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

PEEK: Guiding and Minimal Image Representations for Zero-Shot Generalization of Robot Manipulation Policies

Jesse Zhang*[1,2,3], Marius Memmel*[1,2], Kevin Kim[3],
Dieter Fox[1,4], Jesse Thomason[3], Fabio Ramos[2], Erdem Bıyık[3], Abhishek Gupta†[1], Anqi Li†[2]

***Abstract*— Robotic manipulation policies often fail to generalize because they must simultaneously learn *where* to attend, *what* actions to take, and *how* to execute them. We argue that high-level reasoning about *where* and *what* can be offloaded to vision-language models (VLMs), leaving policies to specialize in *how* to act. We present PEEK (Policy-agnostic Extraction of Essential Keypoints), which fine-tunes VLMs to predict a unified point-based intermediate representation: (1) end-effector paths specifying *what* actions to take, and (2) task-relevant masks indicating *where* to focus. These annotations are directly overlaid onto robot observations, making the representation policy-agnostic and transferable across architectures. To enable scalable training, we introduce an automatic annotation pipeline, generating labeled data across 20+ robot datasets spanning 9 embodiments. In real-world evaluations, PEEK consistently boosts zero-shot generalization, including a 41.4× real-world improvement for a 3D policy trained only in simulation, and 2–3.5× gains for both large VLAs and small manipulation policies. By letting VLMs absorb semantic and visual complexity, PEEK equips manipulation policies with the minimal cues they need—*where*, *what*, and *how*. Website at https://peek-robot.github.io.**

I. Introduction

Imagine walking through a crowded store when your child suddenly cries out, "I want the Labubu!" Though you've never heard the word before, context clues guide your eyes to the fuzzy toy on the shelf, and you effortlessly weave through the crowd to grab it. What makes this possible is not raw perception ability, but the ability to interpret ambiguous instructions and distill them into just the right cues—*where* to focus, *what* actions to take, and *how* to perform these actions at the low level. Similarly, if given *where* to focus and *what* motions to take, a robot manipulation policy should be able to achieve the visual robustness and semantic generalization necessary for open-world deployment by focusing only on *how* to perform actions.

A common tactic for training manipulation policies is through imitation learning of human-collected robotics data [1]–[4], which attempts to learn the where, what, and how all at the same time. Yet their performance degrades on novel objects, clutter, or semantic variations [5], [6], since the policy alone bears the burden of handling task, semantic, and visual complexity. Such failures often entangle the axes of *where*, *what*, and *how*—for example, grasping a distractor simultaneously reflects misplaced attention, an incorrect object choice, and a wrong motion.

*Co-first authors, †Equal Advising, [1]University of Washington, [2]NVIDIA, [3]University of Southern California, [4]Allen Institute for AI

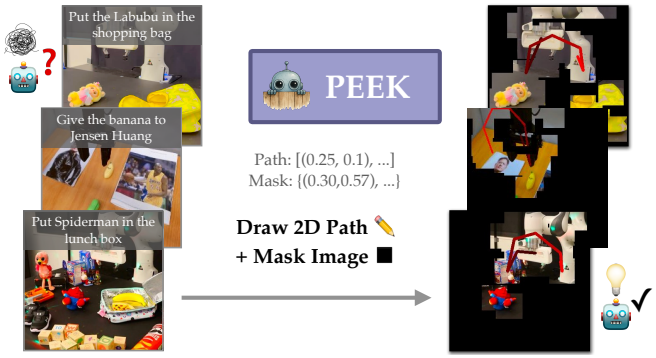


Fig. 1: PEEK enables policy generalization by modulating minimal representations of *where* to focus and *what* to do for robust policy learning.

Our key idea is to offload high-level reasoning to vision-language models (VLMs), which can excel at semantic and visual generalization [7], [8], leaving the policy to determine how low-level behavior should be executed. Instead of forcing the policy to directly parse raw images and instructions, a high-level VLM modulates the input representation to the low-level policy by providing: (1) a path that encodes *what* the policy should do, and (2) masks showing *where* to attend. By "absorbing" semantic and visual variation, the VLM provides the policy a simplified, annotated "peek" of the scene that gives the what and the where, while the policy only needs to learn *how* to perform the low-level actions. This intermediate representation helps policy execution inherit many of the VLM's semantic and visual generalization capabilities. Our VLM-modulated representation is naturally policy-agnostic, allowing it to be applied to arbitrary image-input robot manipulation policies, including state-of-the-art RGB and 3D manipulation policies [1], [3], [9].

To concretely instantiate this insight into a practical algorithm, we introduce PEEK (**P**olicy-agnostic **E**xtraction of **E**ssential **K**eypoints), which proposes a unified, point-based intermediate representation that trains VLMs to predict *what* policies should do and *where* to focus on. Specifically, we propose to finetune pretrained VLMs [10] to predict a sequence of points corresponding to (1) a *path* that guides the robot end-effector in what actions to take and (2) a set of task-relevant *masking points* that show the policy where to focus on (see Figure 1). During low-level visuomotor policy training and inference, we modulate the policy's image observations by directly drawing these VLM-predicted paths and masks onto the image, allowing the policy to simply focus on how to act, rather than learning all

three simultaneously. Doing so significantly bolsters policy generalization, combining the generality of high-level VLM predictions with the precision of low-level policy learning. In this paper, we instantiate a full-stack implementation of PEEK, from devising a scalable data annotation scheme that enables large-scale VLM finetuning on robotic datasets to representation-modulated training of low-level robot policies from simulation and real world data.

In 535 real-world evaluations across 17 task variations, we demonstrate that PEEK consistently boosts zero-shot policy generalization: a 3D policy (3DDA [9]) trained only in simulation achieves **41.4**× higher success in the real world when guided by PEEK, and both large-scale vision-language-action models (π_0 [3]) and small transformer-based policies [1] see **2–3.5**× success rate improvements. These results demonstrate the power of using high-level VLMs to absorb task complexity, providing low-level policies with exactly the minimal cues they need for generalizable manipulation.

II. Related Works

Object-Centric Representations. One approach to improving the visual generalization of imitation learning (IL) policies is to build object-centric representations [11]–[17]. Earlier works relied on human-selected abstractions or manual annotation [11], while more recent methods leverage pre-trained, open-vocabulary segmentation models to visually isolate task-relevant objects [12]–[15], [17]. Among these, ARRO [13] is closest to our work, proposing a policy-agnostic masking scheme using GroundingDINO [18] to filter images for task-relevant objects. However, we found in Section IV that such object detectors often fail in cluttered, realistic scenes. By contrast, PEEK queries a fine-tuned VLM to predict task-relevant masking points directly, resulting in more robust masks than using object-detection models due to the VLM's extensive pre-training. Another approach, OTTER [16], implements *implicit* masking by filtering CLIP image patches, but this approach is architecture specifc. PEEK's policy-agnostic explicit masking allows us to integrate it with far more powerful policy backbones than OTTER, i.e., vision-language-action models like π_0 [3]. Finally, while masking alone helps mitigate visual distractors, it alone cannot handle semantic variation; PEEK also provides explicit action guidance via predicted paths.

Another line of object-centric methods relies on *learning* to decompose scenes into object-level representations in a self-supervised manner, e.g., via slot-attention [19]–[22], which learns to map visual features into a set of discrete, object-centric "slots" through competitive attention mechanisms. However, these methods have not been applied to real-world robot manipulation settings and generally do not work zero-shot. PEEK's use of a pre-trained VLM helps it predict task-relevant points on new objects and tasks.

Guiding Manipulation Policies. A separate line of work improves generalization by explicitly guiding policies in *how* to perform tasks via 2D gripper paths. RT-Trajectory introduced this concept using human-drawn sketches at inference time [23]. Later methods integrated 2D path prediction into VLA training objectives [8], [24]–[26], but these approaches are tied to specific architectures. More relevant is HAMSTER [7], which trains a VLM to predict future 2D gripper paths that a lower-level 3D policy conditions on. While this approach aids with policy understanding of *what* high-level motions to perform, we found in Section IV that HAMSTER-trained policies are easily confused by visual variation. In contrast, PEEK's VLM predicts a single point-based representation that also includes masks helping the policy understand *where* to focus on.

Other works propose guiding policies via relabeling language instructions [27]–[31] or behavioral priors, i.e., latent *skills*, learned from data [32]–[34]. These approaches are complementary to PEEK's image-based input representation.

III. PEEK: Guiding and Minimal Image Representations

We study how to enhance the generalization capability of arbitrary visuomotor policies to semantic and visual task variation. To do so, PEEK proposes to offload high-level task reasoning to VLMs to produce a *guiding* (what) and *minimal* (where) image representation for a low-level policy, which in turn actualizes *how* to actually perform the task through real-world actions. Concretely, we instantiate this representation via 1) 2D gripper paths and 2) task-relevant masks (see Figure 1). This hierarchical approach shifts the burden of generalization from the low-level policy to the high-level VLM, allowing the policy to focus only on *how* to execute low-level actions.

A. Conceptual Insight

Imitation learning methods train a policy $\pi(a_t \mid o_t, s_t, l)$ predicting an action a_t given RGB observations o_t, proprioceptive and other sensory data (e.g., depth) s_t, and a task instruction l. Given an expert-collected robot dataset $\mathcal{D}_\pi$, π is trained with maximium likelihood estimation, i.e., $\max_\pi \mathbb{E}_{(o_t,s_t,l,a_t)\sim\mathcal{D}_\pi}[\log \pi(a_t \mid o_t, s_t, l)]$.

PEEK explores how to improve imitation learning methods by training a VLM to map (l, o_t) to a *guiding* but *minimal* representation, $o_t^{p,m}$, that enables zero-shot generalization to significant visual and semantic variation beyond that in $\mathcal{D}_\pi$. Downstream task variation can include any combination of, e.g., new scenes, visual clutter not present during training, new objects, and unseen language instructions.

Formally, PEEK fine-tunes a pre-trained VLM conditioned on (l, o_t) to produce a set of points, i.e., $p_t, m_t \sim \text{VLM}(\cdot \mid o_t, l)$, corresponding to: (1) 2D gripper paths, p_t, indicating where the end-effector should move to solve the task, and (2) a set of task-relevant masking points, m_t, that indicate objects and regions of relevance. 2D gripper paths are defined as $p_t = [(x, y)_t, ..., (x, y)_T]$ where $(x, y) \in [0, 1]^2$ are normalized pixel locations of the end effector's positions at timestep t until trajectory end point T. Masking points are defined as $m_t = \{(x, y)_i\}_{i=1}^M$, an unordered set of pixel locations $(x, y) \in [0, 1]^2$ of task-relevant points.

Although any pre-trained text and image input VLM can be used to predict these path and mask points, prior work

has found that even the best closed-source models struggle with predicting robot gripper paths without fine-tuning [7], [8], let alone masking points. Therefore, we need to *fine-tune* a VLM on a large dataset that grounds it to a diverse set of robot scenes and embodiments. PEEK introduces a scalable data-labeling scheme which we use to create a dataset of over 2M VQA pairs, spanning 148k trajectories, 9 embodiments, and 21 robotics datasets.

B. VLM Data Preparation

To finetune VLMs for PEEK we assemble a dataset $\mathcal{D}_{\text{VLM}} = \{(o, l, \texttt{ans})_i\}_{i=1}^{V}$ of image inputs o, instructions l, and text-based responses `ans` depending on the dataset. In this section, we introduce our datasets, and then we detail our automatic robot data labeling pipeline.

Point Prediction and VQA Datasets. Like prior work [7], [8], we first incorporate readily available pixel point prediction and visual question answering (VQA) data into $\mathcal{D}_{\text{VLM}}$ to maintain the VLM's general world knowledge and object reasoning capabilities. We use the RoboPoint dataset [35] with 770k pixel point prediction tasks, e.g., l = "Point to the cushions," and `ans` = $[(0.56, 0.69), (0.43, 0.67)]$, and 665k VQA examples, e.g., l = "What is the cat eating?," and `ans` = "An apple."

Robotics Datasets. Our main robotics dataset comes from the Open X-Embodiment (OXE) dataset [36], where we label 20 datasets from the OXE "magic soup" [2]. Notably, our data labeling pipeline works effectively on datasets with lots of clutter or awkward viewpoints that make task-relevant objects appear very small, such as DROID [37] (e.g., the pen in Figure 6). In contrast, we found the pre-trained object detection models [18] used by prior works to extract object-centric representations [13], [17] to be ineffective. Finally, we also include a robotics simulation dataset (LIBERO-90 [38]) in our training mix to broaden the visual feature coverage of the VLM. Now we describe how we scalably label our robotics datasets.

Automatically Labeling Robotic Datasets. PEEK's VLM needs to predict a list of 2D gripper path points p_t and task-relevant masking points m_t given arbitrary task instructions and robot observations. Prior works label their dataset using calibrated 3D cameras (in simulation and the real world) or human annotations [7], [8], limiting the scalability of data annotation. In contrast, we devise an automatic and scalable multi-step tracking pipeline that extracts how to solve the task and what to focus on directly from robot videos.

First, our representation should be *minimal*, i.e., it should encode task-relevant entities at each timestep t. To extract this information from a video, we have to ask the following question: *What entities are relevant to the task?* We answer this question by tracking a grid of points through time with a visual point tracking model [39]. Points that move significantly throughout the trajectory correspond to the robot arm or objects being manipulated. We define this set as *task-relevant* points $P_t^{task} = \{(x, y)_i\}_{i=1}^{N}$, tracked across all timesteps of a trajectory $t \in [1, T]$, as they capture the minimal information needed by a policy to solve the task.

Second, our representation should be *guiding*, i.e., capture information about the (1) future relevant object movement and (2) robot gripper movement. (1) The tracking points tell us the entities' position at each timestep t. To capture how they move and where they end up, e.g., object placement locations, we include points at the last timestep P_T^{task}. (2) We additionally construct a set of *end-effector* points $P_t^{grip} = [(x, y)]_t^T$ by tracking the gripper throughout the video.

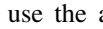
Finally, we process the data into subtrajectories separated by when the robot manipulates an object, and construct the 2D paths $p_t = P_t^{grip}$ and masking points $m_t = P_t^{task} \cup P_T^{task}$. The natural language prediction target `ans` for the VLM is then a combination of the shortened p_t and m_t: `TRAJECTORY: [(0.25, 0.1), ...] MASK: [(0.30,0.57), ...]`. See Section I-A and Figure 6 for details regarding the data labeling pipeline.

C. VLM and Policy Training/Inference with PEEK

VLM Fine-tuning. We use `VILA-1.5-3b` [10] as our base VLM, a 3B parameter VLM trained on interleaved image-text datasets and video captioning data. We fine-tune our VLM for one epoch using the combined datasets totalling 3.5M samples with a learning rate of $5e^{-2}$ and a batch size of 16. Fine-tuning takes $\sim 20h$ on 8 NVIDIA A100 GPUs. We fine-tune the VLM with a standard supervised prediction objective to maximize the log-likelihood of the answers `ans`: $\max_{\text{VLM}} \mathbb{E}_{(o,l,\text{ans})\sim\mathcal{D}_{\text{VLM}}} \log \text{VLM}(\texttt{ans} \mid o, l)$.

VLM Inference. During deployment, PEEK's VLM acts at a higher level, absorbing the semantic complexity and visual clutter of the scene and providing a lower-level policy with a guiding and minimal representation. However, querying the high-level VLM at every timestep is unnecessary because the scene is unlikely to change significantly at the same frequency as the policy is acting. Since frequent VLM queries are expensive and must be run sequentially, we run the VLM at a reduced frequency. While prior works predict paths either at the start of a rollout [7] or at every timestep [24], our hybrid approach strikes a balance between inference speed and responsiveness. To minimize the gap between training and deployment, our data labeling and training scheme reflects this design choice by querying the VLM at a fixed frequency of every H timesteps.

VLM / Policy Interface. During inference, the policy receives an augmented image input $o_t^{p,m}$ created by drawing the path p_t and mask m_t onto the image observation o_t.

We *draw* the 2D path p_t by connecting each subsequent point in p_t with a colored line segment. This drawing guides the policy for which path to follow to accomplish the task. To indicate passage of time, the line segment changes from dark to light red  . To create masks, we start from a black canvas and use the area around the predicted task-relevant points to reveal parts of the image. For each predicted task-relevant point $(x, y) \in m$, we create a square centered around (x, y) with edge length 8% of the image's size. See Figure 2 for a visual depiction of path and mask drawing.

We query the VLM every H steps to generate p_t, m_t based on the current environment observation o_t and apply the same

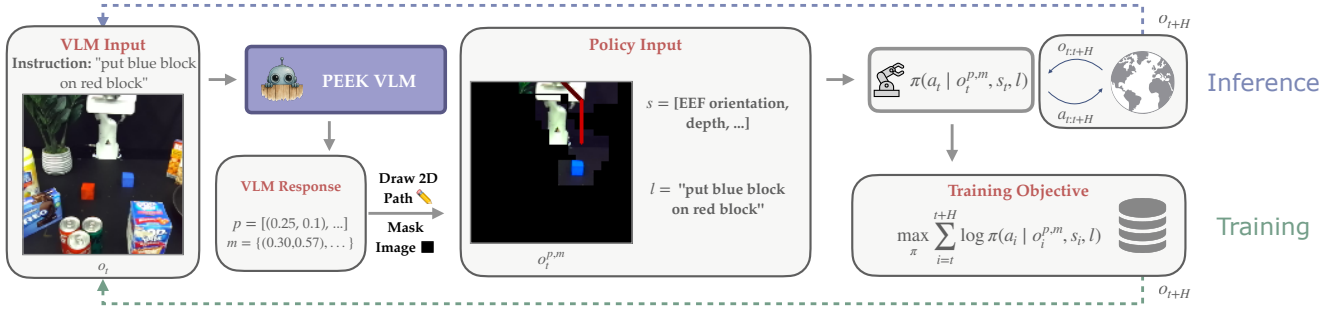


Fig. 2: **Policy Training and Inference Pipeline.** The VLM is called every H steps to generate a path and task-relevant points. An (arbitrary) RGB-input policy is conditioned on the path and masked image to either predict actions for inference or for training. The same path and mask is applied onto incoming observations for H steps, after which the VLM is re-queried.

annotations p_t and m_t to all incoming observations $o^{p,m}_{t:t+H}$ until H steps have passed.

Each VLM query takes about 4-6 seconds on an RTX 3090 without any explicit speed optimization, but until the next VLM query, the policy π runs at its own inference speed.

Policy Training. Consequently, we annotate all the trajectories in the policy training data $\mathcal{D}_\pi$ to create an annotated dataset $\mathcal{D}^{p,m}_\pi$. We train π on the PEEK-labeled dataset $\mathcal{D}^{p,m}_\pi$ using its original training objective, e.g., maximizing log-likelihood of the actions: $\max_\pi \mathbb{E}_{\mathcal{D}^{p,m}_\pi} \log \pi(a_t \mid o^{p,m}_t, s_t, l)$. We list policy and VLM query frequencies in Section I-B.

IV. Experimental Setup

To demonstrate the broad applicability of PEEK, we evaluate across two real-world robot embodiments, both 2D (π_0 [3], `ACT` [1]) and 3D (`3DDA` [9]) policy classes, fine-tuning and training policies from scratch. We evaluate zero-shot generalization from publicly available [40] and simulation-generated datasets to our custom setups, varying the task semantics and introducing visual clutter.

Franka Sim-to-Real. To study the semantic generalization and visual robustness induced by PEEK, we require a large-scale robotic dataset to cover all possible motions the policy might encounter during inference. Simulation offers a cheap, scalable approach to generate such a dataset without going through the effort of manual data collection.

We collect 2.5k trajectories of *cube stacking* with a motion planner in MuJoCo environments with three colored cubes (sampled from {red, green, blue, yellow}) placed randomly on a 40×40cm grid. See Figure 3 for a visualization of the data. Our real-world setup consists of a Franka Emika Panda robot [41], [42] with depth from processing RGB images from a Zed 2 stereo camera with FoundationStereo [43].

In the real world, we first test policy transfer on four fixed cube configurations (Basic), then add visual Clutter to assess visual robustness, and finally evaluate three Semantic tasks requiring reasoning about unseen objects and placements (Figure 3). Each policy is evaluated for 5 trials per task, totaling 220 evaluations across 4 methods and 11 variations.

WidowX BRIDGE. Our second environment uses a WidowX250 robot with a single Logitech C920 RGB camera, resembling the BRIDGE [40] environment, albeit without exactly reproducing camera angles and with a different table, objects, and background wall. We re-label the BRIDGE-v2 dataset [40] (single camera angle) with PEEK according to Section III-C and zero-shot evaluate it on our setup.

We evaluate on a set of three tasks, representing basic generalization (to our custom robot setup), visualized in the Basic column of Figure 4. We then evaluate Clutter, which adds significant visual clutter to each of the three Basic tasks, and finally Semantic, representing difficult tasks that require visual-language reasoning to complete. We perform 5 evals per task with randomized object locations.

Baselines. In our Sim-to-Real experiments, we evaluate PEEK's application to 3D policies. We use `3DDA` [9] as our base policy and implement all baselines on top of it.

- `3DDA` [9]: A state-of-the-art language-conditioned 3D policy conditioned on depth, RGB, and language.
- `HAMSTER` [7]: Fine-tunes a 13B parameter VLM to predict *2D gripper paths* for a 3D policy to condition on.
- `ARRO` [13]: An *explicit masking* baseline using GroundingDINO [18] to segment gripper and objects.

We apply masks from `ARRO` and `PEEK` to both the RGB image and point clouds input to `3DDA`.

To show PEEK also applies to 2D policies of different architectures, we evaluate it on `ACT` [1] and π_0 [3].

- `ACT` [1]: a small 90M parameter transformer policy we additionally condition with language embeddings [44].
- π_0 [3]: A 3.5B parameter VLA first pre-trained on a large dataset, which we LoRA fine-tune on BRIDGE.
- `OTTER` [16]: A 400M parameter transformer which *implicitly masks* observations by discarding image patches with low CLIP-feature alignment to the task instruction.
- `ARRO` [13]: *Explicit masking* baseline introduced above.

We evaluate both `ARRO` and `PEEK` on top of both `ACT` and π_0 as they are both policy-agnostic.

V. Experimental Results

Our evaluation aims to address the following questions: (Q1) How much does PEEK improve semantic and visual generalization across **diverse** policy architectures? (Q2) How accurately does PEEK help with *where* and *what*? and (Q3) How much does each component of PEEK contribute? We answer these questions in order below.

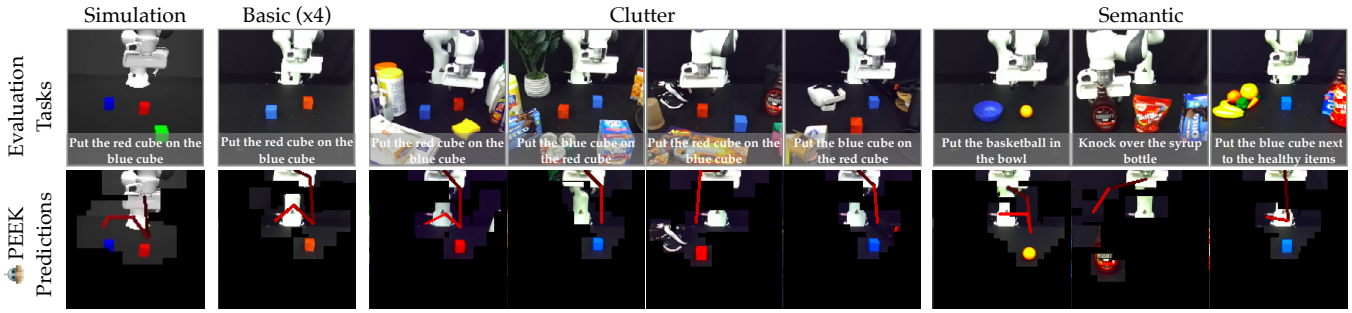


Fig. 3: **Franka Sim-to-Real Tasks.** Zero-shot evaluation environments along with associated path-drawn and masked images produced by PEEK. SIMULATION denotes the generated simulation data that the policies were trained on.

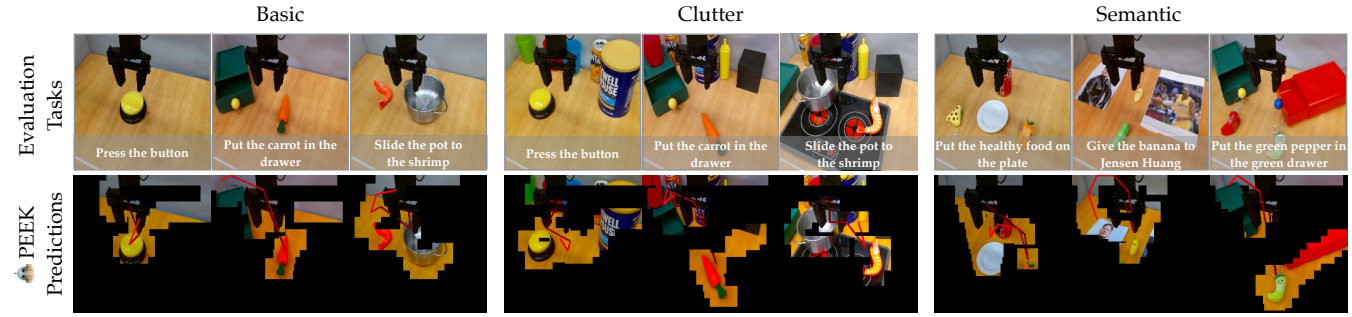


Fig. 4: **WidowX Tasks.** Evaluation environments along with associated path-drawn and masked images produced by PEEK.

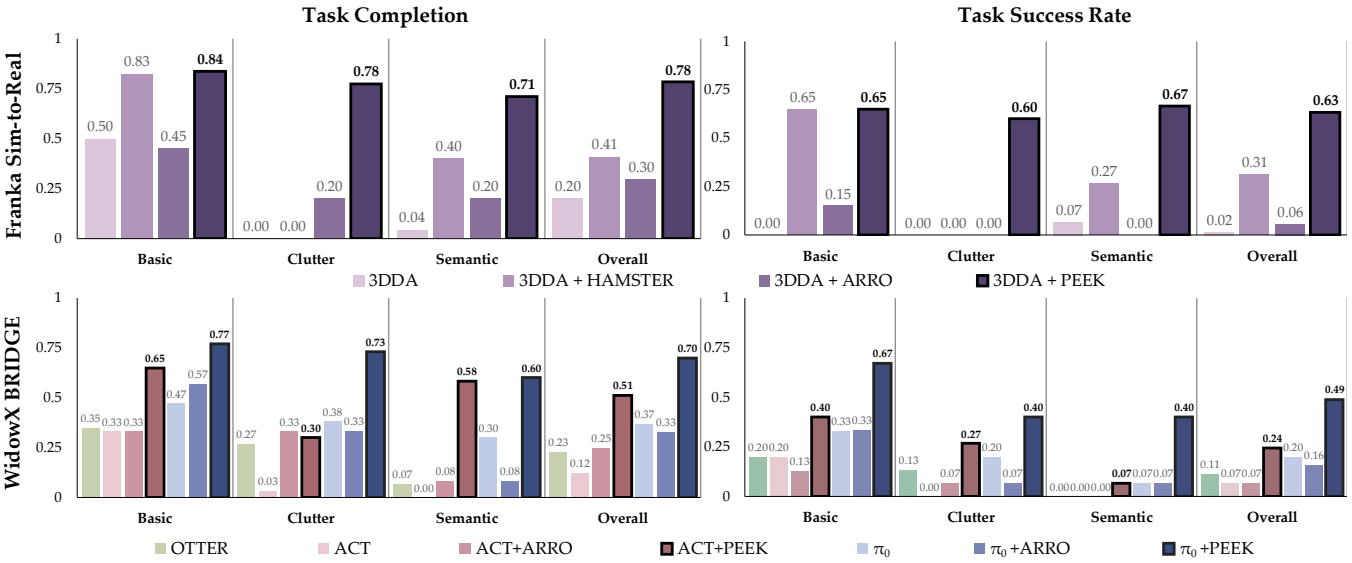


Fig. 5: **Real-World Zero-Shot Generalization Results.** Task completion rates (including partial credit for grasping or reaching objects correctly) and task success rates across 3 task variants: BASIC, CLUTTER, and SEMANTIC in our Franka Sim-to-Real experiments (top) and WidowX BRIDGE experiments (bottom). Results are averaged over all trials and tasks within each variant. PEEK results are bolded for visibility.

A. Q1: Real-World Zero-Shot Generalization Experiments

Franka Sim-to-Real. We plot results in Figure 5 (top). Overall, 3DDA+PEEK improves vanilla 3DDA by **41.4**× and outperforms the best baseline, 3DDA+HAMSTER, by **2**× in overall success rates. While HAMSTER shows some semantic generalization via drawing paths (40% partial success on SEMANTIC), it fails catastrophically when distractor objects are present in the scene (0% partial success on CLUTTER). Instead, PEEK's ability to also mask out irrelevant parts of the image usually completely hides task-irrelevant objects, allowing the policy to solve the task more often by only focusing on low-level control.

Meanwhile, ARRO, which masks-in the robot end-effector and task-relevant objects with pre-trained object detection models, often also includes task-irrelevant objects, confusing the 3DDA policy. PEEK's VLM generalizes better to new objects and its paths help guide the policy even in cases where parts of irrelevant objects are included in the observation. The baseline results demonstrate that answering only one

of *where* to focus or *what* to do is not enough to achieve semantic generalization and visual robustness. We visualize example PEEK VLM predictions in Figure 3.

WidowX BRIDGE. Next, we plot the WidowX results in Figure 5 (bottom). ACT+PEEK and π_0+PEEK outperform their base models by **3.4**× and **2.5**× in overall success rates. ARRO does not improve overall success rates of either base model as its pre-trained object detection module often fails to identify correct objects in clutter, and almost always fails to detect the robot gripper. PEEK's use of a VLM allows it to consistently mask-in the correct object and draw paths accurately starting from the gripper. The VLM predictions visualized in Figure 4 show how PEEK's VLM provides effective paths and masks even in the face of distractors and tasks that require semantic and visual reasoning.

Meanwhile, OTTER performs poorly—better than ACT but worse than standard π_0, and far worse than either PEEK variation—π_0+PEEK overall achieves a **4.5**× better success rate. This result highlights the importance of a policy-agnostic approach, such as PEEK, that can provide explicit path and mask guidance even to already strong base policies.

B. Q2: Does PEEK answer the where and what?

Comparing the first columns of Franka Sim-to-Real (Figure 3) in SIMULATION, BASIC, and CLUTTER, the benefits of paths and masking become apparent: masks remove distractors from the image—showing where to attend to—and the paths guide the policy to pick up the object—showing what to do. Similar findings hold for the WidowX (Figure 4).

While the masks tell the policy what to focus on, they alone are insufficient for solving semantic variation. Take, for example, the SEMANTIC tasks; the policies' training data does not contain demonstrations featuring celebrities ("Give the banana to Jensen Huang" in Figure 4) or various kinds of sweet treats ("Knock over the syrup bottle", "Put the blue cube next to the healthy items" in Figure 3). By letting the high-level VLM absorb the semantic generalization—proposing guiding paths—the policy can simply actualize the path into low-level actions to solve the task.

C. Q3: How does each component contribute?

Ablating Paths and Masks. We ablate the contributions of paths p and masks m on the performance of a language-conditioned 3D policy (3DDA) on the simulated cube stacking task in Table I. While the language-conditioned base policy can stack cubes, it often ignores instruction order, e.g., placing the blue cube on the red instead of the reverse. Adding only paths or only masks improves performance by $+19.3\%$ and $+32.1\%$, respectively. Masks outperform paths since they simplify the scene by removing the distractor cube, while paths alone leave ambiguity. Yet both remain limited: cube stacking highlights the insufficiency of purely predictive or minimal representations. Combining paths and masks, PEEK achieves gains of $+7.9\%$ over paths, $+20.8\%$ over masks, and $+40.1\%$ over the base policy.

Paths p	Masks m	Success (%)
✗	✗	33.5 ± 3.1
✓	✗	52.8 ± 2.9
✗	✓	65.6 ± 3.1
✓	✓	73.6 ± 3.9

TABLE I: Ablation of paths and masks on success rate.

VLM Design Choices. To study VLM design choices, we evaluate on 1k holdout samples from BRIDGE-v2 [40], using Dynamic Time Warping (DTW) for paths [45] and Intersection over Union (IoU) for masks. Reducing the base model from 13B to 3B yields no loss in accuracy (both have DTW 0.12, IoU 0.68) while enabling faster closed-loop inference. Adding RoboPoint slightly improves these metrics and preserves semantic reasoning ability [7]. Finally, joint prediction of paths and masks improves performance, giving a $+19.3\%$ relative gain over a mask-only model (IoU 0.57) without degrading path accuracy (DTW 0.12).

Overall, we see that both paths and masks are essential to PEEK's ability to enhance *policy* generalization, and our choice of a small VLM model that jointly predicts a unified path and mask representation great performance without sacrificing inference speed.

VI. CONCLUSION AND LIMITATIONS

We presented **PEEK (Policy-agnostic Extraction of Essential Keypoints)**, a framework that leverages VLMs to offload high-level reasoning in robot manipulation. By predicting point-based intermediate representations—paths that specify *what* to do and masks that indicate *where* to attend—PEEK provides policies with simplified, annotated observations, allowing them to focus on *how* to act. Real-world evaluations demonstrate substantial improvements in zero-shot generalization across various policies.

However, PEEK still inherits the biases and limitations of the underlying VLMs, which may fail in out-of-distribution scenarios or produce incorrect annotations. Our current representation is also limited to 2D point paths and masks; extending it to richer 3D or multimodal cues is an exciting direction. Moreover, although our annotation pipeline scales across existing robotics datasets, future work could explore how to bootstrap from a much broader corpus of video data.

ACKNOWLEDGEMENTS

We thank Abrar Anwar for helping create the PEEK logo, Helen Wang for lending us a difficult-to-obtain, official Labubu doll, Raymond Yu for help setting up the initial BRIDGE table and FoundationStereo pipeline, Markus Grotz for assistance in setting up the Franka controller stack (robits), Yi Li for HAMSTER baseline help, Andy Tang for assisting with initial BRIDGE camera alignment, and William Chen for providing exact measurements for us to align the BRIDGE camera positions as best as possible. We also thank Yondu.ai for hosting the Los Angeles Lerobot hackathon where we tried an early version of PEEK, and Yutai Zhou and Minjune Hwang for joining us in the competition.

Additionally, we acknowledge funding from the Army Research Lab and compute resources from the University of Southern California's Center for Advanced Research Computing (CARC).

REFERENCES

[1] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, K. E. Bekris, K. Hauser, S. L. Herbert, and J. Yu, Eds., 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[3] K. Black, N. Brown, D. Driess, *et al.*, "*pi*_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[4] G. Yan, J. Zhu, Y. Deng, *et al.*, "Maniflow: A dexterous manipulation policy via flow matching," in *Conference on Robot Learning (CoRL)*, 2025.

[5] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh, "A taxonomy for evaluating generalist robot policies," 2025.

[6] P. Atreya, K. Pertsch, T. Lee, *et al.*, "Roboarena: Distributed real-world evaluation of generalist robot policies," in *Proceedings of the Conference on Robot Learning (CoRL 2025)*, 2025.

[7] Y. Li, Y. Deng, J. Zhang, *et al.*, "HAMSTER: Hierarchical action models for open-world robot manipulation," in *The Thirteenth International Conference on Learning Representations*, 2025.

[8] J. Lee, J. Duan, H. Fang, *et al.*, *Molmoact: Action reasoning models that can reason in space*, 2025.

[9] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3d diffuser actor: Policy diffusion with 3d scene representations," in *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024.

[10] J. Lin, H. Yin, W. Ping, P. Molchanov, M. Shoeybi, and S. Han, "Vila: On pre-training for visual language models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2024, pp. 26 689–26 699.

[11] J. Shi, J. Qian, Y. J. Ma, and D. Jayaraman, "Plug-and-play object-centric representations from what and where foundation models," *ICRA*, 2024.

[12] D. Emukpere, R. Deffayet, B. Wu, *et al.*, *Disentangled object-centric image representation for robotic manipulation*, 2025.

[13] R. Mirjalili, T. Jülg, F. Walter, and W. Burgard, "Augmented reality for robots (arro): Pointing visuomotor policies towards visual robustness," *arXiv preprint arXiv:2505.08627*, 2025.

[14] A. J. Hancock, A. Z. Ren, and A. Majumdar, "Run-time observation interventions make vision-language-action models more visually robust," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 9499–9506.

[15] C. Yuan, S. Joshi, S. Zhu, H. Su, H. Zhao, and Y. Gao, "Roboengine: Plug-and-play robot data augmentation with semantic robot segmentation and background generation," *arXiv preprint arXiv:2503.18738*, 2025.

[16] H. Huang, F. Liu, L. Fu, *et al.*, "Otter: A vision-language-action model with text-aware feature extraciton," *arXiv preprint arXiv:2503.03734*, 2025.

[17] P. Li, Y. Wu, Z. Xi, *et al.*, "Controlvla: Few-shot object-centric adaptation for pre-trained vision-language-action models," *arXiv preprint arXiv:2506.16211*, 2025.

[18] S. Liu, Z. Zeng, T. Ren, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[19] F. Locatello, D. Weissenborn, T. Unterthiner, *et al.*, "Object-centric learning with slot attention," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 11 525–11 538.

[20] O. Biza, S. van Steenkiste, M. S. M. Sajjadi, G. F. Elsayed, A. Mahendran, and T. Kipf, "Invariant slot attention: Object discovery with slot-centric reference frames," in *ICML*, 2023.

[21] Y. Zhang, D. W. Zhang, S. Lacoste-Julien, G. J. Burghouts, and C. G. M. Snoek, "Unlocking slot attention by changing optimal transport costs," in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., ser. Proceedings of Machine Learning Research, vol. 202, PMLR, 23–29 Jul 2023, pp. 41 931–41 951.

[22] M. Mosbach, J. N. Ewertz, A. Villar-Corrales, and S. Behnke, "Sold: Slot object-centric latent dynamics models for relational manipulation learning from pixels," in *International Conference on Machine Learning (ICML)*, 2025.

[23] J. Gu, S. Kirmani, P. Wohlhart, *et al.*, *Rt-trajectory: Robotic task generalization via hindsight trajectory sketches*, 2023.

[24] D. Niu, Y. Sharma, G. Biamby, *et al.*, "LLARVA: Vision-action instruction tuning enhances robot learning," in *8th Annual Conference on Robot Learning*, 2024.

[25] C.-P. Huang, Y.-H. Wu, M.-H. Chen, Y.-C. F. Wang, and F.-E. Yang, "Thinkact: Vision-language-action reasoning via reinforced visual latent planning," *arXiv preprint arXiv:2507.16815*, 2025.

[26] R. Zheng, Y. Liang, S. Huang, *et al.*, "TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies," in *The Thirteenth International Conference on Learning Representations*, 2025.

[27] T. Xiao, H. Chan, P. Sermanet, *et al.*, "Robotic skill acquistion via instruction augmentation with vision-language models," in *Proceedings of Robotics: Science and Systems*, 2023.

[28] J. Zhang, J. Zhang, K. Pertsch, *et al.*, "Bootstrap your own skills: Learning to solve new tasks with large language model guidance," in *7th Annual Conference on Robot Learning*, 2023.

[29] J. Zhang, K. Pertsch, J. Zhang, and J. J. Lim, "Sprint: Scalable policy pre-training via language instruction relabeling," in *International Conference on Robotics and Automation*, 2024.

[30] L. Smith, A. Irpan, M. G. Arenas, *et al.*, "Steer: Flexible robotic manipulation via dense language grounding," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 16 517–16 524.

[31] W. Chen, S. Belkhale, S. Mirchandani, *et al.*, "Training strategies for efficient embodied reasoning," in *9th Annual Conference on Robot Learning*, 2025.

[32] K. Pertsch, Y. Lee, and J. J. Lim, "Accelerating reinforcement learning with learned skill priors," in *Conference on Robot Learning (CoRL)*, 2020.

[33] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum, "{opal}: Offline primitive discovery for accelerating offline reinforcement learning," in *International Conference on Learning Representations*, 2021.

[34] J. Zhang, M. Heo, Z. Liu, *et al.*, "EXTRACT: Efficient policy learning by extracting transferrable robot skills from offline data," in *Conference on Robot Learning*, 2024.

[35] W. Yuan, J. Duan, V. Blukis, *et al.*, "Robopoint: A vision-language model for spatial affordance prediction in robotics," in *8th Annual Conference on Robot Learning*, 2024.

[36] O. X.-E. Collaboration, A. O'Neill, A. Rehman, *et al.*, *Open X-Embodiment: Robotic learning datasets and RT-X models*, 2023.

[37] A. Khazatsky, K. Pertsch, S. Nair, *et al.*, "Droid: A large-scale in-the-wild robot manipulation dataset," 2024.

[38] B. Liu, Y. Zhu, C. Gao, *et al.*, "Libero: Benchmarking knowledge transfer for lifelong robot learning," *arXiv preprint arXiv:2306.03310*, 2023.

[39] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht, "Cotracker: It is better to track together," in *European Conference on Computer Vision*, Springer, 2025, pp. 18–35.

[40] H. Walke, K. Black, A. Lee, *et al.*, "Bridgedata v2: A dataset for robot learning at scale," in *Conference on Robot Learning (CoRL)*, 2023.

[41] M. Grotz, M. Shridhar, Y.-W. Chao, T. Asfour, and D. Fox, "Peract2: Benchmarking and learning for robotic bimanual manipulation tasks," in *CoRL 2024 Workshop on Whole-body Control and Bimanual Manipulation: Applications in Humanoids and Beyond*.

[42] K. Zakka, *Mink: Python inverse kinematics based on MuJoCo*, version 0.0.11, May 2025.

[43] B. Wen, M. Trepte, J. Aribido, J. Kautz, O. Gallo, and S. Birchfield, "Foundationstereo: Zero-shot stereo matching," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5249–5260.

[44] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019.

[45] M. Memmel, J. Berg, B. Chen, A. Gupta, and J. Francis, "Strap: Robot sub-trajectory retrieval for augmented policy learning," *arXiv preprint arXiv:2412.15182*, 2024.

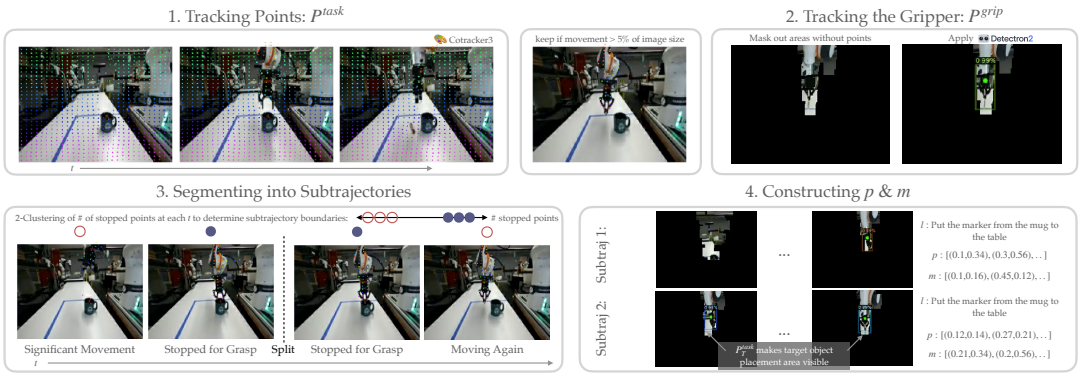


Fig. 6: **Data Labeling Pipeline.** A detailed overview of the data labeling pipeline as described in Section I-A: We (1) use CoTracker3 [39] to detect moving points across each trajectory, points are discarded if they do not move significantly, and the rest become task-relevant points P^{task}; (2) mask areas without P^{task} to black and apply a pre-trained gripper detector to construct 2D gripper path points P^{grip}; (3) segment each trajectory into subtrajectories; and (4) construct gripper paths p and task-relevant masking points m for each subtrajectory. Notice in (4) that target object placement areas become visible beforehand through including points from the last timestep P_T^{task}.

APPENDIX I

A. Data Annotation Pipeline Details

Tracking Points. Given a trajectory of image observations $o_{1:T}$, we apply CoTracker3 [39] to the video to track all moving points within the scene. Points are initialized in a uniform grid across the entire pixel space (a 15×15 grid to 30×30 grid depending on how far objects are from the camera). We initialize this point grid from the middle of the trajectory because in many datasets, the gripper is not visible at the first timestep. Running CoTracker returns a set of all points and their normalized image locations across the trajectory. We discard any point that does not move much throughout the trajectory, i.e., less than 5% of the image size, because they are unlikely to be task-relevant. The N remaining points at each timestep t, $[\{(x,y)_i\}_{i=1}^N]_{t=1}^T$ with $(x,y) \in [0,1]^2$, indicate both objects that move at some point during the trajectory and the robot gripper location. Therefore, these are the *task-relevant* points $P_t^{task} = \{(x,y)_{i=1}^N\}_t$. See Figure 6 (1) for an overview of point tracking.

Tracking the Gripper.

However, we found that naïvely applying the gripper detector resulted in noisy predictions that often did not include the robot. To reduce the noise, we only keep pixels in o_t around the significant points P_t^{task}, essentially masking out distractions irrelevant to the task. We apply the detector to this reduced representation to obtain per-timestep gripper bounding boxes (filling in frames with no detected gripper with the average of adjacent detections) and average across them to obtain the end-effector points $P_t^{grip} = [(x_t, y_t)]_t$. See Figure 6 (2) for a visual depiction.

Segmenting into Subtrajectories. Because masks m_t include points at the final timestep T, $m_t = P_t^{task} \cup P_T^{task}$, constructing m_t from long-horizon trajectories can violate the minimality principle. For example, the policy does not have to know the placement location of an object until it has actually picked it up. Therefore, we automatically break down the trajectories into *subtrajectories*. Our key insight is that when many task-relevant points *stop moving*, the robot is likely to be manipulating an object. Vice versa, when those points start moving again, the robot is likely reaching or carrying an object. This creates a natural approach to splitting a trajectory: by how many points in P^{task} stop moving.

For each frame, o_t, we track how many points in P_t^{task} don't move for the next 5 frames (3 for BRIDGE due to its higher control frequency). This creates a list of length T containing the number of "stopped points" for each timestep, t. On this list, we perform K-Means clustering with $K = 2$, where the cluster with the smaller mean (fewer stopped points) corresponds to significant movement, e.g., the robot arm reaching an object, and the cluster with the larger mean corresponds to the robot performing fine-grained manipulation, e.g., grasping. Finally, we use these cluster assignments to find continuous sections $i, i+1, .., j-1, j$ where the robot is manipulating an object, and use the middle frame of these sections $(j+i)/2$ as subtrajectory split points. This procedure results in a split of subtrajectories that end when the robot finishes a manipulation and start before it moves onto the next object manipulation. These subtrajectories create natural, shorter-horizon VLM prediction targets (see Figure 6 (3)).

B. VLM and Implementation Details

Dataset Preparation. For **OXE**, we discard the initial 20% of trajectories and augment subtrajectories by re-sampling start/end points $5\times$ within the first/last 20% of the remainder. We exclude *FurnitureBench, Roboturk, Dobbe, BerkeleyCableRouting, LangTable, Kuka,* and *FMB*. For **LIBERO-90**, we follow [2] by re-rendering 3,958 successful demonstrations to 256×256.

Postprocessing. To minimize inference latency, we simplify p_t and m_t using the Ramer–Douglas–Peucker algorithm with thresholds $\epsilon = 0.05$ and $\epsilon = 0.1$, respectively.

Env.	Train (H)	Rollout (H)	Chunk (A)
BRIDGE	30	25	5
Sim-Real	32	32	8

TABLE II: Hyperparameters for VLM + policy execution.

Training & Rollout. We use specific query frequencies (H) and action chunk lengths (A) as detailed in Table II.