Stabilize to Act: Learning to Coordinate for Bimanual Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: The ability to coordinate meaningfully between two hands unlocks a variety of skills to humans. Bimanual robotics opens the door to accomplishing a similarly large set of manipulation tasks. However, constructing control policies for dual arm autonomous systems brings inherent difficulties—the high-dimensionality of the bimanual action space adds complexity to both model-based and data-driven methods. We counteract this challenge by drawing inspiration from humans to propose a novel role assignment framework: a *stabilizing* arm holds an object in place to simplify the environment while an *acting* arm executes the task. We instantiate this framework with BimanUal Dexterity from Stabilization (BUDS), which uses a learned *restabilizing classifier* to alternate between updating a learned stabilization position to keep the environment unchanged, and accomplishing the task with an acting policy learned from demonstrations. We evaluate BUDS on four dexterous bimanual tasks on real-world robots and present experiments highlighting overall task success and policy generalizability across objects within a class. Given 20 demonstrations, BUDS achieves 76.9% success on completing a wide variety of physical bimanual tasks, and achieves 52.7% success on when generalizing to out-of-distribution objects. BUDS is 56.0% more successful than a unstructured baseline that instead learns a BC stabilizing policy due to the precision required of these complex tasks. Supplementary material and videos can be found at https://tinyurl.com/stabilizetoact.

Keywords: Bimanual Manipulation, Learning from Demonstrations, Deformable Object Manipulation

1 Introduction

Numerous household, surgical, and industrial tasks require reasoning about bimanual coordination, such as cutting food, suturing a wound, or plugging sockets to connect two cables. Often, humans choose to alternate focus between holding a point stationary and executing a precise motion to accomplish the task. In robotics, the addition of a second arm opens the door to a higher level of task complexity, but comes with a number of control challenges. With a second arm, we have to reason about how to produce coordinated behavior in a higher dimensional action space, resulting in more computationally challenging learning, planning, and optimization problems. The addition of a second arm also complicates data collection—it requires teleoperating a robot with more degrees of freedom—which hinders our ability to rely on methods that require expert bimanual demonstrations.

Previous model-based techniques have proposed planning algorithms for bimanual tasks such as collaborative transport or scooping [1, 2, 3], but require hand-designed specialized primitives or follow predefined trajectories limiting their abilities to learn new skills or adapt. On another extreme, we turn to reinforcement learning (RL) techniques that do not need costly primitives. However, RL methods are notoriously data hungry and a high-dimensional bimanual action space further exacerbates this problem. While simulation-to-real transfer techniques offer an appealing alternative [4, 5, 6, 7], the domain gap between simulation and reality is often significantly larger for

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

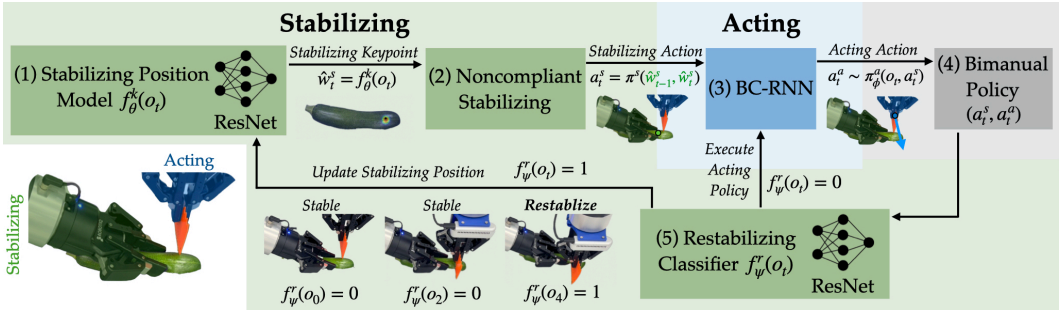

Figure 1: **BUDS: BimanUal Dexterity from Stabilization**: We present BUDS, an algorithm for bimanual manipulation that uses a novel stabilizing and acting role assignment framework to efficiently learn to coordinate a dual-arm robot. For the stabilizing role, BUDS learns a Stabilizing Position Model (1) to determine a point to hold stationary using a noncompliant controller (2). In this simplified environment, BUDS learns to act from single-arm demonstrations (3). Combined, these two actions comprise a bimanual policy (4). Finally, at every timestep, BUDS classifies whether the stabilizing position is effective or needs to be updated using a Restabilizing Classifier (5).

bimanual tasks and a large number of tasks that benefit from dual arms also require closed-chain contacts or high-force interactions which can lead to unstable simulation dynamics [8, 9, 10]. A more promising direction is learning from demonstration, which balances between model-based hand-crafted primitives and sample-inefficient exploration of reinforcement learning. However, bimanual demonstrations can be equally as time and labor intensive to collect—data collection for simultaneously controlling two high-degree freedom arms often requires specialized hardware or multiple operators [11, 12, 8, 13, 14]. The increased dimensionality of the action space also necessitates significantly more demonstrations, especially for more precise or dexterous tasks [15].

Given the high-dimensionality of the bimanual setting, we draw inspiration from humans' coordination for these tasks. We notice that when humans perform bimanual tasks, they often use two roles: one arm stabilizing and one arm acting. In tasks such as plugging in a phone or cutting a steak, a stabilizing arm holds an object (e.g. the phone or steak) stationary to simplify the environment and make it easier for the acting arm to complete the task with high precision. In more complex tasks, the stabilizing policy should adjust throughout the task to keep an object stationary. For example, a fork must skewer a steak at different points depending on where the knife cuts.

Our key insight is that for a variety of bimanual tasks, each arm follows one of two roles: a ***stabilizing*** *arm stabilizes an object to simplify the environment for an* ***acting*** *arm to perform the task.*

We propose a method for bimanual manipulation by leveraging this coordination paradigm to decompose our bimanual problem into two single agent problems: learning to stabilize and learning to act. The stabilizing policy decides *where* to stabilize in the scene and *when* to adjust, while the acting arm learns to perform the task in this simpler *stabilized* environment. For example when cutting a steak, our *stabilizing policy* learns where to hold a steak and when to adjust its hold so the steak will remain stationary while an *acting policy* makes the cut.

To learn where to stabilize, we use a vision-based system that takes an environment image as input and outputs a stabilization keypoint position. We then learn a restabilizing classifier that determines from images when the stabilizing keypoint is no longer effective and needs to be updated. We deploy this stabilizing policy while collecting *single-arm* trajectory demonstrations for an acting policy to sidestep the need for a precise and expensive bimanual demonstration collection interface. Using these demonstrations, the acting arm learns a policy via imitation learning to accomplish the task in this simplified, stationary environment. We demonstrate the efficacy of this paradigm on four diverse and dexterous tasks on a physical bimanual robot. BUDS achieves 76.9% success, and outperforms an unstructured baseline fully learned from expert trajectory demonstrations by 56.0%. Additionally, BUDS achieves 52.7% when generalizing to unseen objects of similar morphology (e.g. transferring a cutting policy trained on jalapeños to cutting zucchini and celery).

Our contributions are: (1) A paradigm for learning bimanual policies with role assignments, where the stabilizing arm stabilizes the environment and reduces the non-stationarity present while an acting arm learns to perform a task allowing in a simpler learning setting, (2) A framework for

collecting bimanual demonstrations that bypasses the need for a dual-arm interface by collecting demonstrations for the stabilizing and acting roles independently, and (3) A system, BUDS, that instantiates this paradigm to learn a centralized bimanual policy to perform a broad range of tasks.

2 Related Work

In this section, we describe the new challenges introduced by the addition of a second arm, along with the current data-driven and model-based methods available for bimanual manipulation tasks.

Learning-based Bimanual Manipulation. A recurring challenge throughout bimanual manipulation is the high-dimensionality of the action space. This appears both in reinforcement learning (RL) and imitation learning (IL) works [16, 11, 17, 18, 19, 20, 21]. Prior multi-agent coordination works have considered shrinking the high-dimensionality of the problem by using a second agent stabilizing a latent intent [22], however learning a stabilizing policy and a latent intent mapping both require a significant amount of data that is not realistic for physical robot manipulation tasks.

RL methods for learning high-frequency bimanual control policies can require a large number of samples and many hours of robot time, which makes simulation to real policy transfer an appealing approach [21, 7, 6]. However, sim-to-real approaches are limited to settings where the sim-to-real gap is small, which precludes many contact-rich bimanual tasks such as zipping a zipper or cutting food [8, 9, 10]. Instead, works in both RL and IL settings have proposed using parameterized movement primitives to reduce the action space, and have achieved reasonable success on tasks such as opening a bottle and lifting a ball [17, 21, 18, 19, 20, 23, 24, 25, 26, 27, 28, 29, 15, 30]. However, these movement primitives greatly limit the tasks achievable by the method as they often require costly demonstrations or labor-intensive hard-coded motions for each task-specific primitive. Additionally, learning from demonstrations in bimanual settings is difficult as teleoperating two high-degree-freedom robots or collecting kinesthetic demonstrations on both arms simultaneously is challenging and sometimes impossible for a single human and may require specialized hardware [16, 11, 31, 12, 8, 13]. Recent works have demonstrated more effective interfaces for data collection in a bimanual setting, but these interfaces are limited to specific hardware instantiations and would still require large amounts of expert data to learn a high dimensional policy [14]. To avoid the need for expert bimanual demonstrations, we use a novel stabilizing paradigm to decouple the arms' policies and learn a role-specific policy for each arm from single-arm demonstrations. This added structure also brings down the dimensionality of the large action space in a task-agnostic way.

Model-based Bimanual Manipulation. The majority of model-based bimanual manipulation methods are limited to using task planning and constraint solving methods to jointly move or hold a large object [32, 33, 34, 35, 12, 2, 1, 36]. Bersch et al. [37] and Grannen et al. [3] present systems using a sequence of hard coded actions for folding a t-shirt and scooping food items respectively. However, as tasks become more complex and dexterous, the primitives required also become more unintuitive and costly to hand-design. We instead learn a high-frequency control policy from single-arm demonstrations, avoiding the need for labor-intensive hand-coded primitives while being able to perform dexterous bimanual manipuation tasks.

3 Stabilizing for Bimanual Tasks

Given a set of expert demonstrations, we aim to produce a bimanual policy for executing a variety of manipulation tasks, such as capping a marker or zipping up a jacket. We formulate each bimanual task as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{O}, \mathcal{A}, \mathcal{T})$. We assume access to observations at each time step $o_t = (i_t, p_t) \in \mathcal{O}$ including image $i_t \in \mathbb{R}^{H \times W \times 3}$ and proprioceptive state $p_t \in \mathbb{R}^{14}$. $\mathcal{A}$ is the action space that contains 14 degrees of freedom (DoF) joint actions $a_t = (a_t^s, a_t^a)$ for the two robot arms, i.e., $a_t^s, a_t^a \in \mathbb{R}^7$, that we refer to as the *stabilizing* and *acting* arms respectively. We assume unknown transition dynamics $\mathcal{T} : \mathcal{O} \times \mathcal{A} \rightarrow \mathcal{O}$.

Assumptions. We assume access to a bimanual manipulator and workspace that is reachable for both arms, along with a standard (x, y, z) coordinate frame in this workspace. We assume access to

overhead RGB-D workspace images and the corresponding pixel-to-world calibration transformations that convert a pixel location to a $(x, y, z) \in \mathbb{R}^3$ location in the world coordinate frame, which we later refer to as a keypoint. We first assume access to a set of expert bimanual demonstrations $\mathcal{D}$, and later relax this assumption to two sets of expert unimanual demonstrations $\mathcal{D}^a$ and $\mathcal{D}^s$ to avoid the challenges of collecting bimanual demonstrations. Each demonstration—$d \in \mathcal{D}$, $d^a \in \mathcal{D}^a$, or $d^s \in \mathcal{D}^s$—is a sequence of observation, action pairs: $d = [(o_0, a_0^s, a_0^a), (o_1, a_1^s, a_1^a), \dots]$, $d^a = [(o_0, a_0^a), (o_1, a_1^a), \dots]$, or $d^s = [(o_0, a_0^s), (o_1, a_1^s), \dots]$.

3.1 Monolithic 14-DoF Policy

Let us first consider learning a monolithic 14-DoF policy $\pi_\theta(a_t^s, a_t^a | o_t)$ that is parametrized by θ, takes an observation o_t as input, and outputs a bimanual action (a_t^s, a_t^a). We aim to find a policy that matches the expert demonstrations in $\mathcal{D}$. Given a set of bimanual demonstrations $\mathcal{D}$, we learn this policy by minimizing the supervised loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(o, a^s, a^a) \sim \mathcal{D}} \log \pi_\theta(a^s, a^a | o). \tag{1}$$

While this is feasible in theory, in practice this optimization struggles with instability and sample inefficiency during learning due to the difficulty of collecting high-quality bimanual expert demonstrations and the high-dimensionality of the action space. These challenges are further exacerbated when the precision and dexterity of a task increases: for example it is more difficult for an expert to teleoperate a robot to cap a marker in a consistent way than to put a lid on a pot. Overcoming both the high-variability of these demonstrations as well as the high-dimensional action space necessitates more demonstrations, which translates to costly hours of expert time. To mitigate these challenges, we are motivated by the stabilizing structures present in many bimanual tasks, and incorporate a task-agnostic role-assignment structure into learning of bimanual policies.

3.2 Bringing Down the Dimensionality: Role Assignments

We make the observation that a wide variety of human bimanual tasks leverage a similar paradigm: one arm *stabilizes* objects in the scene to simplify the environment while the other arm *acts* to accomplish the task. In a robotics setting, we translate this observation into a generalizable insight for more efficient learning of bimanual policies: assign either a stabilizing or acting role to each arm to specify a coordination framework and shrink the action space during data collection and policy learning. However, once we assign these roles, a question that arises is how the stabilizing arm's policy depends on the acting arm's policy or vice versa?

Our bimanual policy $\langle a_t^s, a_t^a \rangle \sim \pi(\cdot | o_t)$ can be decomposed into two role-specific policies: a stabilizing policy $a_t^s \sim \pi^s(\cdot | o_t, a_t^a)$ and an acting policy $a_t^a \sim \pi^a(\cdot | o_t, a_t^s)$. To avoid the problems with learning the monolithic policy discussed earlier, we would like to disentangle these two policies. We make the insight that there often exists a stabilizing action $\bar{a}_t^s$ that for every acting action a_t^a withstands the forces produced by the acting action at a task-specific location or region $s(o_t)$. In practice, we instantiate $s(o_t)$ as a keypoint on an image in Section 4.1. Under such a stabilizing policy, we can reduce the action space of bimanual tasks, and focus on only learning the acting policy within a smaller distribution space of simplified states. One can attempt to find a stabilizing policy that is not conditioned on the acting arm actions a_t^a, by looking for stabilizing actions that minimize the maximum possible change in some task-relevant representation to be held stationary $c(s(o_{t-1}), s(o_t))$ that the acting arm can create, i.e.,

$$\bar{\pi}^s(s(o_{t-1}), s(o_t)) = \arg\min_{a_t^s} \max_{a_t^a} c(s(o_{t-1}), s(o_t)). \tag{2}$$

Given the stabilizing action $\bar{\pi}^s(s(o_{t-1}), s(o_t)) = \bar{a}_t^s$, we obtain an acting action $a_t^a \sim \pi^a(\cdot | o_t, \bar{a}_t^s)$.

In practice, we instantiate the task-relevant representation to be stabilized $s(o_t)$ as a keypoint model learned from expert demonstrations (using a learned mapping from an image to a keypoint $f^k : i \mapsto w^s$). We do not solve Eq. (2) but instead utilize a noncompliant controller to hold this point stationary over time (see Section 4.1). Furthermore, we assume access to this stabilizing policy that outputs a constant stabilizing action for a set of future acting actions $a_{[t,t+n]}^a$, where n is of

variable length. We learn a restabilizing classifier $f^r(o_t) = \{0, 1\}$ that determines when n has been surpassed and a new learned stabilizing action should be predicted. We describe this implementation further in Section 4 and show in our experiments in Section 5 that this approximation holds.

4 BUDS: BimanUal Dexterity from Stabilization

We describe our system BimanUal Dexterity from Stabilization (BUDS), which instantiates the stabilizing and acting role assignment described in Section 3. As shown in Fig. 1, we learn a model for each role — f^k_θ for stabilizing and π^a_ϕ for acting — which are each parameterized by weights θ and ϕ respectively. We also learn a restabilizing classifier f^r_ψ where ψ represents the learned model weights. All three models are learned from human-annotated images or single-arm teleoperated robot demonstrations, avoiding the difficulties of collecting bimanual demonstrations.

4.1 Learning a Stabilizing Policy

From Section 3, we aim to find a stabilizing policy $\bar{\pi}^s(s(o_{t-1}), s(o_t)) = \bar{a}^s_t$. More specifically, we aim to learn a task-specific representation s to be stabilized. We make the observation that when humans stabilize in bimanual tasks, they *hold a point in the environment stationary over time*. Thus, $\mathcal{D}^s$ contains two types of actions: stationary or zero-actions that hold a point in place with a noncompliant controller or transient actions that move between stabilizing positions. Additionally, this observation implies s can be instantiated as a mapping from an observation o_t to a stabilization point or position w^s_t. As a result, we decompose the task of stabilizing into two parts: (1) selecting a stabilization position w^s to hold stationary for some period of time and (2) sensing when to update the stabilization position.

Algorithm 1 Stabilizing with BUDS

while Task Incomplete **do**
  $\hat{w}^s_t = f^k_\theta(o_t)$
  while $f^r_\psi(o_t) = 0$ **do**
    $a^s_t = \pi^s(\hat{w}^s_{t-1}, \hat{w}^s_t)$ $\triangleright$ $\{a^s_t : \hat{w}^s_t \simeq \hat{w}^s_{t-1}\}$
    $a^a_t \sim \pi^a_\phi(o_t, a^s_t)$
    Execute a^s_t, a^a_t. Observe $o_{t+1}, f^r_\psi(o_{t+1})$.
  end while
end while

We parameterize w^s as a keypoint on an overhead image of the workspace. We use a ResNet-34 [38] backbone to learn a mapping $f^k_\theta : \mathbb{R}^{640\times480\times3} \to \mathbb{R}^{640\times480}$, which takes as input an overhead image and outputs a Gaussian heatmap centered around the predicted stabilizing keypoint $\hat{w}^s$. This mapping is learned from the stationary actions in the demonstration data $\mathcal{D}^s$, indicating that the arm is at a stabilizing position in this demonstration. In practice, we bypass the need for full trajectory demonstrations and provide supervision in the form of keypoint annotations. Given $\hat{w}^s$ and a depth value from the overhead camera, a non-compliant controller grasps this 3D point and holds it stationary. Thus, we approximate the stabilizing action a^s_t with the action that keeps the keypoint stationary, i.e., $\hat{w}^s_t \approx \hat{w}^s_{t-1}$. We can thus write $\pi^s(s(o_{t-1}), s(o_t))$ as a function of two consecutive learned keypoints $\pi^s(\hat{w}^s_{t-1}, \hat{w}^s_t)$, where the stabilizing points are learned from demonstrations: $\hat{w}^s_t = f^k_\theta(o_t)$.

The learned keypoint mapping f^k_θ is trained with a hand-labelled dataset of 30 image and keypoint pairs, where the keypoint is annotated as the stabilizing keypoint w^s_t for the image. We fit a Gaussian heatmap centered at the annotation with a standard deviation of 8px. This dataset is augmented 10X with a series of label-preserving image transformations [39] (see Appendix A). From this dataset, f^k_θ learns to output the stabilizing keypoint $\hat{w}^s$ for the stabilizing policy to hold stationary.

To determine when to update w^s, we close the feedback loop by learning a restabilizing classifier $f^r_\psi : \mathbb{R}^{640\times480\times3} \to \{0, 1\}$ that maps input workspace images to a binary output indicating whether or not to update w^s. This mapping is learned from the transient actions in the demonstration data $\mathcal{D}^s$—indicating that the stabilizing positions at these states need to be updated. In practice, we forgo using full trajectory demonstrations for supervision in the form of binary expert annotations. We instantiate f^r_ψ with a ResNet-34 [38] backbone and train this classifier with an expert-labelled dataset of 2000 images. For each rollout, an expert assigns when in the rollout a new stabilizing position w^s is needed; the preceding images are labelled 0 while the following images are labelled 1. This dataset is augmented 2X with affine image transformations (See Appendix A for details). f^r_ψ learns to predict a binary classification of when the stabilizing point is no longer effective and needs to be updated with f^k_θ. Together, f^k_θ and f^r_ψ define a stabilizing policy π^s as outlined in Algorithm 1.

4.2 Learning an Acting Policy

Given a stabilization policy $\pi^s(\hat{w}^s_{t-1}, \hat{w}^s_t)$, an acting policy π^a_ϕ learns to accomplish the task in a simpler stationary environment. We instantiate π^a_ϕ with a BC-RNN architecture that is trained on 20 single-arm demonstrations. A expert teleoperates the acting arm using a SpaceMouse [40], a 3D joystick interface. During data collection, the stabilizing arm is assumed to be in the expert-labelled stabilizing position w^s and the environment is in a simplified state. π^a_ϕ optimizes the standard imitation learning loss as defined in Eq. (1), and we refer the reader to Appendix A for more details.

To further increase sample efficiency, we assume that our expert acting demonstrations start from a pre-grasped initial position. To achieve this pre-grasped position, we train an optional grasping keypoint model f^g for the acting policy that maps input workspace images $i_t \in \mathbb{R}^{640\times480\times3}$ to a Gaussian heatmap centered around the grasp point. This grasping model is instantiated with the same ResNet-34 [38] and dataset parameters as used for the stabilizing keypoint model f^k_θ. The acting arm moves to the keypoint position in a fixed orientation, and grasps to begin the task.

5 Experiments

We validate the efficacy of BUDS on four diverse bimanual tasks. We use two UR16e arms each with a Robotiq 2F-85 gripper, mounted at a 45° angle off a vertical mount, 0.3m apart. We use a RTDE-based impedance controller to control the arms at 10Hz on an Intel NUC. We use three Intel Realsense cameras: two 435 cameras mounted at a side view and on the robot wrist, and one 405 camera mounted overhead. For additional details, see Appendix B.

Figure 2: **Task Generalization**: We present the objects used for the Jacket Zip, Marker Cap, and Cut Vegetable tasks, including the *Unseen* objects used to test the generalizability of our system. We classify these OOD objects into two classes, Easy and Hard, based on their visual similarity to the objects seen during training.

Bimanual Tasks. We consider four bimanual tasks, as shown in Fig. 3, and test the generalization of BUDS to unseen objects (Fig. 2). Each task requires both a high-precision acting policy and a dynamic stabilizing policy that restabilizes multiple times during task execution. We emphasize the complexity of the coordination required of these dexterous tasks. Together, these four tasks represent a wide range of real-world bimanual manipulation tasks, which highlights the prevalence of the stabilizing and acting role assignments. For all tasks, we vary the initial position of all objects over each trial. For more details and videos, see Appendix B and our website.

- **Pepper Grinder.** We grind pepper on three plates in order of color—yellow, pink, then blue as shown in Fig. 3. This task requires restabilizing the pepper grinder over each plate in succession.
- **Jacket Zip.** We zip a jacket by pinning down the jacket's bottom and pulling the zipper to the top. Due to the jacket's deformability, the robot must pin the jacket as close as possible to the zipper. We train all models with a red jacket, and the keypoint models on two more jackets: dark grey and blue. We aim to generalize to light grey and black jackets with different material and zippers.
- **Marker Cap.** We cap three markers in sequence from bottom to top of the workspace. This task requires restabilizing after each marker is capped. We train all policies on red, green, and blue Crayola markers and test generalization with Expo and Redimark markers.
- **Cut Vegetable.** We cut a vegetable half (7-9cm) into four 1-4cm pieces with three cuts. This task requires restabilizing the grasp on the vegetable as each cut is made, as the stabilizing arm should hold the vegetable as close as possible to the cut to prevent tearing and twisting. We train on a jalapeño and test generalization with zucchini halves (15-18cm) and celery sticks (8-10cm).

Baselines. We compare BUDS to a **BC-Stabilizer** baseline, which replaces the stabilizing keypoint model f^k_θ with a policy learned from 20 trajectory demonstrations. This stabilizing policy is instantiated with the same BC-RNN architecture and training procedure as used for BUDS's acting policy. We use an oracle classifier to determine when BC-Stabilizer has reached a valid stabilizing

Figure 3: **Experiment Rollouts**: We visualize experiment rollouts of our system. All tasks switch between updating a stabilizing position and executing an acting policy. We visualize the stabilizing position w^s for each time step, along with the acting policy's actions to accomplish the task.

Task	BC-Stabilizer	BUDS	BUDS Failures			
			$\hat{w}^s$	π^a	f^r_ψ	G
Pepper Grinder	0.399 ± 0.21	$\mathbf{1.00} \pm \mathbf{0.00}$	0	0	0	0
Jacket Zip (Clean)	0.282 ± 0.24	$\mathbf{0.721} \pm \mathbf{0.18}$	0	3	3	1
Jacket Zip (Occluded)	0.216 ± 0.17	$\mathbf{0.557} \pm \mathbf{0.37}$	1	2	1	2
Marker Cap	0.000 ± 0.00	$\mathbf{0.901} \pm \mathbf{0.16}$	1	2	0	0
Cut Vegetable	0.150 ± 0.17	$\mathbf{0.668} \pm \mathbf{0.24}$	2	4	0	3

Table 1: **Physical Results:** We report average success and standard deviation across 10 trials of 4 bimanual tasks with randomly initialized object positions. For Jacket Zip, we classify initial configurations as Clean or Occluded based on whether none or up to 30% of the zipper is occluded respectively. We report 4 BUDS failure modes: $\hat{w}^s$ stabilizing keypoint, π^a acting policy, f^r_ψ restabilizing, and *(G)* poor grasps. We compare to a BC-Stabilizer baseline where a single-arm IL policy replaces the stabilizing keypoint model f^k.

position, at which point a noncompliant controller holds the point stationary as in BUDS while the unimanual acting policy from BUDS accomplishes the task. When the learned restabilizing classifier from BUDS f^r_ψ is triggered, the process repeats. This baseline illustrates the importance of using a low-dimensional state representation for performing high-precision stabilizing actions. We do not compare to a Monolithic baseline (as in Section 3.1) as it achieves zero success for all tasks.

We report the average task success and standard deviation for four bimanual tasks, where all tasks require dynamic restabilizing. Task success is measured as the proportion of the task completed over the total amount to be completed, for example the zipped length over the total zipper length. As shown in Table 1, BUDS is able to achieve a 76.9% success rate across these four tasks, and we visualize rollouts for each of the restabilizing tasks in Fig. 3. We report four failure modes: (1) an incorrect predicted stabilizing position w^s, (2) an acting policy failure π^a, (3) a restabilizing error f^r_ψ that does not detect when a stabilizing point is no longer effective, and (4) a failed grasp. The most common failure is the acting policy error, due to the low amount of data used to train the acting policy and the high precision required. The stabilizing failures (w^s and f^r_ψ) are predominantly caused by large visual differences between the task environment and the training data, including heavy occlusions. With these failures, the environment is no longer stable and the scene quickly moves out

of the distribution of the stable, simplified states seen during the acting policy's training. Across all tasks, BUDS outperforms the unstructured BC-Stabilizer baseline due to the high precision required of a stabilizing role. Where BUDS learns a relevant point from a visual input, the BC-Stabilizer baseline must both learn to recognize a relevant stabilizing position and learn the policy to travel to this position. Thus, the BC-Stabilizer baseline struggles to learn a robust stabilizing policy with the wide diversity of possible task configurations, as indicated by its 20.9% success rate.

We stress test the generalizability of our system on out-of-distribution (OOD) objects completely unseen during training, classified into two classes based on visual similarity to in-distribution objects (See Fig. 2). We run 10 trials for each object, and report BUDS is able to achieve an average success rate of 52.7%, as shown in Table 2. In two of the three tasks, we observe a slight drop in performance compared to the in distribution settings of Table 1, with a worsening difference for the Hard objects. Along with this expected performance drop, we observe an increase in the stabilizing failure modes (w^s and f^r_ψ). This is due to the stabilizing policy's high visual dependence, which makes it difficult to generalize to novel object appearances. For the Jacket Zip task, we attempt to improve performance by training the stabilizing keypoint model f^k_θ on three jackets, but the policy still struggles on the generalizing to the vastly different Hard black jacket. One exception to this trend is the Easy zucchini for the Cut Vegetable task, where the success rate is higher than that of the in-distribution jalapeño. This is due to the simplified dynamics of the solid zucchini compared to the hollow jalapeño, which is unforgiving of slight errors in the acting policy and often twists or tears if the cut actions are imprecise. The zucchini, however, can withstand shear forces from noisy policies, yielding a higher success rate.

Task	**BUDS OOD**		**Failures**			
	Easy	*Hard*	w_s	π^a	f^r_ψ	G
Jacket Zip	0.623 ± 0.40	0.288 ± 0.27	10	3	0	2
Marker Cap	0.600 ± 0.14	0.533 ± 0.39	17	1	0	0
Cut Vegetable	0.850 ± 0.13	0.266 ± 0.26	4	6	0	6

Table 2: **Generalizability Physical Robot Experiments:** We test the generalizability of BUDS across objects of similar morphology. For three tasks, we consider two classes of OOD objects—Easy and Hard—which are respectively more and less similar in visual appearance and dynamics to the training objects (see Fig. 2). We report the average and standard deviation task success achieved over ten trials per object, along with failure modes over 20 trials per task.

6 Conclusion

We present BUDS, a system for performing dexterous bimanual manipulation that leverages a novel role assignment paradigm: a stabilizing arm holds a point stationary for the acting arm to act in a simplified environment. BUDS uses a learned keypoint as the stabilizing point and avoids the need for bimanual demonstrations by learning an acting policy from 20 unimanual trajectory demonstrations in a stabilized environment. During task execution, BUDS employs a learned restabilization classifier to determine when a stabilizing keypoint is no longer effective and an updated stabilization keypoint should be obtained. BUDS achieves 76.9% success on four dexterous bimanual tasks, and generalizes to objects unseen during training with 52.7% success.

Limitations and Future Work. BUDS struggles with novel objects with largely different visual appearances from those seen during training. For example, BUDS can zip various jackets but struggles with dresses, even though the task is similar. Because BUDS learns a visual acting policy from only 20 demonstrations, it falls short when performing fine-grained tasks where tactile feedback is critical, such as plugging in a USB. Lastly, BUDS grasps all points with a parallel gripper, which would not work when more nuanced stabilizing is required, such as to open a Tupperware. A more skilled grasping policy could even benefit current tasks as shown by the grasping failure mode (Table 1). In future work, we hope to explore object-centric representations, including learning object affordances, in pursuit of learning an object-class-wide stabilizing policy. Using relevant object-centric representations would also shrink the acting policy's dimensionality, which may improve performance on high-precision tasks. We will incorporate tactile sensing and consider other end effectors, such as the Shadow Hand, to perform even more complex tasks like buttoning a shirt.

References

[1] D. P. Losey, M. Li, J. Bohg, and D. Sadigh. Learning from My Partner's Actions: Roles in Decentralized Robot Teams. In *Conf. on Robot Learning (CoRL)*, 2019.

[2] E. Ng, Z. Liu, and M. Kennedy III. It Takes Two: Learning to Plan for Human-Robot Cooperative Carrying. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2023.

[3] J. Grannen, Y. Wu, S. Belkhale, and D. Sadigh. Learning Bimanual Scooping Policies for Food Acquisition. In *Conf. on Robot Learning (CoRL)*, 2022.

[4] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard, C. Karen Liu, J. Peters, S. Song, P. Welinder, and M. White. Sim2Real in Robotics and Automation: Applications and Challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2):398–400, 2021. doi:10.1109/TASE.2021.3064065.

[5] Z. Fu, X. Cheng, and D. Pathak. Deep Whole-Body Control: Learning a Unified Policy for Manipulation and Locomotion. *CoRL*, 2022.

[6] Y. Chen, Y. Yang, T. Wu, S. Wang, X. Feng, J. Jiang, S. M. McAleer, H. Dong, Z. Lu, and S.-C. Zhu. Towards Human-Level Bimanual Dexterous Manipulation with Reinforcement Learning, 2022.

[7] S. Kataoka, S. K. S. Ghasemipour, D. Freeman, and I. Mordatch. Bi-Manual Manipulation and Attachment via Sim-to-Real Reinforcement Learning, 2022.

[8] S. Stepputtis, M. Bandari, S. Schaal, and H. B. Amor. A system for imitation learning of contact-rich bimanual manipulation policies. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[9] Íñigo Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba. A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robotics and Computer-Integrated Manufacturing*, 81:102517, 2023. ISSN 0736-5845.

[10] O. Kroemer, S. Niekum, and G. Konidaris. A Review of Robot Learning for Manipulation: Challenges, Representations, and Algorithms. *Journal of Machine Learning Research*, 22(30): 1 – 82, January 2021.

[11] L. P. Ureche and A. Billard. Constraints extraction from asymmetrical bimanual tasks and their use in coordinated behavior. *Robotics and Autonomous Systems*, 103:222–235, 2018. ISSN 0921-8890. doi:https://doi.org/10.1016/j.robot.2017.12.011.

[12] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic. Dual arm manipulation—A survey. *Robotics and Autonomous Systems*, 60(10):1340–1353, 2012. ISSN 0921-8890. doi:https://doi.org/10.1016/j.robot.2012.07.005.

[13] R. Lioutikov, O. Kroemer, G. Maeda, and J. Peters. Learning manipulation by sequencing motor primitives with a two-armed robot. 302:1601–1611, 01 2016. doi:10.1007/978-3-319-08338-4_115.

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

[15] F. Xie, A. Chowdhury, M. C. D. P. Kaluza, L. Zhao, L. L. S. Wong, and R. Yu. Deep imitation learning for bimanual robotic manipulation. 2020.

[16] N. Figueroa and A. Billard. Learning Complex Manipulation Tasks from Heterogeneous and Unstructured Demonstrations. In Proceedings of Workshop on Synergies between Learning and Interaction. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017.

[17] A. Batinica, B. Nemec, A. Ude, M. Raković, and A. Gams. Compliant movement primitives in a bimanual setting. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 365–371, 2017. doi:10.1109/HUMANOIDS.2017.8246899.

[18] A. Colomé and C. Torras. Dimensionality Reduction for Dynamic Movement Primitives and Application to Bimanual Manipulation of Clothes. *IEEE Transactions on Robotics*, 34(3): 602–615, 2018. doi:10.1109/TRO.2018.2808924.

[19] A. Colomé and C. Torras. *Reinforcement Learning of Bimanual Robot Skills*. Springer Cham, 2020.

[20] G. Franzese, L. de Souza Rosa, T. Verburg, L. Peternel, and J. Kober. Interactive imitation learning of bimanual movement primitives, 2022.

[21] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta. Efficient Bimanual Manipulation Using Learned Task Schemas. In *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2020.

[22] W. Z. Wang, A. Shih, A. Xie, and D. Sadigh. Influencing towards stable multi-agent interactions. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[23] J. Grannen, P. Sundaresan, B. Thananjeyan, J. Ichnowski, A. Balakrishna, M. Hwang, V. Viswanath, M. Laskey, J. E. Gonzalez, and K. Goldberg. Untangling Dense Knots by Learning Task-Relevant Keypoints. In *Conf. on Robot Learning (CoRL)*, 2020.

[24] Y. Avigal, L. Berscheid, T. Asfour, T. Kröger, and K. Goldberg. SpeedFolding: Learning Efficient Bimanual Folding of Garments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2022.

[25] A. Ganapathi, P. Sundaresan, B. Thananjeyan, A. Balakrishna, D. Seita, J. Grannen, M. Hwang, R. Hoque, J. E. Gonzalez, N. Jamali, et al. Learning Dense Visual Correspondences in Simulation to Smooth and Fold Real Fabrics. *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2021.

[26] F. Amadio, A. Colomé, and C. Torras. Exploiting Symmetries in Reinforcement Learning of Bimanual Robotic Tasks. *IEEE Robotics and Automation Letters*, 4(2):1838–1845, 2019. doi:10.1109/LRA.2019.2898330.

[27] X. Yin and Q. Chen. Learning nonlinear dynamical system for movement primitives. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3761–3766, 2014. doi:10.1109/SMC.2014.6974516.

[28] L. Fu, H. Huang, L. Berscheid, H. Li, K. Goldberg, and S. Chitta. Safely Learning Visuo-Tactile Feedback Policies in Real For Industrial Insertion, 2022.

[29] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta. Intrinsic Motivation for Encouraging Synergistic Behavior. In *International Conference on Learning Representations*, 2020.

[30] H. Ha and S. Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conf. on Robot Learning (CoRL)*, 2021.

[31] R. Zöllner, T. Asfour, and R. Dillmann. Programming by Demonstration: Dual-Arm Manipulation Tasks for Humanoid Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 479–484, 2004.

[32] P. Hsu. Coordinated control of multiple manipulator systems. *IEEE Transactions on Robotics and Automation*, 9(4):400–410, 1993. doi:10.1109/70.246051.

[33] S. S. Mirrazavi Salehian, N. Figueroa, and A. Billard. Dynamical System-Based Motion Planning for Multi-Arm Systems: Reaching for Moving Objects. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 4914–4918, 2017. doi:10.24963/ijcai.2017/693.

[34] P. Lertkultanon and Q.-C. Pham. A certified-complete bimanual manipulation planner. In *IEEE Transactions on Automation Science and Engineering*, pages 1355–1368, 2018. doi:10.1109/TASE.2018.2791478.

[35] J. Gudiño Lau. Dynamic model and simulation of cooperative robots: A case study. *Robotica*, 23:615 – 624, 09 2005. doi:10.1017/S0263574704001213.

[36] N. Xi, T.-J. Tarn, and A. Bejczy. Intelligent planning and control for multirobot coordination: An event-based approach. *IEEE Transactions on Robotics and Automation*, 12(3):439–452, 1996. doi:10.1109/70.499825.

[37] C. Bersch, B. Pitzer, and S. Kammel. Bimanual robotic cloth manipulation for laundry folding. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1413–1419, 2011. doi:10.1109/IROS.2011.6095109.

[38] K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *European Conference on Computer Vision*, pages 630–645. Springer International Publishing, 2016.

[39] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte, et al. imgaug. https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

[40] Dec 2022. URL https://3dconnexion.com/us/product/spacemouse-compact/.

[41] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[42] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conf. on Robot Learning (CoRL)*, 2021.