Parse-Augment-Distill: Learning Generalizable Bimanual Visuomotor Policies from Single Human Video

Georgios Tziafas[1*], Jiayun Zhang[1], Hamidreza Kasaei[1]

***Abstract*—Learning visuomotor policies from expert demonstrations is an important frontier in modern robotics research, however, most popular methods require copious efforts for collecting teleoperation data and struggle to generalize out-of-distribution. Scaling data collection has been explored through leveraging human videos, as well as demonstration augmentation techniques. The latter approach typically requires expensive simulation rollouts and trains policies with synthetic image data, therefore introducing a sim-to-real gap. In parallel, alternative state representations such as keypoints have shown great promise for category-level generalization. In this work, we bring these avenues together in a unified framework: PAD (Parse-Augment-Distill), for learning generalizable bimanual policies from a single human video. Our method relies on three steps: (a) parsing a human video demo into a robot-executable keypoint-action trajectory, (b) employing bimanual task-and-motion-planning to augment the demonstration at scale without simulators, and (c) distilling the augmented trajectories into a keypoint-conditioned policy. Empirically, we showcase that PAD outperforms state-of-the-art bimanual demonstration augmentation works relying on image policies with simulation rollouts, both in terms of success rate and sample/cost efficiency. We deploy our framework in six diverse real-world bimanual tasks such as pouring drinks, cleaning trash and opening containers, producing one-shot policies that generalize in unseen spatial arrangements, object instances and background distractors.**

*Corresponding Author { g.t.tziafas@rug.nl }
[1] Department of Artificial Intelligence, University of Groningen, the Netherlands

I. INTRODUCTION

Visuomotor policy learning for robot manipulation has seen great success in recent years [1]–[5], yet it typically demands costly and time-consuming data collection from expert demonstrators. This data-hungriness stems from the different required axes of generalization: a competent policy must generalize in unseen object arrangements (*spatial*) and object instances (*object*), as well as be robust to environmental conditions such as scene background, camera placement etc. (*background*). As a result, most common policies struggle to generalize in out-of-distribution scenarios where corresponding data has not been collected. A recent methodology to tackle this data scarcity is to tap into the vast repository of videos available in the web, showcasing humans interacting with objects in diverse scenarios [6]–[10]. Here the main challenge is bridging the embodiment gap between humans and robot morphologies [11]–[13]. Alternatively, a recent line of works aims at dealing with spatial generalization by augmenting a small number of source demos with structured, object-centric *task-and-motion planning (TAMP)* procedures [14]–[17]. However, most works train image policies that require calibrated digital twins and expensive on-robot rollouts to generate the augmentations, therefore introducing a visual sim-to-real gap, while still struggling with object and background generalization. When it comes to bimanual manipulation, additional considerations

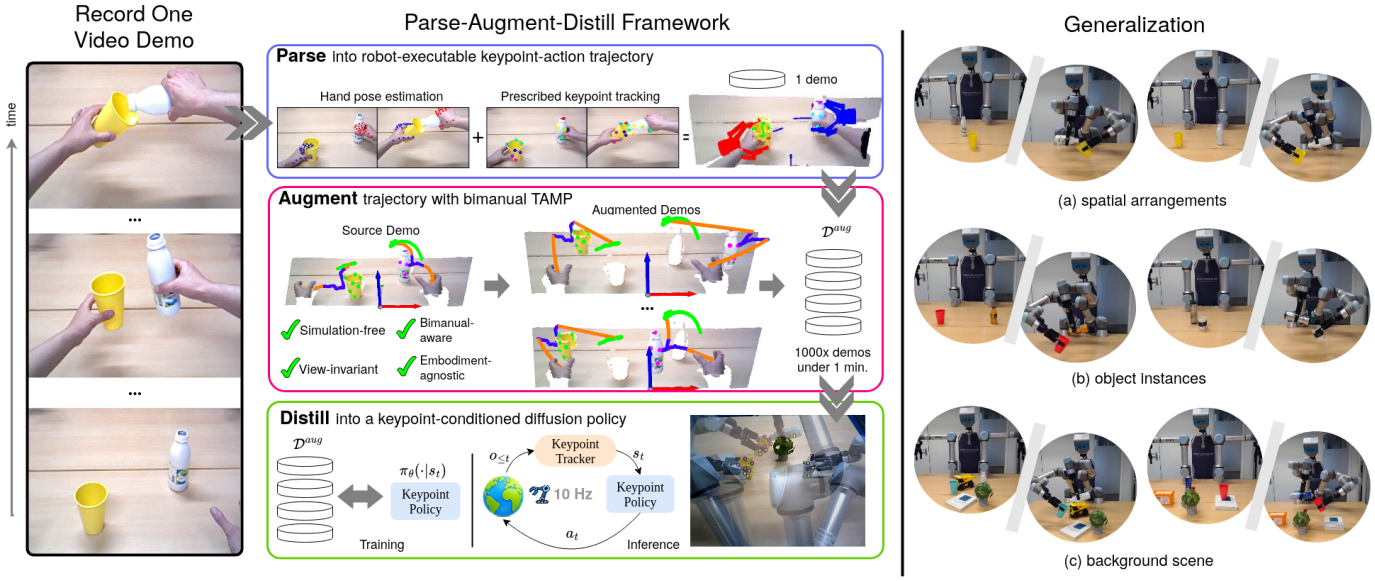


Fig. 1. **PAD framework overview:** Given one human video demonstration, PAD executes three subsequent steps: (a) parsing the video into a robot-executable state-action trajectory, (b) spatially augmenting the demo at scale via bimanual TAMP, and (c) distilling the generated data into a closed-loop keypoint policy. The obtained policy can generalize to unseen spatial arrangements, object instances and background scene noise.

related to arm collaboration strategies for different task scenarios further complicate data collection / generation.

In this work we wish to tackle these challenges by proposing **PAD** (**P**arse-**A**ugment-**D**istill), a unified framework for learning bimanual visuomotor policies from a single human video demonstration. Our framework works in three steps (see Fig. 1): (a) parsing the video into robot-executable data, (b) augmenting the data in a simulation-free fashion and, (c) distilling the augmented data into a closed-loop policy.

In our work, we explicitly seek spatial, object and background generalization. To accommodate this, we utilize 3D keypoint coordinates as state representations for our trained policy, which offers three important advantages: First, keypoints abstract the visual scene into a low-dimensional geometric representation, which is task-specific and decoupled from object semantics, and therefore has empirically shown to aid in sample-efficiency and robustness to background noise [13], [18], [19]. Second, keypoints facilitate category-level object generalization, inherited by the open-world capabilities of pretrained vision models for identifying semantic correspondences [20]–[22]. Finally, 3D point states enable efficient spatial augmentations, as keypoint coordinates can be computed on-the-fly through 3D rigid geometry assumptions [23]. This alleviates the need for a digital twin and expensive simulation rollouts, which would be required by a typical image policy to obtain image observations [14]–[17]. In turn, this significantly improves data collection time and bridges the sim-to-real gap introduced by simulators.

Concretely, in PAD we introduce a general TAMP framework for spatial demo augmentations, specialized for bimanual manipulation. To that end, we introduce bimanual task templates, symbolic representations that declare information about each arm's object assignments, involved contacts and requirements for arm synchronization, while abstracting away the specific semantics of the task. We particularly focus on handling issues related to bimanual manipulation, such as out-of-range arm-object assignments and re-synchronization between the arms during motion planning, which are missing from previous works in bimanual demo augmentation [17]. Our augmentation framework is general, cost-efficient and embodiment-agnostic, as it uses human video as the source demo that can be mapped to any given morphology. Finally, we use prescribed 3D keypoints as our state representation instead of RGB images or point-clouds, and accompany them with augmentations that aid the policy in object generalization. To distill the augmented data, we introduce Kp-RDT, an adapted version of RDT [4] for learning bimanual diffusion policies with keypoint conditioning.

Empirically, we show that our framework outperforms state-of-the-art bimanual demo augmentation methods [17] in four simulation tasks from the DexMimicGen *robosuite* benchmark [24], both in terms of success rates, as well as sample-efficiency and data generation time. We further apply our framework in six diverse real-world tasks and show that PAD obtains policies that generalize to unseen spatial arrangements, object instances and background scene noise, while doing so from a single human demonstration.

In summary, our contributions with this work are threefold:

- We introduce PAD, a unified framework for generalizable bimanual policy learning from a human video.
- We propose a general bimanual TAMP framework for spatial demo augmentations, applicable to a wide variety of manipulation skills and arm-coordination strategies, as well as open-ended object categories.
- We perform extensive robot experiments in 10 tasks, 4 in simulation and 6 with hardware, demonstrating significant gains compared to previous works in terms of success rates and sample/cost efficiency, as well as strong generalization in real-world tasks.

II. RELATED WORK

Learning from egocentric video. Many recent datasets [6]–[10] represent large-scale efforts to collect egocentric hand-object interaction videos in diverse real-world scenes. Such datasets have been used for learning robotics-tailored visual representations as a pretraining step [25]–[30]. Other works learn coarse policies from human videos, using keypoints [31] or generative modeling [32], which will later be fine-tuned for specific embodiments and tasks. The recent work MT-π [11] proposes to co-train a keypoint policy with human videos and robot data simultaneously. All the above lines of work still require robot data to obtain a performant policy, typically collected through costly teleoperation. Phantom [12] proposes to edit human videos with generative models to map them into robot data, while other works propose to utilize hardware platforms such as smart glasses [33] or MoCaP-like setups [34] to collect video data for dexterous manipulation. Recent works [13], [19], [35] unify human video and robot data through point-based representations for both objects and hands. However, these works do not consider demo augmentations and hence require a lot of videos to train policies that generalize. In our work we introduce demo augmentations for bimanual manipulation and obtain generalizable policies from a single human video.

Keypoint-based representations. Finding correspondences between objects [36] has been extensively explored for robotics applications, either analytically [37], [38] or with data-driven methods [39]–[43]. Although data-driven approaches have shown better generalization, they require additional training data such as 3D object shape or labeled keypoint datasets. More recently, several works have shown that the emergent capabilities of modern vision models [20]–[22] for identifying keypoint correspondences in-the-wild serves as a powerful proxy for category-level object generalization [44]–[49]. The recent work KALM [18] utilizes multimodal large language models (MLLMs) [50], [51] to automatically discover keypoint annotations in robot data for training and employs open-loop keypoint-conditioned policies that show good generalization. All above works need to collect robot demonstrations for specific tasks and do not consider augmentations or human video. A recently emerging line of works [52], [53] combines keypoint correspondence from vision models with the in-context learning capabilities of LLMs to transfer policies from human video after retrieval. Such MLLM policies can only be deployed open-loop, due to the

costly nature of MLLM inference, and hence are not reactive to scene changes. Further, as MLLMs have not been trained with spatial data, they struggle to generalize to novel spatial arrangements and require dense scene coverage to obtain useful in-context demos. Closer to our work, recent works [13], [19], [35] parse keypoints from human demo videos and train keypoint-conditioned closed-loop policies, operating on point tracks predicted by an off-the-shelf tracker [54]. All above works require many videos to train generalizable policies and have mostly not been deployed for bimanual manipulation. In our work, we train keypoint-conditioned policies on spatial augmentations performed on a single human demo video, offering sample-efficient and generalizable bimanual policies without need for teleoperation or extensive human videos.

Demonstration augmentation A recent line of works attempts to generate demonstrations from scratch using LLMs for task decomposition into sub-goals and motion primitives or reinforcement learning (RL) for completing the sub-goals [55]–[57]. However, the range and quality of the resulting data is often restricted by the capacity of the underlying planning and RL modules. The seminar work MimicGen [14] and its follow-ups [15]–[17] provide a more effective alternative. Instead of generating data from scratch, MimicGen performs augmentations on human-collected demonstrations to adapt for novel spatial configurations, by decomposing and re-synthesizing corresponding execution plans. DexMimicGen [17] extends MimicGen's strategy to support bimanual tasks and robot platforms. However, plans produced by the MimicGen family [14]–[17] are not ready-to-use demonstration data in the form of state-action pairs, and on-robot execution is necessary for collecting image observation data, required by the downstream policy. Therefore, they rely on rolling-out their execution plans on digital twin platforms, which bottlenecks data collection time while introducing additional sim-to-real challenges for real-world deployment. Closer to our work, DemoGen [23] generates ready-to-use augmentations in a cost-effective manner. As it considers 3D point-clouds as state representations for downstream policy learning, DemoGen supports obtaining the augmented states in a simulation-free fashion, simply via 3D rigid geometry. However, this approach only tackles spatial generalization, as point-cloud policies still struggle to generalize in novel object instances. Further, the DemoGen framework only considers sequential single-arm tasks, which cannot be applied out-of-the-box for bimanual manipulation, while still requiring a few source demonstrations collected via teleoperation. In our work, we extend DemoGen to handle bimanual manipulation tasks in a general framework, while considering keypoint states (obtained via parsing video) for downstream policy learning, thus also enabling category-level object generalization and robustness to background noise.

III. METHOD

Our problem setup states that given a single recorded video V_M, demonstrating an expert behavior for completing a bimanual task M, our goal is to obtain a policy π_M, trained solely from the source video, that can autonomously complete the task. We explicitly seek generalization to novel spatial arrangements, object instances and background scene noise. In the following sections we describe our framework, *PAD* (see schematic in Fig. 1), which decomposes the problem in three steps: (a) parsing the video into a robot-executable state-action trajectory (Sec. III-A), (b) augmenting the trajectory at scale via bimanual TAMP (Sec. III-B), and (c) distilling the generated data into a keypoint-conditioned diffusion policy (Sec. III-C).

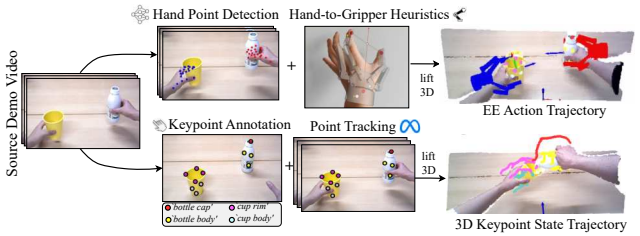


Fig. 2. **Parse source video into state-action trajectory:** We use several off-the-shelf computer vision assets such as hand pose estimators and 2D point trackers, in-tandem with hand-to-gripper heuristics, to parse the source video into 3D keypoint and absolute end-effector action trajectories.

Demonstration setup and assumptions We record our demo video with an RGB-D camera, placed under the same orientation from the table as in our robot setup. We assume that a calibration procedure is performed, such that the extrinsics matrix of the recording camera relative to the tabletop matches the one of the robot. We call the result of this single camera-to-tabletop transform the *task frame*, which we use to express all keypoint / action trajectories parsed from the video. We also assume that the human demonstrator provides motion trajectories that lie within the robot's valid kinematic workspace.

A. Parsing Video into Robot-Executable Data

The goal of this step is to parse the source RGB-D video $V_M = \{(I_t, D_t)\}_{t=0}^{L-1}$, where L the total number of frames (recorded at 30 fps), into the corresponding state-action transition trajectory $\zeta_M^{\text{demo}} = \{(s_t, a_t)\}_{t=0}^{L-1}$. In this work we define:

$$s_t := \mathbf{P}_t^{kp} \in \mathbb{R}^{N\times 3}$$

$$a_t := (\mathbf{T}_t^0, g_t^0, \mathbf{T}_t^1, g_t^1) \in SE(3)^2 \times \{0,1\}^2$$

The state is represented by the 3D coordinates of N prescribed keypoints, while the action as a pair of two end-effector (EE) poses $\mathbf{T}_t^j = (\mathbf{R}_t^j | \mathbf{x}_t^j) \in$ SE(3) and gripper open/close commands $g_t^j \in \{0,1\}$. Here and for the rest of the manuscript we use the superscript $j \in \{0,1\}$ to denote each arm (e.g. 0-left and 1-right). The overall system for extracting keypoints and EE actions is illustrated in Fig. 2.

Annotating and tracking keypoints To obtain keypoint annotations, we adopt the method proposed by [19]. In particular, a user selects semantically meaningful 2D points on task-relevant objects in the first frame of the video I_0 through a GUI. We also ask the user to provide a text label for groups of keypoints, corresponding to distinct object-parts that participate in contacts at different stages of the task (e.g. *'bottle body'* for grasping the bottle and *'bottle cap'* for pouring from it – see bottom left of Fig. 2). These

will become useful downstream as positional embeddings for policy learning. Next, we employ an off-the-shelf point tracker, Co-Tracker3 [54], to automatically track the initialized keypoints in the rest of the video frames $I_{t>0}$. The depth maps D_t are used to back-project the 2D keypoint coordinates to 3D. To account for noise in sensor depth, we consider a 2D window around each back-projected pixel and select the median value after removing outliers. Finally, we use the camera extrinsics to transform the keypoint tracks to task frame, resulting in the final 3D keypoint states $\mathbf{P}^{kp}_{0:L-1}$.

Mapping hands to robot actions Following previous works [53], we use HaMeR [58], a recent model trained for hand pose estimation in egocentric hand-object interaction videos [6]. Hand poses are parameterized through 21 points that correspond to the joints of the human hand according to the MANO model [59] (see top left of Fig. 2). We run HaMeR separately for each frame I_t to obtain both 2D pixel and 3D point coordinates for each hand $\mathbf{p}^j_{t,k}$, $k = 0, \ldots, 20$. Depth maps D_t are used to correct mis-calibration errors in HaMeR's 3D prediction, by matching a point which is clearly visible (e.g. wrist) with its back-projected depth and transferring the rest of the points such as their relative positions remain unchanged. To map hand points to robot poses, we use heuristics similar to the ones described in [53]. In particular, of the 21 points, we use the tip of the index finger $\mathbf{p}^j_{t,\texttt{ind}}$, tip of the thumb $\mathbf{p}^j_{t,\texttt{th}}$ and the mid-wrist point $\mathbf{p}^j_{t,\texttt{wr}}$. Then, we define the robot's EE position at the midpoint between the index and thumb tip and the rotation by aligning the line connecting the wrist and index-thumb midpoint with the gripper's approach direction (see top left of Fig. 2). Formally, an EE pose $\mathbf{T}^j_t = (\mathbf{R}^j_t | \mathbf{x}^j_t)$ is given by:

$$\mathbf{x}^j_t = \tfrac{1}{2}(\mathbf{p}^j_{t,\texttt{th}} + \mathbf{p}^j_{t,\texttt{ind}})$$

$$\mathbf{a}^j_t = \mathbf{p}^j_{t,\texttt{ind}} - \mathbf{p}^j_{t,\texttt{th}},\ \mathbf{b}^j_t = \mathbf{x}^j_t - \mathbf{p}^j_{t,\texttt{wr}},\ \mathbf{v}^j_t = \mathbf{a}^j_t \times \mathbf{b}^j_t \quad (1)$$

$$\mathbf{R}^j_t = \mathbf{I}_{3\times 3} + [\hat{\mathbf{v}}^j_t]_\times + \frac{1 - \hat{\mathbf{a}}^j_t \cdot \hat{\mathbf{b}}^j_t}{\|\hat{\mathbf{v}}^j_t\|^2} \cdot [\hat{\mathbf{v}}^j_t]^2_\times$$

where $\times$ denotes cross-product, $\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$ unit-length normalization and $[\mathbf{v}]_\times$ the skew-symmetric matrix of $\mathbf{v}$ [60].

To facilitate robot-executability, we use an Inverse Kinematics (IK) solver to identify "jumps" in the parsed action trajectory that lead to kinematic failures or collisions. We replace those actions with collision-free poses interpolated from the rest of the trajectory with cubic splines [61]. This process smoothens the action trajectory and ensures executability, given that most of the recorded trajectory is within the robot's kinematic range.

B. Trajectory Augmentation via Bimanual Task-and-Motion Planning

The goal of this step is to generate a large-scale dataset $\mathcal{D}^{aug}_M = \{\tilde{\zeta}^{(i)}\}$, where each trajectory $\tilde{\zeta}$ is generated by performing spatial augmentations over the source demo $\zeta^{\texttt{demo}}_M$. Like previous works [14], [23], the augmentations are orchestrated by a TAMP procedure, where the action trajectory is split into *motion* and *skill* segments. Motion segments correspond to the robot moving in free space (e.g. approaching the bottle), while skill segments to robot-object or object-object interactions that manifest a manipulation skill (e.g. grasping the bottle, pouring from bottle to cup). The main idea is to cover the entire workspace by moving the task-relevant objects around, and appropriately adapt the actions in each motion and skill segment through motion planners and SE(3)-equivariant transforms respectively. In PAD we extend this methodology for bimanual tasks, taking special care of related issues that arise with arm collaboration. In the rest of the section we provide step-by-step details of our augmentation framework.

Bimanual task templates The demonstrated bimamual manipulation task M is abstracted into a symbolic template Π_M, which contains: (a) a set of integers $\{1, ..., K\}$, each corresponding to a unique task-relevant object in the scene, and (b) a list of high-level actions, each represented by a *contact*, i.e. a pair sampled from the set $\{ee^0, ee^1\} \cup \{1, \ldots, K\}$ indicating a robot-object or object-object contact, as well as a *reference* index $k \in [0, K]$, indicating whether the action depends on some object's local reference frame ($k > 0$) or not ($k = 0$). Each element in the list can include two actions (one for each arm), indicating parallel execution of two asynchronous motions, or a single bimanually-synchronized action. Examples of task templates are illustrated in Fig. 3.

The task templates serve as essential information for grounding segments in the subsequent TAMP procedure, by means of: (a) decomposing the task into distinct object-centric stages, (b) declaring hand-object assignments, (c) assigning related reference frames to each stage-arm pair, and (d) declaring stages that require bimanual synchronization. The task templates are general, i.e. they don't depend on the semantics of the appearing objects and dont require skill labels. Instead, they provide a high-level overview of the task in terms of what contacts with / between objects take place, in what order, and whether the arms need to be synchronized. Further, they are composable, as arbitrary long-horizon tasks can be constructed by chaining templates for specific short-horizon skills. In practice, we obtain Π_M by feeding sub-sampled frames of the video with a few in-context examples to a MLLM [51], akin to [53].

Grounding timestamp segments Our next step is to ground each stage of our task template into a sequence of motion and skill segments for each arm, with precise timestamps in the parsed action trajectory. First, we use GroundedSAM [62] to segment the K task-relevant objects in the initial video frame I_0, and back-project each region to 3D to obtain a masked object point-cloud. For each object o_k, we calculate its point center and place a local reference frame there $\mathbf{T}^{o_k} \in \text{SE}(3)$, resulting in the *initial object configuration* $\mathcal{O} = [\mathbf{T}^{o_1}, \ldots, \mathbf{T}^{o_K}]$. The reference annotation k in arm-stage pairs of Π_M can then be identified with the corresponding reference frame $\mathcal{F}^j_i \in \{\mathbf{T}^{task}\} \cup \mathcal{O}$, which we call the *assigned* frame for arm j in stage $i = 1, \ldots, n$. Here, the global task frame $\mathbf{T}^{task}$ is assigned to stage-arm pairs that are annotated with reference $k = 0$. Next, for each stage, we classify demo timestamps into motion or skill segments. Following [23], we classify asynchronous skill segments, assigned to a frame $\mathcal{F}^j_i$, by calculating the distance between the frame's center

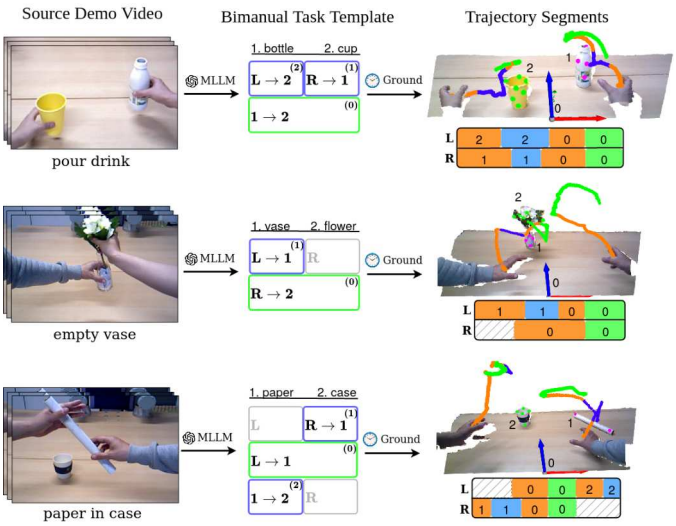


Fig. 3. **Grounding trajectory segments with bimanual task templates.** The video (*left*) is abstracted into a template (*middle*), denoting task stages (as blocks), given by contacts (arrows), related reference frames (number in parenthesis) and arm synchronization (green block) or not (blue block). Timestamp segments for each arm-stage are grounded in the demo trajectory, where orange blocks correspond to *motion*, gray blocks to *idle*, blue blocks to *asynchronous skill* and green to *synchronous skill* segments.

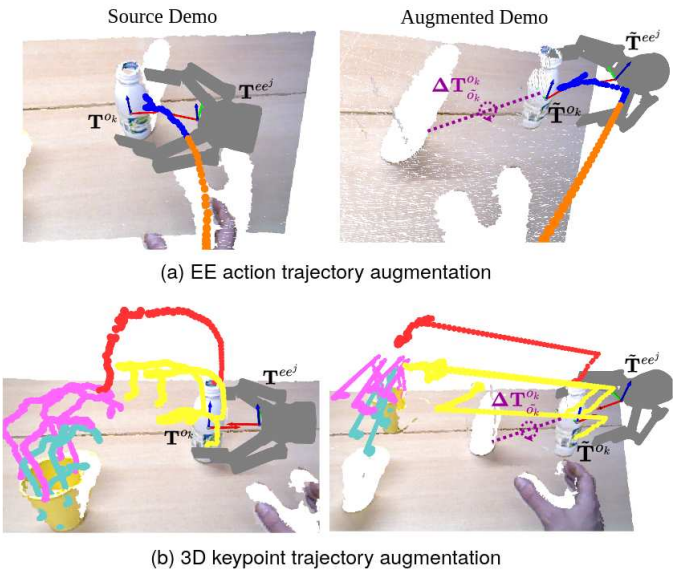


Fig. 4. **Augmenting state-action trajectories with SE(3)-equivariant transforms.** *(Top)* The EE poses are transformed according to the new object pose such as their relative pose remains invariant for the entire skill segment (in blue color). *(Bottom)* We assume the new keypoints will move rigidly with the EE pose, according to their relative pose during grasping.

and the assigned arm's EE position, and checking whether it falls within a specified threshold. Similarly, to adapt to syncrhonous stages, we classify synchronous skill segments as the timestamps whose distance between the two EE poses falls within a specified threshold. Synchronous segments will always have the exact timestamps between arms. The intermediate timestamps between two skill segments are classified as motion segments.

Formally, for a timestamp segment $\tau = [t_a, t_b] \subseteq [0, L)$, we use the bracket notation $[\cdot]$ to denote slicing [23], such that $\zeta[\tau] = (s_{t_a:t_b}, a_{t_a:t_b})$. Then, the state-action trajectory wrt. the initial configuration $\mathcal{O}$ can be written as [1]:

$$Z \parallel \mathcal{O} = [\zeta[\tau_1^m], \zeta[\tau_1^s], \ldots, \zeta[\tau_n^m], \zeta[\tau_n^s]]$$

where superscripts $\{m, s\}$ denote a motion and skill segment respectively.

With timestamp segments in place, we vary the initial object configuration $\tilde{\mathcal{O}} = [\mathbf{T}^{o_{\tilde{1}}}, \ldots, \mathbf{T}^{o_{\tilde{K}}}]$ with SE(3) transformations and obtain $\tilde{\zeta} := \tilde{Z} \parallel \tilde{\mathcal{O}}$, where $\tilde{\zeta}$ an augmented demo that has the same sequence of motion-skill segments as ζ_M^{demo}, but the state-actions in each segment are augmented according to $\tilde{\mathcal{O}}$, as described below (see also Fig. 4).

Action augmentations Let $\mathbf{\Delta T}_{o_{\tilde{k}}}^{o_k} = (\mathbf{T}^{o_k})^{-1} \cdot \mathbf{T}^{o_{\tilde{k}}}$ the relative transformation between the demo and augmented configuration of object o_k. Then, the augmented action trajectory for arm j corresponding to a skill segment τ_i^s with assigned object frame $\mathcal{F}_i^j = \mathbf{T}^{o_k}$, is given by:

$$\tilde{\mathbf{T}}^j[\tau_i^s] = \mathbf{T}^j[\tau_i^s] \cdot \mathbf{\Delta T}_{o_{\tilde{k}}}^{o_k}, \qquad (2)$$

$$\tilde{g}^j[\tau_i^s] = g^j[\tau_i^s]$$

[1]The skill segments are randomly colorized for illustration purposes, in general skill segments can arbitrarily vary between synchronous/asynchronous throughout stages.

which is an SE(3)-equivariant transform that preserves the relative pose between object and robot throughout the segment: $\mathbf{\Delta T}_{ee^j}^{o_k}[\tau_i^s] = \mathbf{\Delta T}_{\tilde{ee}^j}^{o_{\tilde{k}}}[\tau_i^s]$. This augmentation maintains an equal number of steps as in the demo skill segment and is the same for both synchronous and asynchronous segments. Notice that in cases of skill segments where there is no assigned object (i.e. $\mathcal{F}_i^j = \mathbf{T}^{task}$), it's $\mathbf{\Delta T}_{task}^{task} = \mathbf{I}_{4x4}$, so the equation yields $\tilde{\mathbf{T}}_t^j = \mathbf{T}_t^j$, i.e. the segment's trajectory is exactly replayed as in the demo, as it's independent of any object that has been transformed. In all cases the gripper signal remains unchanged, since it is invariant to the object transforms. For motion segments, the goal is to connect the last pose of the previous skill segment with the first pose of the current one, which is obtained via motion planning [2] :

$$\tilde{\mathbf{T}}^j[\tau_i^m] = \texttt{MotionPlan}\left(\tilde{\mathbf{T}}^j_{t=\tau_{i-1}^s[-1]}, \tilde{\mathbf{T}}^j_{t=\tau_i^s[0]}\right) \qquad (3)$$

$$\tilde{g}^j[\tau_i^m] = g^j_{t=\tau_i^m[0]}$$

with the demo EE poses at $t = 0$ used as the start pose for the first stage. The gripper signal is copied from the start timestep to the entire segment, as there are no contacts taking place during motion segments. The full augmented action trajectory under the new object configuration is obtained by concatenating all segments:

$$\tilde{A} \parallel \tilde{\mathcal{O}} = [\tilde{a}[\tau_1^m], \tilde{a}[\tau_1^s], \ldots, \tilde{a}[\tau_n^m], \tilde{a}[\tau_n^s]] \qquad (4)$$

where $\tilde{a}[\tau_i] = \left(\tilde{\mathbf{T}}^0[\tau_i], \tilde{g}^0[\tau_i], \tilde{\mathbf{T}}^1[\tau_i], \tilde{g}^1[\tau_i]\right)$ the augmented bimanual action trajectory for each segment.

Keypoint augmentations We make the 3D rigidity assumption [23] to augment the keypoint trajectory under the new object configuration and augmented actions. In particular,

[2]For uncluttered scenarios, linear interpolation suffices. Alternatively, third-party motion planners [26] for obstacle avoidance should be employed.

throughout skill segments $\tau^s_{1:n}$, we keep track of timestamps where grasping t_g ($\Delta g^j_t > 0$) or releasing t_r ($\Delta g^j_t < 0$) actions took place. For intermediate steps between grasp and release, we consider the object to be attached to the assigned EE and that it moves rigidly with it in 3D space, i.e. their relative pose remains invariant. If $\tilde{\mathbf{T}}^j_{t_g}$ the augmented EE pose during attachment of arm j with object o_k and $\mathcal{P}^{o_k}_t$ the homogeneous coordinates of all N_k keypoints that belong to that object, then the augmented keypoints in a future step t within successive $[t_g, t_r]$ intervals are given by:

$$\tilde{\mathcal{P}}^{o_k}_t = \tilde{\mathbf{T}}^j_t \cdot (\tilde{\mathbf{T}}^j_{t_g})^{-1} \cdot \tilde{\mathcal{P}}^{o_k}_{t_g} \tag{5}$$

Between release and grasp intervals $[t_r, t_g]$, the keypoints are kept static since there is no attachment: $\tilde{\mathcal{P}}^{o_k}_t = \tilde{\mathcal{P}}^{o_k}_{t_r}$, with $\tilde{\mathcal{P}}^{o_k}_0 = \mathbf{\Delta T}^{o_k}_{\tilde{o}_k} \cdot \mathcal{P}^{o_k}_0$ for the start of the episode $t = 0$. Coordinates $\tilde{\mathcal{P}}^{o_k}_t \in \mathbb{R}^{4\times N_k}$ are converted back to non-homogeneous $\tilde{\mathbf{P}}^{o_k}_t \in \mathbb{R}^{N_k\times 3}$ and concatenated across all objects' keypoints: $\tilde{s}_t = [\tilde{\mathbf{P}}^{o_1}_t, \ldots, \tilde{\mathbf{P}}^{o_K}_t] = \tilde{\mathbf{P}}^{kp}_t$. Applying this to all segments we obtain the augmented state trajectory:

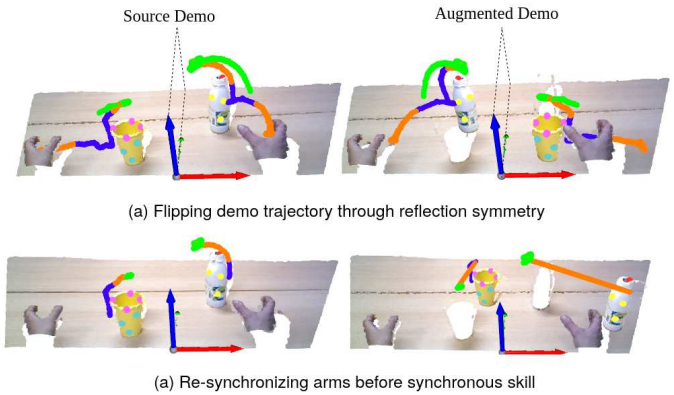
$$\tilde{S} \parallel \tilde{\mathcal{O}} = [\tilde{s}[\tau^m_1],\, \tilde{s}[\tau^s_1], \ldots, \tilde{s}[\tau^m_n],\, \tilde{s}[\tau^s_n]] \tag{6}$$

and finally the entire augmented state-action trajectory by combining equations (4) and (6):

$$\tilde{Z} \parallel \tilde{\mathcal{O}} = [(\tilde{s}[\tau^m_1], \tilde{a}[\tau^m_1]), \ldots, (\tilde{s}[\tau^s_n],\, \tilde{a}[\tau^s_n])] \tag{7}$$

Handling bimanual issues Unlike previous works [17], which assume the same number of steps between demo-augmented motion segments (and hence small perturbations in the objects' augmented poses), in this work we deal with the case where objects have been moved significantly further than their original configuration, resulting in unfeasible motion plans within the demo's number of steps. We use a constant velocity to infer the number of steps for each motion segment based on the covered distance between start and goal pose. However, this process naturally breaks the two arms' synchronization. To tackle this, we compare the start timestamps of the two arms' augmented trajectories during synchronous skill segments, and repeat the last pose to the arm that is the "earliest" for the remaining deficit. This process ensures that the two arms will be in the same relative pose as in the demo when a synchronous segment starts (see Fig. 5-b). Further, some augmentations will move objects outside the kinematic range of their assigned arm. To deal with this, we constraint the transforms to only spawn within-range final object configurations, and separately augment the initial configuration to deal with novel hand-object assignments. In particular, since we have defined our task frame to be symmetric wrt. the two arms, we can obtain a mirrored version of the demo by simply reflecting keypoints and EE poses wrt. the principal plane of symmetry (see Fig. 5-a). The reflected demo and initial configuration is fed to the same TAMP procedure to generate many augmentations for the mirrored case.

Overall, our augmentation framework entertains several desirable properties: (a) it is **general**, as it can represent a broad variety of bimanual manipulation tasks, (b) it is **simulation-free**, i.e. it doesn't require simulation rollouts with digital twins and hence significantly boosts collection time (10^3 demos in under 1 min.), (c) it is **view-invariant**, as all augmentation data are generated wrt. a calibrated task frame, independent of downstream robot camera placement, (d) it is **bimanual-aware**, as it handles common issues of previous bimanual demo augmentation frameworks, such as out-of-range hand-object assignments and de-synchronization, and (e) it is **embodiment-agnostic**, as all planning is done in EE space, and any off-the-shelf motion planner can be plugged into the described TAMP framework, appropriate for a given robot morphology.



(a) Flipping demo trajectory through reflection symmetry

(a) Re-synchronizing arms before synchronous skill

Fig. 5. **Handling bimanual issues during augmentation.** *(Top)* We generate novel hand-object assignment and augment the trajectory by reflecting according to the principal symmetry plane (YZ plane in the image). *(Bottom)* The number of motion planning steps for each arm is inferred in the augmented demo such that they smoothly end up at the same place before the synchronous stage.

C. Keypoint-conditioned Bimanual Visuomotor Policy Learning

The goal of this step is to distill the generated dataset $\mathcal{D}^{aug}_M$ into an autonomous policy π_M. To facilitate learning keypoint-conditioned policies, we adapt RDT [4], a recent diffusion transformer model based on DiT [63], employed for bimanual visuomotor policy learning. Formally, the policy samples from the distribution $p(\mathbf{A}_t|\mathbf{o}_t)$, where $\mathbf{A}_t = \mathbf{a}_{t:t+H-1}$ an action chunk [3] over a prediction horizon H, and $\mathbf{o}_t = (\mathbf{q}_t, \mathbf{P}^{kp}_{t-T_o+1:t})$, where $\mathbf{q}_t$ the current proprioception state and $\mathbf{P}^{kp}_{t-T_o+1:t} \in \mathbb{R}^{N\times T_o\times 3}$ the 3D keypoint coordinates over an observation window of T_o. We use absolute EE control [23], and parameterize actions as $\mathbf{a}_t \in \mathbb{R}^{20}$, where we flatten each EE's 3D position, 6D rotation vector (first two columns of rotation matrix) [18] and gripper open/close command into a single vector representation.

Kp-RDT architecture. Our efforts focus on adapting the conditional branch of the cross-attention layers in RDT to keypoint track states, which we term *Kp-RDT* (see Fig. 6). We flatten the $(T_o, 3)$ dimensions and encode each keypoint's observation with an MLP, resulting in a token sequence $\mathbf{X}^{cond}_t \in \mathbb{R}^{N\times C}$, where C the transformer's hidden size. Here we omit positional encodings on the keypoint token sequence, which drops the requirement that keypoints will be matched in a fixed order. Instead, we use the keypoint group annotations (see Sec. III-A) to encode identifying information about each keypoint via *group embeddings*, which are added

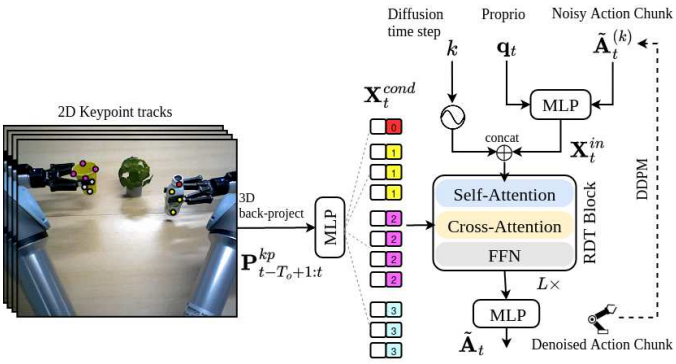


Fig. 6. **Kp-RDT architecture overview.** We adapt RDT to receive 3D keypoint track conditions. We add learnable embeddings to each keypoint's token to discriminate between distinct keypoint groups (i.e. *group embeddings*), without needing a fixed order for keypoints within a group.

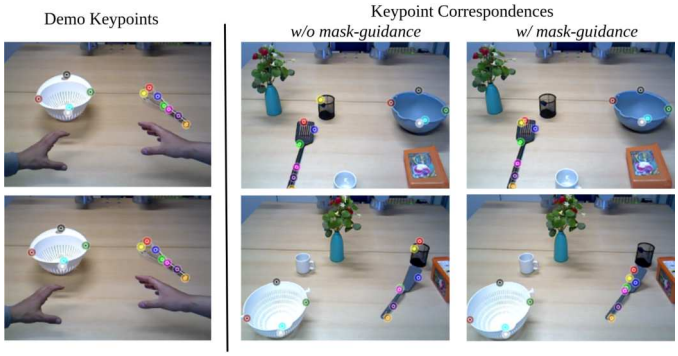


Fig. 7. **Mask-guided keypoint correspondence** improves localization in scenes with background distractors, by constraining the outputs in the corresponding object's segmentation masks.

to the keypoint token sequence after mapping all keypoints to their unique group indices, embedded in $\mathbb{R}^C$ as learnable parameters. The self-attention branch is as in RDT, where the diffusion step k is encoded with a sinusoid MLP encoder [4], while proprioception $\mathbf{q}_t$ and noisy action chunk $\tilde{\mathbf{A}}_t^{(k)}$ with the same MLP encoder, resulting in the input token sequence $\mathbf{X}_t^{in} \in \mathbb{R}^{(2+H)\times C}$. After successive application of L Kp-RDT layers, the last H transformed tokens are mapped to denoised action chunks $\tilde{\mathbf{A}}_t = f_\theta(\mathbf{o}_t, \tilde{\mathbf{A}}_t^{(k)}, k)$ with an MLP decoder.

Training & Inference. Following standard practices [4], we train the parameters θ of Kp-RDT f_θ to estimate a clean action chunk from a noisy one:

$$\mathcal{L}(\theta) := \mathcal{L}\left(\mathbf{A}_t,\ f_\theta(\mathbf{o}_t, \sqrt{\bar{\alpha}^{(k)}}\mathbf{A}_t + \sqrt{1-\bar{\alpha}^{(k)}}\epsilon, k)\right) \quad (8)$$

where $k \sim \texttt{Uniform}(1,\dots,K)$ the diffusion step, $\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ random noise, $(\mathbf{o}_t, \mathbf{A}_t) \sim \mathcal{D}_M^{aug}$ and $\bar{\alpha}^{(k)}$ coefficients pre-defined by a DDPM scheduler [64]. We train using separate L1 losses and noise schedulers for position and rotation logits and binary cross-entropy losses for the gripper logits [18]. For inference, we sample a clean action chunk by iteratively denoising for a small number of diffusion steps (e.g. K=10), starting from $\tilde{\mathbf{A}}_t^{(K)} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ [4].

To obtain keypoint states during inference, we first run keypoint correspondence with SD-DINOv2 [22] on the initial RGB observation to initialize keypoint coordinates, and then employ Co-Tracker3 [54], adapted for on-the-fly execution [13], to track the 2D coordinates in successive frames at real-time. Pixel coordinates are then back-projected to 3D and transformed to task frame to obtain $\mathbf{P}^{kp}_{t-T_o+1:t}$. Importantly, we extend the correspondence algorithm to be *mask-guided*, by first running GroundedSAM [62] to obtain pixel-wise masks of the task-relevant objects in the observed image, and then push cosine similarity between demo-observed SD-DINOv2 features to zero for areas outside the masks. We find this adaptation to significantly improve correspondence results, especially in presence of distractors (see Fig. 7).

Implementation Details. We sample state-action pairs during training at a fixed control frequency of 10 Hz. To emulate keypoint tracking errors and partial visibility, we add Gaussian noise to the keypoint coordinates and randomly dropout a subset of them during training. In practice, we mostly use an observation history of $T_o = 8$, which we find to give robust states for the policy, and an action chunk horizon of $H = 16$. We then execute only $T_a = 4$ actions (i.e. 0.4sec in 10 fps), akin to the *receding horizon planning* strategy [1]. In cases of non egocentric views (see Sec. IV-A), we observed that replacing the point tracker with the keypoint forward model based on the 3D rigidity assumption described in Sec. III-B can provide more robust tracking. Finally, our training setup can be easily adapted to obtain *trajectory-level diffusers* [18], [65], which predict the entire action trajectory from the initial keypoint state observation using a fixed number of steps, e.g. $H = 96$, and execute it open-loop with an IK motion planner. The latter implementation doesn't require online point-tracking, since it operates only on the initial observation, but the resulting policy loses its reactive behavior since it is run open-loop.

IV. EXPERIMENTS

We design our experimental setup to answer the following questions: (a) How does our augmentation framework compare to state-of-the-art bimanual demo augmentation approaches that use simulation rollouts? (b) What are the benefits in terms of task performance, data collection time and sample-efficiency? (c) Can our single video-trained policies generalize across spatial arrangements, object instances and background noise in real-world tasks? and (d) How does our proposed Kp-RDT compare to other recent keypoint-based policies in terms of success-efficiency trade-off?

We explore the first two questions by evaluating on 4 bimanual tasks of the DexMimicGen simulator [17], where we compare our approach using the generated data released by the authors. We explore the latter two questions by evaluating PAD on 6 diverse bimanual tasks recorded in a single video and executed with a real robot, focusing on a broad range of object instances to investigate generalization.

A. Simulation Experiments

We conduct experiments on 4 bimanual tasks from the DexMimicGen benchmark [17], which is based on the robosuite framework [14] that utilizes the MuJoco simulator. We focus on the bimanual Panda tasks, two with parallel jaw

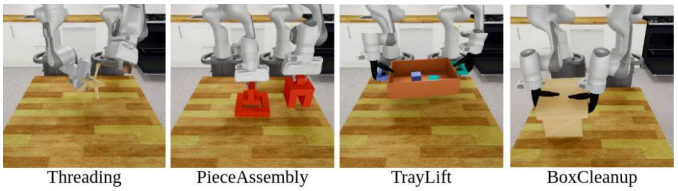


Fig. 8. **Test tasks in DexMimicGen [17] benchmark.** We evaluate on 4 bimanual Panda tasks in DexMimicGen with randomized object placements, using either parallel jaw or dexterous grippers.

gripper and two with dexterous arm (see Fig. 8). We refer the reader to the DexMimicGen paper [17] for details on the tasks. The focus of this section is to compare the augmentation pipelines between DexMimicGen and our work, in terms of final success rate, data collection time and sample-efficiency. To that end, we use the 1000 demos per task released by the authors and replay them offline to save higher-resolution RGB-D observations. We then pick a random demo (with three seeds) and use our augmentation framework to generate our own training dataset.

Baselines We use the re-collected DexMimicGen data to train three different kinds of image policies: (a) RGB (scratch), where we use the RGB observation encoded with a ResNet-18, (b) RGB-D (scratch), where we encode both RGB and depth separately with a ResNet-18 and concatenate to obtain an observation, and (c) RGB (SD-DINOv2), where we feed the RGB image to SD-DINOv2 [22] (similar to our method) to obtain feature maps as condition to the policy. For fair comparisons with our work, we use single agent-view observation and exactly the same RDT backbone [4], with only difference the type of conditioning used (image vs. keypoint). Kp-RDT is trained with our own augmentation dataset and the image policies with DexMimicGen demos, while all methods are evaluated on the same 1000 test scenes generated for each of the three source demo seeds. For tracking in Kp-RDT, we use the 3D keypoint forward model based on the 3D rigidity assumption instead of Co-Tracker, since the provided view is non egocentric and most of the objects are occluded by the robot during execution.

Sample and cost efficiency We compare performance for different number of augmented demos between the best image policy and our Kp-RDT. We run experiments for the *Threading* and *PieceAssembly* task, and report averaged success rates in Fig. 9. To compute collection time, we use the same method as described in [23]. In particular, both for our method and for DexMimicGen, we don't consider multi-processing / GPU parallelization and simply add the total time for a single process. For DexMimicGen, we multiply the average time taken in our hardware to rollout one demo with the number of demos used. Since our method doesn't need simulation rollouts, it is significantly faster, achieving generation of 10^3 demos within 11sec, compared to 33min. by DexMimicGen. Further, our Kp-RDT policy is much more sample-efficient, as observed by the higher deltas between successive experiments, especially in the range 250-500 and 500-750 demos. Notably, our policy achieves an average success rate across tasks of $\sim 60\%$ with 500 demos, while the image policy is slightly above 30%. This result clearly showcases the sample-efficiency benefits of

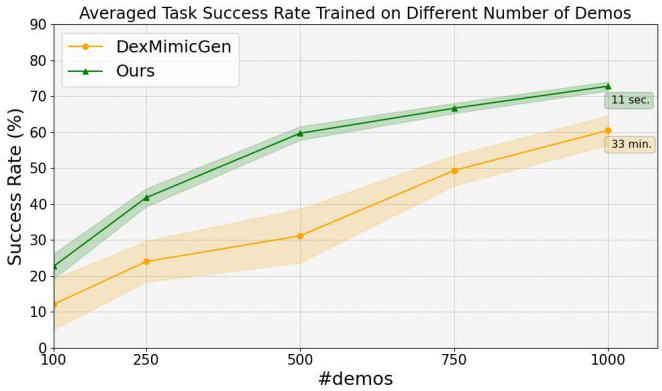


Fig. 9. **Sample-efficiency analysis**. PAD leads to higher success rates and faster learning compared to DexMimicGen due to keypoint abstractions, while doing so by generating data in a much more efficient manner. See text for details on collection time computation. Mean and std results are averaged over 3 source demo seeds.

TABLE I
IMAGE VS. KEYPOINT POLICY SUCCESS RATES (%) IN 4 SIMULATION TASKS OVER 3 SOURCE DEMO AND TEST SEEDS.

Policy	**Gen.Data**	Threading	PieceAssembly	BoxCleanup	TrayLift
RGB *(scratch)*	DexMimicGen [17]	45.0 ± 3.7	76.0 ± 4.5	86.3 ± 1.7	82.3 ± 2.0
RGB-D *(scratch)*	DexMimicGen [17]	35.3 ± 3.3	77.3 ± 2.8	86.6 ± 1.8	85.0 ± 2.1
RGB *(SD-DINOv2)*	DexMimicGen [17]	45.3 ± 3.3	77.0 ± 3.2	86.3 ± 1.7	82.0 ± 2.1
Keypoint *(SD-DINOv2)*	Ours	$\mathbf{73.9 \pm 2.1}$	$\mathbf{84.6 \pm 0.5}$	$\mathbf{94.6 \pm 0.6}$	$\mathbf{91.6 \pm 0.8}$

keypoint representations over RGB images.

Comparison with image policies We conduct experiments for all aforementioned baselines for all four tasks, using 1000 fixed initial test scenes for each of the three source demo seeds as before. We then compare the final success rates of policies trained in DexMimicGen's and our's datasets, for the total amount of demos that each method is able to generate within a fixed collection time budget. In particular, using the calculation explained previously, our policy generates $> 10^6$ augmentations within the same collection budget as DexMimicGen (33 min.). We however use early stopping when training our policy to avoid diminishing returns due to overfitting to the augmentations [17], which we observe is at average between 5 and 10 thousand demos. Results are reported in Tab. I. Our policy far exceeds the performance of all image-based baselines, with highest margin in the *Threading* task (28.9% delta).

B. Real Robot Experiments

We conduct experiments on 6 bimanual real-world tasks: (a) *pour drink*, picking a bottle and a cup with each arm and pouring from the bottle to the cup, (b) *mix bowl*, picking a whisk and a bowl with each arm and mixing the bowl with the whisk, (c) *trash in bin*, picking a trash soda can and a bin with each arm and dropping the trash in the bin, (d) *empty vase*, picking a vase with flowers with one arm and removing the flowers with the other, (e) *open container*, picking a container with one arm and removing it's lid with the other, and (f) *paper in case*, picking a paper roll with one arm, handing it over to the other arm and then placing it inside a pencil case. Our tasks are chosen to represent a broad variety of manipulation skills, different arm coordination strategies and diverse object

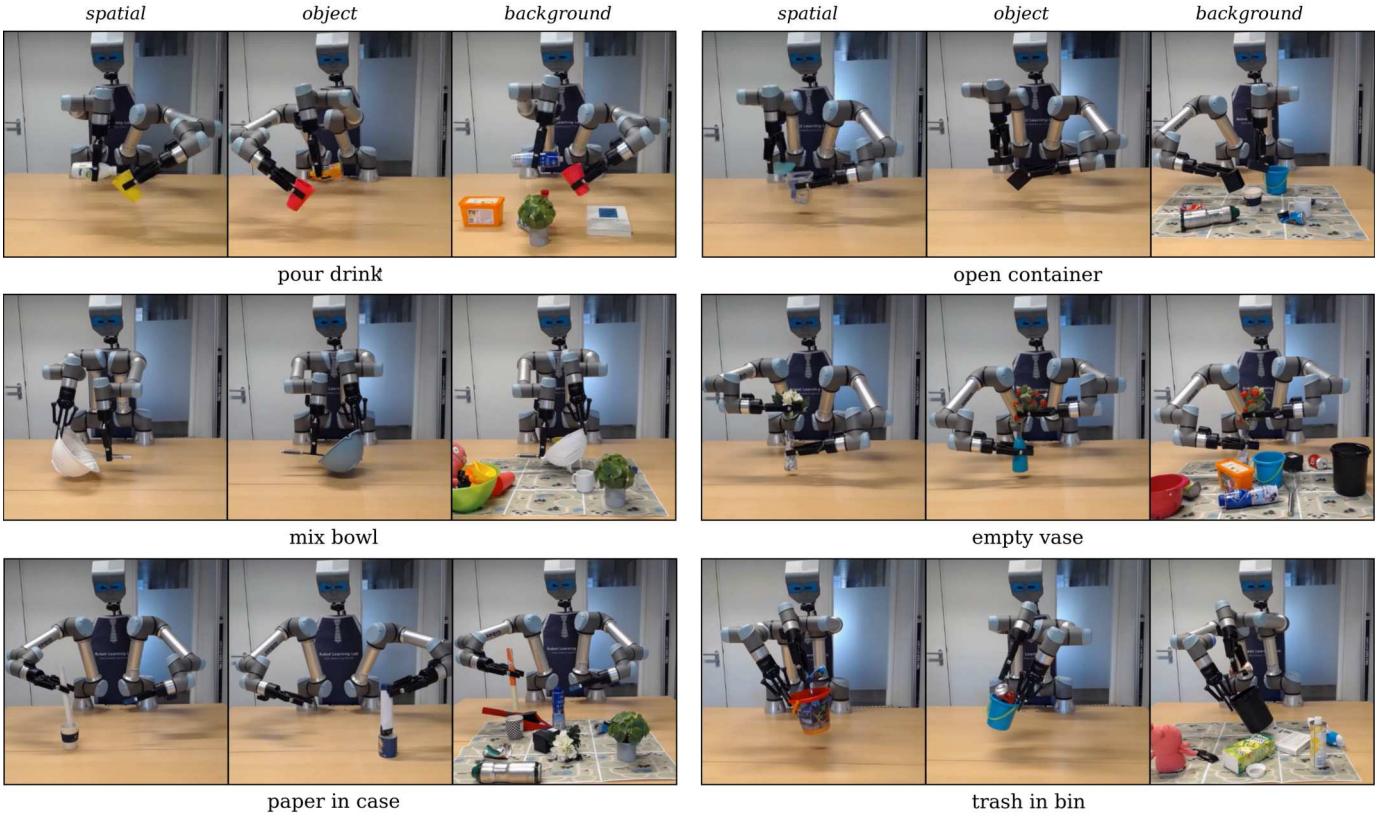


Fig. 10. **Test tasks with a real robot.** We conduct extensive experiments in 6 diverse tasks, containing different object categories, manipulation skills and arm-coordination strategies. For each task, we explicitly test for generalization in novel spatial arrangements, object instances and background scene noise.

Fig. 11. **Generalization experiments in 6 real-world tasks.** We evaluate for unseen *spatial* arrangements, *object* instances and *background* noise. We conduct 12 trials per task-scenario combination and report the total number of successes on top of each bar.

categories. For each task, we explicitly collect object instances of the same category with variations in appearance and shape to perform generalization experiments. An illustration of task trials is shown in Fig. 10.

Generalization We evaluate for all 6 tasks in three generalization splits: (a) *spatial*, containing unseen arrangements of the demo objects, (b) *object*, containing unseen object instances in unseen arrangements, and (c) *background*, containing cluttered scenes with random background objects and both seen/unseen object instances in unseen arrangements. We perform 12 trials per task and split, for a total of 216 trials. Results are shown in Fig. 11.

We observe that our trained policies are robust in spatial arrangements, with an average success rate of 91.6%, showcasing the effectiveness of our spatial augmentations. In novel object instances, the average success rate is 72.2%, with the policy mostly failing in the *open container* and *paper in case* tasks, which is due to considerable change in shape between the demo-test objects. This leads to failures in high-precision tasks such as removing a small container lid or placing the paper in a small case. We visually verify that keypoint correspondences are in all cases (reasonably) sound, so this failure mode comes from the policy's inability to handle diverse shapes, due to the single demonstration used. In the other 4 tasks, similar issues are much less prominent, as the demonstrated behavior suffices to cover the shape distribution of different objects. Overall, we find our results promising with regards to the use of keypoints as underlying state representations. In the background split, the average success rate is 59.7%, with most failures due to infeasible grasps and collisions with neighbouring objects. In some cases we also observed floaters from the point tracker, as an artifact of loss of visibility due to scene clutter and access to only single-view. We note however that the reported performance gap in

TABLE II
TRAJECTORY-LEVEL POLICY SUCCESS RATES IN 3 REAL-WORLD TASKS.

Method	**#Params**	pour_drink	mix_bowl	trash_in_bin
R+X (*gpt-4o*) [53]	-	6/9	3/9	5/9
KALM-diffuser [18]	$56.6M$	7/9	**8/9**	6/9
Kp-RDT (Ours)	$5.7M$	**8/9**	**8/9**	**7/9**

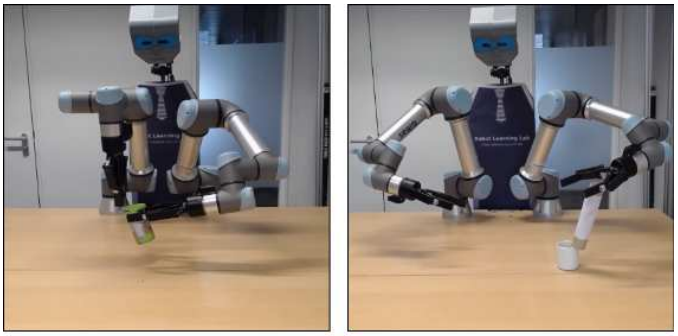


Fig. 12. **Most common failure cases:** In high-precision tasks such as removing a thin container lid or placing a thick paper roll in a small pencil case, the policy tends to fail in unseen object instances when they significantly vary in shape from the demo objects.

this split is expected, considering that the policy has not been distilled from obstacle avoidance-integrated TAMP and relies simply on IK motion planning. Some examples of failure cases are illustrated in Fig. 12.

Comparisons with trajectory-level policies We also investigate the performance of our Kp-RDT policy, when compared to recent works that rely on predicting action trajectories from keypoint-based representations. We use two recent works: (a) **R+X** [53], which uses an LLM with keypoint-action token prompting strategy [52] and in-context examples, and (b) **KALM-diffuser** [18], which trains a keypoint-conditioned variant of Diffuser-Actor [65] for trajectory prediction. Both these works predict the entire action trajectory from the initial keypoint state and execute it open-loop. We use *gpt-4o* [51] for R+X, the UNet implementation from the authors for KALM [18] and use 5000 demo augmentations for all methods. We follow the same setup with Kp-RDT as in KALM and train trajectory-level policies with $H = 96$ and 100 diffusion steps. For R+X, we compute MSE between the observed keypoint coordinates and those in memory (from the augmentations) and retrieve the top-3 most similar for in-context prompting.

We conduct experiments in 3 of our tasks (*pour drink, mix bowl* and *trash in bin*), with 3 trials per task-split, for a total of 27 trials per method, where we set the initial object arrangement as close as possible. Results are given in Tab. II. The zero-shot LLM policy achieves an average success rate of 51.8%, with most failures in the mixing task due to inability to grasp the whisk under certain orientations. These results showcase the necessity for distilling the augmentations with a trained policy, instead of relying on LLM in-context interpolation. Both KALM-diffuser and our method achieve similar results $\sim 80\%$, although our method is an order of magnitude smaller in raw parameter count. This is because unlike KALM, our Kp-RDT architecture does not require SD-DINO feature maps and projectors for them for conditioning the policy, instead only relying on 3D keypoint coordinates and group embeddings to encode semantics.

V. CONCLUSIONS

In this work we present *PAD*, a unified framework for learning bimanual visuomotor policies from a single human video demonstration. Our framework proposes to use keypoints as underlying state representations to parse the video into robot-executable data, augment the data at scale in a simulation-free fashion and distill the augmented data with a diffusion policy. We design a bimanual TAMP procedure for specializing demo augmentations to the bimanual case and propose Kp-RDT, an adapted version of RDT [4] that supports keypoint conditioning. Empirically, we show that our framework is superior to state-of-the-art bimanual demo augmentation methods relying on simulation rollouts, in terms of success rates, data collection time and sample-efficiency. We apply our framework in six real-world tasks and show that PAD obtains policies that generalize to unseen spatial arrangements, object instances and background scene noise, while doing so from a single human demonstration.

Limitations & Future Work As discussed earlier, our policies have a considerable error rate at high-precision tasks for unseen object instances that significantly vary in shape from the demo object. This can be alleviated by using more training videos, for demonstrating how to adapt the policy in cases of varying shapes. Second, our TAMP procedure and final policies currently do not consider obstacle avoidance, which can be integrated in the future via using third-party motion planners and some keypoint representation for obstacles. Third, our augmentation framework currently relies on the 3D rigidity assumption, and hence doesn't support deformable objects. Finally, in its current form our work only supports training single-task policies from separate videos. A future avenue will explore combining keypoint with language-conditioning to obtain multi-task policies that can follow text instructions, while still enjoying the generalization benefits of keypoint abstractions.

REFERENCES

[1] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *ArXiv*, abs/2303.04137, 2023.

[2] Zipeng Fu, Tony Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *ArXiv*, abs/2401.02117, 2024.

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

[4] Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *ArXiv*, abs/2410.07864, 2024.

[5] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *ArXiv*, abs/2406.09246, 2024.

[6] Kristen Grauman and Andrew Westbury et al. Ego4d: Around the world in 3,000 hours of egocentric video. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18973–18990, 2021.

[7] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F. Fouhey. Understanding human hands in contact at internet scale. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9866–9875, 2020.

[8] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. *ArXiv*, abs/1804.02748, 2018.

[9] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter N. Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5843–5851, 2017.

[10] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. Dexycb: A benchmark for capturing hand grasping of objects. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9040–9049, 2021.

[11] Juntao Ren, Priya Sundaresan, Dorsa Sadigh, Sanjiban Choudhury, and Jeannette Bohg. Motion tracks: A unified representation for human-robot transfer in few-shot imitation learning. *ArXiv*, abs/2501.06994, 2025.

[12] Marion Lepert, Jiaying Fang, and Jeannette Bohg. Phantom: Training robots without robots using only human videos. *ArXiv*, abs/2503.00779, 2025.

[13] Siddhant Haldar and Lerrel Pinto. Point policy: Unifying observations and actions with key points for robot manipulation. *ArXiv*, abs/2502.20391, 2025.

[14] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj S. Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning*, 2023.

[15] Ryan Hoque, Ajay Mandlekar, Caelan Reed Garrett, Ken Goldberg, and Dieter Fox. Intervengen: Interventional data generation for robust and data-efficient robot imitation learning. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2840–2846, 2024.

[16] Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. Skillmimicgen: Automated demonstration generation for efficient skill learning and deployment. In *Conference on Robot Learning*, 2024.

[17] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. *ArXiv*, abs/2410.24185, 2024.

[18] Xiaolin Fang, Bo-Ruei Huang, Jiayuan Mao, Jasmine Shone, Joshua B. Tenenbaum, Tom'as Lozano-P'erez, and Leslie Pack Kaelbling. Keypoint abstraction using large models for object-relative imitation learning. *ArXiv*, abs/2410.23254, 2024.

[19] Mara Levy, Siddhant Haldar, Lerrel Pinto, and Abhinav Shirivastava. P3-po: Prescriptive point priors for visuo-spatial generalization of robot policies. *ArXiv*, abs/2412.06784, 2024.

[20] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CoRR*, abs/2104.14294, 2021.

[21] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *ArXiv*, abs/2306.03881, 2023.

[22] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. *ArXiv*, abs/2305.15347, 2023.

[23] Zhengrong Xue, Shuying Deng, Zhenyang Chen, Yixuan Wang, Zhecheng Yuan, and Huazhe Xu. Demogen: Synthetic demonstration generation for data-efficient visuomotor policy learning. *ArXiv*, abs/2502.16932, 2025.

[24] Yuke Zhu, Josiah Wong, Ajay Mandlekar, and Roberto Mart'in-Mart'in. robosuite: A modular simulation framework and benchmark for robot learning. *ArXiv*, abs/2009.12293, 2020.

[25] Chethan Bhateja, Derek Guo, Dibya Ghosh, Anikait Singh, Manan Tomar, Quan Vuong, Yevgen Chebotar, Sergey Levine, and Aviral Kumar. Robotic offline rl from internet videos via value-function learning. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16977–16984, 2024.

[26] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhi Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, 2022.

[27] Hongtao Wu, Ya Jing, Chi-Hou Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *ArXiv*, abs/2312.13139, 2023.

[28] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *ArXiv*, abs/2210.00030, 2022.

[29] Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, 2023.

[30] Siddharth Karamcheti, Suraj Nair, Annie S. Chen, Thomas Kollar, Chelsea Finn, Dorsa Sadigh, and Percy Liang. Language-driven representation learning for robotics. *ArXiv*, abs/2302.12766, 2023.

[31] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables diverse zero-shot robot manipulation. *ArXiv*, abs/2405.01527, 2024.

[32] Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. *ArXiv*, abs/2409.16283, 2024.

[33] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. Egomimic: Scaling imitation learning via egocentric video. *ArXiv*, abs/2410.24221, 2024.

[34] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, 2021.

[35] Vincent Liu, Ademi Adeniji, Haotian Zhan, Siddhant Haldar, Raunaq Bhirangi, Pieter Abbeel, and Lerrel Pinto. Egozero: Robot learning from smart glasses, 2025.

[36] Zihang Lai, Senthil Purushwalkam, and Abhinav Kumar Gupta. The functional correspondence problem. *ArXiv*, 2021.

[37] Diego Rodriguez and Sven Behnke. Transferring category-based functional grasping skills by latent space non-rigid registration. *IEEE Robotics and Automation Letters*, 3:2662–2669, 2018.

[38] Ondrej Biza, Skye Thompson, Kishore Reddy Pagidi, Abhinav Kumar, Elise van der Pol, Robin Walters, Thomas Kipf, Jan-Willem van de Meent, Lawson L. S. Wong, and Robert W. Platt. One-shot imitation learning via interaction warping. In *Conference on Robot Learning*, 2023.

[39] Lucas Manuelli, Wei Gao, Peter R. Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *International Symposium of Robotics Research*, 2019.

[40] Wei Gao and Russ Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letters*, 6:2962–2969, 2021.

[41] Dylan Turpin, Liquang Wang, Stavros Tsogkas, Sven J. Dickinson, and Animesh Garg. Gift: Generalizable interaction-aware functional tool affordances without labels. *ArXiv*, abs/2106.14973, 2021.

[42] Bowen Wen, Wenzhao Lian, Kostas E. Bekris, and Stefan Schaal. You only demonstrate once: Category-level manipulation from single visual demonstration. *ArXiv*, abs/2201.12716, 2022.

[43] Anthony Simeonov, Yilun Du, Andrea Tagliasacchi, Joshua B. Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400, 2021.

[44] Norman Di Palo and Edward Johns. Dinobot: Robot manipulation via retrieval and alignment with vision foundation models. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2798–2805, 2024.

[45] Yuanchen Ju, Kaizhe Hu, Guowei Zhang, Gu Zhang, Mingrun Jiang, and Huazhe Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation. In *European Conference on Computer Vision*, 2024.

[46] Qianxu Wang, Haotong Zhang, Congyue Deng, Yang You, Hao Dong, Yixin Zhu, and Leonidas J. Guibas. Sparsedff: Sparse-view feature distillation for one-shot dexterous manipulation. *ArXiv*, abs/2310.16838, 2023.

[47] Yixuan Wang, Zhuoran Li, Mingtong Zhang, Katherine Rose Driggs-Campbell, Jiajun Wu, Fei-Fei Li, and Yunzhu Li. D3fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *ArXiv*, abs/2309.16118, 2023.

[48] Walter Goodwin, Ioannis Havoutis, and Ingmar Posner. You only look at one: Category-level object representations for pose estimation from a single example. *ArXiv*, abs/2305.12626, 2023.

[49] Denis Hadjivelichkov, Sicelukwanda Zwane, Marc Peter Deisenroth, Lourdes de Agapito, and D. Kanoulas. One-shot transfer of affordance regions? affcorrs! *ArXiv*, abs/2209.07147, 2022.

[50] Machel Reid, Nikolay Savinov, and Denis Teplyashin et. al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv*, abs/2403.05530, 2024.

[51] OpenAI. Gpt-4 technical report. 2023.

[52] Norman Di Palo and Edward Johns. Keypoint action tokens enable in-context imitation learning in robotics. *ArXiv*, abs/2403.19578, 2024.

[53] Georgios Papagiannis, Norman Di Palo, Pietro Vitiello, and Edward Johns. R+x: Retrieval and execution from everyday human videos. *ArXiv*, abs/2407.12957, 2024.

[54] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *Proc. ECCV*, 2024.

[55] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models. *ArXiv*, abs/2310.01361, 2023.

[56] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *ArXiv*, abs/2311.01455, 2023.

[57] Pu Hua, Minghuan Liu, Annabella Macaluso, Yunfeng Lin, Weinan Zhang, Huazhe Xu, and Lirui Wang. Gensim2: Scaling robot data generation with multi-modal and reasoning llms. In *Conference on Robot Learning*, 2024.

[58] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *CVPR*, 2024.

[59] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), November 2017.

[60] Joan Solà, Jérémie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *ArXiv*, abs/1812.01537, 2018.

[61] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Fei-Fei Li. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *ArXiv*, abs/2409.01652, 2024.

[62] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[63] William S. Peebles and Saining Xie. Scalable diffusion models with transformers. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182, 2022.

[64] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *ArXiv*, abs/2102.09672, 2021.

[65] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *ArXiv*, abs/2402.10885, 2024.