembly. In 2019 International Conference on Robotics and Automation (ICRA), pages 3080–3087. IEEE, 2019.
- [55] Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Wenzhao Lian, Chang Su, Mel Vecerik, Ning Ye, Stefan Schaal, and Jonathan Scholz. Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-

Scale Study. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.088.

- [56] Shan Luo, Wenzhen Yuan, Edward Adelson, Anthony G Cohn, and Raul Fuentes. Cloth texture recognition using vision and tactile sensing. In *ICRA 2018 workshop: Active touch for perception and interaction: how nature inspires robotics*, 2018.
- [57] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.
- [58] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In 2010 IEEE International Conference on Robotics and Automation, pages 2308–2315. IEEE, 2010.
- [59] Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Petter Ögren. Towards a unified behavior trees framework for robot control. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 5420–5427. IEEE, 2014.
- [60] Toki Migimatsu and Jeannette Bohg. Grounding predicates through actions. *IEEE International Conference on Robotics* and Automation (ICRA), 2022.
- [61] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [62] Takuma Morita, Jun Takamatsu, Koichi Ogawara, Hiroshi Kimura, and Katsushi Ikeuchi. Knot planning from observation. In 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), volume 3, pages 3887–3892. IEEE, 2003.
- [63] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining selfsupervised learning and imitation for vision-based rope manipulation. In 2017 IEEE international conference on robotics and automation (ICRA), pages 2146–2153. IEEE, 2017.
- [64] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. ACM Computing Surveys (CSUR), 54(5):1–35, 2021.
- [65] Chris Paxton, Nathan D. Ratliff, Clemens Eppner, and Dieter Fox. Representing robot task plans as robust logical-dynamical systems. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5588–5595, 2019.
- [66] Jan Peters and Stefan Schaal. Learning to control in operational space. *The International Journal of Robotics Research*, 27(2): 197–212, 2008. doi: 10.1177/0278364907087548.
- [67] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In 2016 IEEE international conference on robotics and automation (ICRA), pages 3406–3413. IEEE, 2016.
- [68] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627– 635, 2011.
- [69] John Schulman, Alex X. Lee, Jonathan Ho, and P. Abbeel. Tracking deformable objects with point clouds. 2013 IEEE International Conference on Robotics and Automation, pages 1130–1137, 2013.
- [70] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning \$k\$ modes with one stone. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022.
- [71] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and

- [72] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation, 2023.
- [73] Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2020.
- [74] Yu She, Shaoxiong Wang, Siyuan Dong, Neha Sunil, Alberto Rodriguez, and Edward Adelson. Cable manipulation with a tactile-reactive gripper. *The International Journal of Robotics Research*, 40(12-14):1385–1401, 2021.
- [75] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. J. Big Data, 6:60, 2019. doi: 10.1186/s40537-019-0197-0.
- [76] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiveractor: A multi-task transformer for robotic manipulation. *ArXiv*, abs/2209.05451, 2022.
- [77] Dong Sun, Xiaolun Shi, and Yunhui Liu. Modeling and cooperation of two-arm robotic system manipulating a deformable object. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2346–2351. IEEE, 1996.
- [78] Richard S Sutton and Andrew G Barto. *Reinforcement learning:* An introduction. MIT press, 2018.
- [79] Finn Süberkrüb, Rita Laezza, and Yiannis Karayiannidis. Feel the tension: Manipulation of deformable linear objects in environments with fixtures using force information. In 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 11216–11222, 2022. doi: 10.1109/ IROS47612.2022.9982065.
- [80] Vainavi Viswanath, Kaushik Shivakumar, Justin Kerr, Brijen Thananjeyan, Ellen Novoseller, Jeffrey Ichnowski, Alejandro Escontrela, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. Autonomously untangling long cables.
- [81] Vainavi Viswanath, Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Ellen Novoseller, Jeffrey Ichnowski, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. Disentangling dense multi-cable knots. In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3731–3738. IEEE, 2021.
- [82] Takahiro Wada, Shinichi Hirai, Sadao Kawamura, and Norimasa Kamiji. Robust manipulation of deformable objects by a simple pid feedback. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 1, pages 85–90. IEEE, 2001.
- [83] Gabriel Arslan Waltersson, Rita Laezza, and Yiannis Karayiannidis. Planning and control for cable-routing with dual-arm robot. In 2022 International Conference on Robotics and Automation (ICRA), pages 1046–1052. IEEE, 2022.
- [84] Chen Wang, Danfei Xu, and Li Fei-Fei. Generalizable task planning through representation pretraining. *IEEE Robotics and Automation Letters*, 7:8299–8306, 2022.
- [85] Fei Wang, Etienne Burdet, Ronald Vuillemin, and Hannes Bleuler. Knot-tying with visual and force feedback for vr laparoscopic training. In 2005 IEEE engineering in medicine and biology 27th annual conference, pages 5778–5781. IEEE, 2006.
- [86] Thomas Weng, Sujay Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, 2021.
- [87] Achu Wilson, Helen Jiang, Wenzhao Lian, and Wenzhen Yuan. Cable routing and assembly using tactile-driven motion primitives. In 2023 IEEE International Conference on Robotics and Automation (ICRA), pages 10408–10414, 2023. doi: 10.1109/ICRA48891.2023.10161069.

- [88] Yuxin Wu and Kaiming He. Group normalization. In Proceedings of the European conference on computer vision (ECCV), pages 3–19, 2018.
- [89] Markus Wulfmeier, Dushyant Rao, Roland Hafner, Thomas Lampe, Abbas Abdolmaleki, Tim Hertweck, Michael Neunert, Dhruva Tirumala, Noah Siegel, Nicolas Heess, et al. Dataefficient hindsight off-policy option learning. In *International Conference on Machine Learning*, pages 11340–11350. PMLR, 2021.
- [90] Fan Xie, Alexander Chowdhury, M De Paolis Kaluza, Linfeng Zhao, Lawson Wong, and Rose Yu. Deep imitation learning for bimanual robotic manipulation. *Advances in neural information* processing systems, 33:2327–2337, 2020.
- [91] Zhenjia Xu, Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Dextairity: Deformable manipulation can be a breeze. arXiv preprint arXiv:2203.01197, 2022.
- [92] Mengyuan Yan, Yilin Zhu, Ning Jin, and Jeannette Bohg. Self-supervised learning of state estimation for manipulating deformable linear objects. *IEEE robotics and automation letters*, 5(2):2372–2379, 2020.
- [93] Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54):eabd8803, 2021.
- [94] Harry Zhang, Jeffrey Ichnowski, Daniel Seita, Jonathan Wang, Huang Huang, and Ken Goldberg. Robots of the lost arc: Selfsupervised learning to dynamically manipulate fixed-endpoint cables. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 4560–4567. IEEE, 2021.
- [95] Tony Z. Zhao, Jianlan Luo, Oleg Sushkov, Rugile Pevceviciute, Nicolas Heess, Jon Scholz, Stefan Schaal, and Sergey Levine. Offline meta-reinforcement learning for industrial insertion. In 2022 International Conference on Robotics and Automation (ICRA), pages 6386–6393, 2022. doi: 10.1109/ICRA46639. 2022.9812312.
- [96] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with lowcost hardware, 2023.
- [97] Jihong Zhu, Benjamin Navarro, Robin Passama, Philippe Fraisse, André Crosnier, and Andrea Cherubini. Robotic manipulation planning for shaping deformable linear objects withenvironmental contacts. *IEEE Robotics and Automation Letters*, 5(1):16–23, 2019.
- [98] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In 2017 IEEE international conference on robotics and automation (ICRA), pages 3357–3364. IEEE, 2017.