Visual-Geometric Policy Steering from In-Context Aligned Abstractions

Arthur Bucker[1], Pramod Anantharam[2], Jonathan Francis[1,2] and Jean Oh[1,3]

***Abstract*— Adapting robot policies to out-of-distribution (OOD) settings with limited real-world data remains a key bottleneck to generalist robots. The multimodality and scarcity of robotics data lead end-to-end approaches to overfit and break under small distribution shifts. Test-time adaptation can alleviate this by correcting frozen policies at inference, but most methods ground their correction in a signal learned implicitly from robotic datasets, making it susceptible to the same shifts as the policy it corrects. Recent methods that instead exploit explicit spatial information are typically limited to coarse corrections, missing the nuanced, task-relevant motion most manipulation tasks require. We correct existing policies by transferring the nuanced motion of successful in-distribution rollouts, captured along meaningful geometric axes, known here as object-relational distance pairs. Our framework leverages the semantic understanding of Large Vision Models (LVMs) and Vision-Language Models (VLMs) reasoning, which combines the task-relevant entity pairs and their object-relational distance projections into a single tailored visual-geometric guidance signal. In a simulation benchmark, it efficiently corrects flow-matching policies under pose and task-goal shifts; on a real robot experiment, it widens the region of the workspace the frozen policy covers, more than doubling its success rate on cable placements one grid step outside the training positions, and steers a single-task policy toward objects it was never trained on while nearly eliminating object confusion. Project website: https://icra2027ab.github.io/**

I. Introduction

Robotic manipulation policies trained at scale achieve strong in-distribution performance [1]–[5], yet degrade under modest shifts in object appearance, geometry, layout, language instruction, or embodiment [6]–[8]. Retraining for every new setting is costly and data-hungry: collecting hundreds of demonstrations per task in each new environment is impractical.

Test-time adaptation offers an alternative: keep the policy fixed and use external correction signals to rerank its actions [9]–[13] or steer their generation [14]–[16]. However, many steering approaches rely on learned value, reward, or latent signals that themselves fail under distribution shift, as reward models are prone to overoptimization on out-of-distribution inputs [17]. Recent work introduces explicit spatial cues [10], [18], [19], but is typically limited to coarse, high-level corrections and does not reliably shape the fine-grained, task-relevant motion many manipulation tasks require.

[1]Arthur Bucker, Jonathan Francis and Jean Oh are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA {abucker,jmf1,jeanoh}@cs.cmu.edu

[2]Pramod Anantharam and Jonathan Francis are with the Robot Learning Lab, Bosch Center for Artificial Intelligence, Pittsburgh, PA 15222, USA

[3]Jean Oh is also with Lavoro AI, Inc., Pittsburgh, PA 15232, USA

This paper addresses that gap with a visual-geometric steering framework that transfers behavior from retrieved in-distribution demonstrations to OOD scenes without updating policy weights. Our key idea is a single guidance signal built from object-relational distance pairs, evaluated per timestep along VLM-selected axes, so that it captures the nuanced motion of the retrieved rollout rather than only coarse endpoints. We (i) retrieve relevant demonstrations by visual and task similarity, (ii) extract object-relational distance-pair trajectories along automatically selected axes, and (iii) inject a differentiable guidance signal spanning position, orientation, and grasp directly into the policy's action-chunk generation (Figure 1). The guidance is modular, test-time only, and compatible with pretrained flow-matching or diffusion action heads.

We evaluate on challenging OOD settings in LIBERO-Pro [8]. Across two base policies, $\pi_{0.5}$ and GR00T N1.7 [2], [3], our approach consistently improves success rates under pose and task-goal shifts, outperforming VLS, our main training-free gradient-based steering baseline. We further deploy the complete pipeline, including open-vocabulary perception, on a physical UR5 under positional and task shift, where it widens the region of the workspace the frozen policy covers, more than doubling its success rate one grid step outside the training positions, and redirects a single-task policy toward objects it was never trained on, reducing object confusion by an order of magnitude. Our contributions are:

1) A retrieval-to-steering pipeline that transfers behavior to OOD environments through explicit visual-geometric abstractions
2) A distance-pair representation that captures the nuanced motion of a demonstration by tracking VLM-selected object relations per timestep along automatically chosen axes
3) An empirical study on LIBERO-Pro and a UR5 deployment showing consistent gains over existing steering methods without any weight update.

II. Related Works

A. Policy Steering: Flow-Matching and Diffusion

Policy steering adapts frozen generative policies at inference time through external guidance signals, avoiding costly retraining [15], [16], [18]. Methods differ primarily in how they construct those signals.

Implicit and gradient-free approaches. Several works learn value or reward functions and use them to bias action sampling. VGPS [9] computes a value-weighted score to

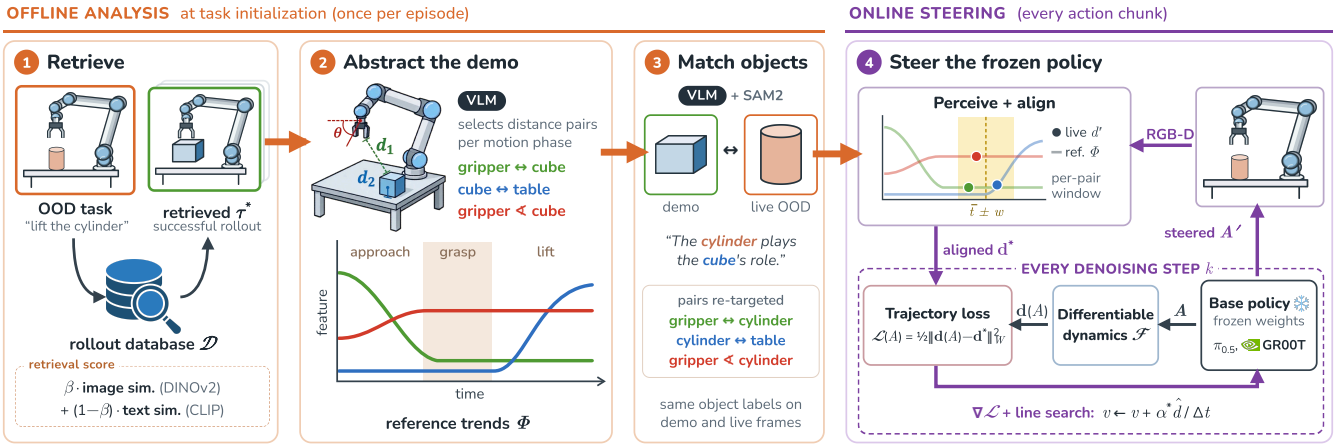


Fig. 1. **Overview of our visual-geometric policy steering framework.** *Offline analysis (at task initialization, once per episode):* (1) we retrieve the most relevant successful in-distribution rollout τ^* by weighted image (DINOv2) and text (CLIP) similarity; (2) a VLM selects the task-relevant object-relational distance pairs per motion phase (e.g., gripper–cube distance d_1, cube–table distance d_2, gripper orientation θ), evaluated per timestep into the reference trends Φ; (3) a VLM with SAM2 matches the demonstration objects to the live OOD scene (e.g., the cylinder takes the cube's role), re-targeting each pair. *Online steering (every action chunk):* (4) perception projects the live scene into the same distance space and aligns each pair to Φ within a window, giving the reference $\mathbf{d}^*$. At every denoising step, a differentiable dynamics model $\mathcal{F}$ forecasts the features $\mathbf{d}(A)$ of a candidate chunk A, and a line-searched gradient step on the trajectory loss corrects the frozen policy's velocity field, yielding the steered chunk A'.

rerank diffusion samples, and DynaFlow [20] embeds physical dynamics into the flow-matching vector field. Effective in distribution, these methods struggle under OOD shifts where the learned models themselves stop providing reliable guidance, such as fine-grained or novel-object interactions absent from currently limited robotics datasets.

Explicit spatial approaches. A complementary line grounds guidance in semantically aligned geometric structures. OmniGuide [19] constructs guidance fields from structured scene representations, and GRAPPA [10] and VLS [18] use explicit guidance functions over 3D geometric features. Despite their explicit spatial cues aligned with LVM semantic features, these methods target coarse, high-level corrections and lack mechanisms to align fine-grained contact and motion detail in OOD settings.

Our work bridges this gap with a single object-relational visual-geometric signal [21] that, by tracking VLM-selected distance pairs per timestep along meaningful axes, aligns the nuanced motion of a demonstration rather than only its coarse endpoints. Steering the denoising process with this differentiable signal, like [16], improves OOD generalization without modifying base policy weights.

B. Visual-geometric representations

Structured geometric representations have demonstrated the ability to enhance policy generalization. Wang et al. [22] encode 3D keypoints for tool-object relations, and [23] fuses spatial and semantic features for strong manipulation performance. HEPi [24] organizes robots and objects as graphs, but these methods typically rely on latent, non-transparent spaces.

III. Method

Our goal is to steer a pretrained policy $\pi(A_t|o_t)$ to succeed in an out-of-distribution (OOD) target environment by leveraging relevant in-distribution rollouts $\mathcal{D} = \{\tau_i\}$. We propose a modular, test-time framework that generates a single visual-geometric guidance signal, which transfers the nuanced motion of a retrieved successful rollout by tracking it along VLM-selected object-relational distance axes, and uses this signal to correct the policy's action generation process. An overview of our framework is shown in Figure 1.

A. Problem Formulation

We consider a pretrained generative policy $\pi(A_t \mid o_t)$ mapping an observation $o_t \in \mathbb{O}$ to an action chunk $A_t \in \mathbb{A}$, trained on an in-distribution environment and held *fixed* throughout, and a database of successful in-distribution rollouts $\mathcal{D} = \{\tau_i\}_{i=1}^{|\mathcal{D}|}$ with $\tau_i = \{(o_0, A_0), \ldots, (o_H, A_H)\}$. At test time the robot operates in an OOD environment whose observations o'_t exhibit shifts in appearance, object geometry, layout, language instruction, or embodiment.

Goal. Without updating π, generate corrected action chunks $A'_0, \ldots, A'_T$ that transfer the task-relevant behavior from $\mathcal{D}$ to the OOD scene. We distill the retrieved demonstration $\tau^* \in \mathcal{D}$ into a compact, task-tailored guidance signal: a set of object-relational distance pairs that evolve over time and encode an efficient heuristic for completing the task. This time-varying signal then steers the policy's action generation at inference.

B. Background: Steering the Denoising Process

The policies we target synthesize an action chunk by iteratively denoising a noisy sample, either by integrating a learned velocity field (flow matching) or by following a learned score (diffusion). Such a frozen policy can be *steered* at inference, without any weight update, by perturbing each denoising step with the gradient of an auxiliary guidance objective: an idea originating in classifier guidance [25], generalized to arbitrary differentiable objectives by training-free universal guidance [26], and used by diffusion planners to bias sampling toward high-return plans [27]. For flow matching, the correction is applied to the velocity field rather than the score.

Recent robot-policy work adopts this mechanism, differing mainly in *what* guidance signal is constructed: human interactions [16], learned dynamics or value models [9], [15], or explicit spatial energies [10], [18], [19]. Our contribution is thus not the steering machinery, which we inherit, but the guidance *signal*: a visual-geometric, object-relational distance-pair objective distilled from a temporally aligned retrieved demonstration (Sections III-D and III-E).

C. Demonstration Retrieval and Abstraction

Retrieval. Given the current OOD task defined by initial observation o'_0 and language instruction ℓ, we retrieve the most relevant demonstration by combining a visual and a language similarity as a weighted average:

$$\tau^* = \arg\max_{\tau_i \in \mathcal{D}} \beta \,\mathrm{sim}(\psi_v(o'_0),\, \psi_v(\tau_i[0])) + (1-\beta)\,\mathrm{sim}(\psi_\ell(\ell),\, \psi_\ell(\ell_i))\,, \tag{1}$$

where $\mathrm{sim}(a,b) = \cos(a,b)$ is the cosine similarity between ℓ_2-normalized embeddings, ψ_v embeds the initial RGB observation with a DINOv2 image encoder [28] (averaged over frames), and ψ_ℓ embeds the language instruction with a CLIP text encoder [29]. The weight β trades off visual against language similarity and is held fixed across experiments.

Abstraction. The retrieved demonstration τ^* is processed into an *object-relational distance-pair trajectory* Φ encoding the geometric relations between the end-effector (EEF) and task-relevant objects (Section III-D). Because these relations are evaluated per timestep along VLM-selected axes, Φ captures how the demonstrated motion unfolds along the task-relevant axes rather than only its coarse endpoints.

D. Visual-Geometric Guidance via Object-Relational Distance Pairs

We represent task structure as a set of pairwise geometric relations between task-relevant entities, where both the relevant entities *and* the appropriate distance projection are determined automatically per sub-trajectory phase. Rather than a single coarse constraint, evaluating these relations at every timestep yields a time-resolved profile that captures the nuanced motion of the reference rollout along the meaningful axes the VLM selects (Appendix B).

Sub-trajectory decomposition. We segment τ^* into M semantically coherent phases $\{ST_m\}_{m=0}^{M-1}$ following the velocity-based heuristic of STRAP [30], splitting at low end-effector velocity points. Each phase's end-effector path is rendered as a distinct colored overlay on the demo frame, giving the VLM a visual summary of how the motion unfolds.

Scene segmentation. We obtain object instance masks $\{(s_j, \mathrm{name}_j)\}$ with an open-vocabulary pipeline: a VLM (Gemini 3 Flash [31]) proposes a grounded bounding box per task-relevant object and SAM2 [32] converts each box into a precise mask, with s_j the instance ID and name_j its label. Running the same pipeline on the demonstration frame and the live OOD frame gives the two scenes a consistent language-grounded labeling, and yields the segmentation- and sub-trajectory-overlay images fed to the VLM.

VLM-guided distance pair selection. A VLM is queried with the task description, the original RGB frame, the segmentation-overlay image, and the sub-trajectory-overlay image. For each sub-trajectory phase ST_m the VLM outputs a set of entity pairs $\mathcal{P}_m$, each annotated with a *relation type* and an optional projection:

- **positional**: a **centroid** (Euclidean distance between 3D centroids) or **surface** (shortest source-to-surface distance, used when contact or proximity is critical) distance, with an optional *projection* that keeps only the signed component along an **axis** $\hat{u} \in \mathbb{R}^3$ or the in-plane component w.r.t. a **plane** normal $\hat{n} \in \mathbb{R}^3$ (null retains the full 3D distance);
- **orientation**: the alignment of an end-effector body-frame vector $R\,\hat{e}$ with a selected axis (a signed cosine) or plane (the residual in-plane magnitude), constraining pose rather than position;
- **gripper**: the gripper aperture, capturing grasp timing.

The source of a positional/orientation relation is the end-effector, a fingertip, or a grasped object, and the paired entity is a segmented object identified by its instance ID. Each phase thus receives its own relation set, grounded in the visual content of the demonstration, with position, orientation, and grasp in a single common form.

Geometric feature trajectory. For phase m and timestep $t \in ST_m$, each relation $n \in \mathcal{P}_m$ is reduced to a scalar feature:

$$d_t^n = \Pi_n(\delta_t^n)\,, \quad \phi^t = [d_t^n]_{n \in \mathcal{P}_{m(t)}} \in \mathbb{R}^{|\mathcal{P}_{m(t)}|}, \tag{2}$$

where δ_t^n is the raw geometric quantity of relation n (a centroid/surface distance vector, a body-frame orientation vector, or the gripper aperture) and $\Pi_n(\cdot)$ is its read-out: the configured axis/plane projection or identity for positional relations, the signed-cosine/in-plane projection for orientation relations, and the identity for gripper relations. Object centroids and surface points are obtained by unprojecting the segmentation mask with the depth image. This defines the visual-geometric trajectory $\Phi = \{\phi^t\}_{t=0}^{H}$.

Decoding an action chunk to geometric features. In simulation, the base policies we steer emit operational-space-control (OSC) action chunks: per-step end-effector linear and angular velocities plus a gripper command. To evaluate the relations on a candidate chunk A_t, a differentiable dynamics model $\mathcal{F}$ rolls it out from the current state $s_0 = (p_0, R_0, g_0)$,

$$\{(p_{t'}, R_{t'}, g_{t'})\}_{t'=1}^{T} = \mathcal{F}(A_t,\, s_0), \tag{3}$$

predicting end-effector position, orientation, and gripper aperture at each chunk step t'; fingertip and held-object source points follow by forward kinematics. Combined with the object centroids and surface points read from the current observation, this turns a candidate chunk into a *forecast* of the distance pairs it would induce: a predicted feature $d_{t'}^n(A_t)$ for every relation n and step t', under the same pairs $\mathcal{P}_{m^*}$ and projections selected on the demonstration. The whole read-out is differentiable and JIT-compilable, adding only a lightweight forward pass, so it can be evaluated at every denoising step. We implement $\mathcal{F}$ as a kinematic integrator

of the OSC velocities for the simulated experiments (Appendix F). In the real world, the policy acts in joint positions, so $\mathcal{F}$ is simply the UR5 forward kinematics.

Visual-geometric guidance loss. Let $\{d_{t'}^{*n}\}$ be the reference distance trajectory of the retrieved demonstration, aligned to the robot's current progress (Section III-E). The guidance objective penalizes the per-relation, per-step deviation from this reference across the entire chunk:

$$\mathcal{L}_{guide}(A_t) = \tfrac{1}{2} \sum_{n=1}^{N} \sum_{t'=1}^{T} w_n \left(d_{t'}^{n}(A_t) - d_{t'}^{*n} \right)^2, \qquad (4)$$

where N is the number of active relations, T the chunk horizon, and $w_n \geq 0$ a per-relation weight combining the VLM-assigned importance with a relevance gate that activates each relation only as the end-effector approaches the relevant object (Section III-E). Since orientation and gripper relations enter the same sum, a single objective jointly captures position, orientation, and grasp timing along the whole predicted chunk rather than at a single waypoint.

E. Differentiable Guidance Injection

Trajectory alignment. The reference targets $d_{t'}^{*n}$ in Eq. 4 are obtained by aligning the live execution to the demonstration in the feature space. Instead of matching all relations to one shared frame, we align *each relation independently* within a bounded window around a shared cursor $\bar{t}$ (the MIXED scheme in our implementation): for relation n with current live value d'^{n},

$$t_n^* = \underset{|t-\bar{t}| \leq w}{\arg\min} \left(d'^{n} - d_t^n \right)^2, \qquad (5)$$

and we read $d_{t'}^{*n} = d_{t_n^*+t'}^{n}$ as that relation's per-step references. The cursor is *not* constrained to advance monotonically: within the active sub-trajectory it may move forward or backward by at most a fixed number of steps per update, so the guidance can follow corrective, retry behavior (e.g. re-approaching an object after a missed grasp) rather than being dragged through stages the robot has not completed. Sub-trajectory switching is likewise bidirectional and gated on sub-goal satisfaction, analogous to the stage-wise relational keypoint constraints of ReKep [21]: the active relation set $\mathcal{P}_m$ advances when a phase's sub-goal is met and falls back if it no longer holds. Window sizes and switching hysteresis are in Appendix E.

Orientation without rotation-representation artifacts. Euclidean distances over rotation parameters cause singularities and trajectory artifacts. Rather than add a separate rotation loss, we constrain orientation *inside* $\mathcal{L}_{guide}$ through the orientation relations of Section III-D, each projecting a rotated end-effector body-frame vector $R_{t'}\hat{e}$ onto a VLM-selected axis or plane. Because orientation is propagated as a rotation and only its projection enters the loss, the objective stays well-behaved on $SO(3)$ while remaining a plain distance-pair term.

Line-searched guidance correction. We inject guidance at every denoising step k, without modifying policy weights. Let $A_t^{(k)}$ be the current noisy chunk and v the policy's predicted velocity field (for diffusion, the score-derived update). We form a unit descent direction from the guidance gradient at the policy's tentative next iterate,

$$g = \nabla_A \mathcal{L}_{guide}\left(A_t^{(k)} + \Delta t\, v \right), \qquad \hat{d} = -\,g/\|g\|, \qquad (6)$$

so the correction magnitude is set by a step size rather than the raw gradient scale. A single global step is brittle, as early high-noise steps tolerate only small corrections while later ones benefit from larger ones, so we select the step *per denoising step* with a one-dimensional line search,

$$\alpha^\star = \underset{\alpha \in [0,\, \alpha_{\max}]}{\arg\min}\ \mathcal{L}_{guide}\left(A_t^{(k)} + \alpha\, \hat{d} \right), \qquad (7)$$

solved by a golden-section search [33] with a *do-no-harm* guard: if no positive step reduces the loss, $\alpha^\star = 0$ and the unsteered base step is kept (Algorithm 1). The step is injected into the velocity field,

$$v \leftarrow v + \frac{\alpha^\star}{\Delta t}\, \hat{d}\, c, \qquad (8)$$

so the sampler update becomes $A_t^{(k+1)} = A_t^{(k)} + \Delta t\, v + c\,\alpha^\star\,\hat{d}$: the base step plus a descent step of length $c\,\alpha^\star$ along $\hat{d}$, where c is a global guidance scale. This reshapes the velocity field of a flow-matching policy, or the score estimate of a diffusion one, analogous to classifier guidance [25], while the policy weights stay frozen.

Action-space mapping. In simulation, the policies we evaluate emit OSC delta actions in the end-effector Cartesian frame, so $\nabla_{A_t}\mathcal{L}_{guide}$ is already expressed in the policy's action space and the correction is applied without kinematic remapping. For the real-world experiment, the policies operate in joint-space, and the same Cartesian gradient maps into the action space through the pseudo-inverse Jacobian, $\delta A_t = J(q_t)^\dagger \nabla_{e_t}\mathcal{L}_{guide}$, analogous to how dynamics-aware flow-matching methods enforce embodiment-consistent corrections [20].

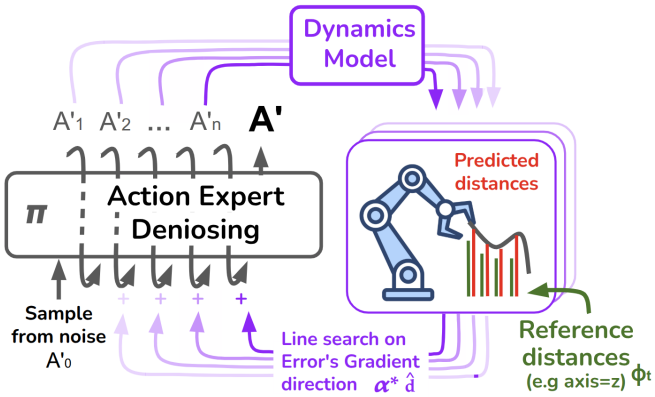


Fig. 2. Differentiable guidance injection. At each denoising step, the candidate chunk is rolled out by forward kinematics, and the guidance loss (position, orientation, gripper) is differentiated; a golden-section line search sets the step size, and the correction is applied to the policy's velocity field, steering the trajectory without modifying weights.

IV. Implementation Details

Base policies. We evaluate on two pretrained flow-matching manipulation policies: $\pi_{0.5}$ [2], a vision-language-action policy pretrained on a large multi-embodiment dataset, and GR00T N1.7 [3], a humanoid foundation policy. Both are kept frozen and only the test-time guidance signal is applied during denoising.

Simulation benchmark. We report simulation results on LIBERO_PRO [8], an OOD extension of LIBERO introducing pose, object, instruction, and task-goal shifts.

Guidance hyperparameters. At each denoising step the guidance scale $\alpha^\star$ is chosen by the golden-section line search (Algorithm 1), whose do-no-harm guard always admits the unsteered base step ($\alpha^\star = 0$). The global scale c and the relevance-gating thresholds setting w_n in Eq. 4 are constant across all benchmarks and policies; only the search bound $\alpha_{\max}$ is rescaled per policy for its action parameterization (velocity field versus score). The remaining hyperparameters (retrieval weight β, STRAP thresholds, VLM prompts) are shared across experiments. Full values and the perception, alignment, and weighting details are in Appendix H.

V. Simulation Experiments

We evaluate on LIBERO-Pro across two base policies, measuring task success rate under multiple types of distribution shift, against VLS [18], our main training-free gradient-based steering baseline.

Q1: Does visual-geometric steering improve OOD generalization over existing policy steering baselines? We center the comparison on the *task* and *positional* perturbations, the regime where the base policies are weakest and the only one the VLS protocol evaluates. Like VLS, these experiments use the simulator's ground-truth segmentations and labels (real open-vocabulary perception in Appendix J).

TABLE I
Success rate (%) on the task and positional perturbations of LIBERO-Pro ($\pi_{0.5}$), each averaged over the four suites, and their mean (*Average*). *VLS reproduced in our setup using the released VLS codebase (*grad-only* disables its population resampling); see Appendix N.

Method ($\pi_{0.5}$)	Task	Pos	Average
No steering	26.3	26.3	26.3
VLS (reproduced, full)*	32.0	27.6	29.8
VLS (grad-only)*	17.3	21.5	19.4
Ours	**40.8**	**34.3**	**37.5**

Our guidance raises the task success rate from 26.3 to 40.8 and the positional rate from 26.3 to 34.3 (37.5 average against 26.3), surpassing full VLS (29.8). The gap is starkest against the compute-matched grad-only VLS: without population resampling it falls to 19.4, *below* the unsteered base, whereas our single-chain gradient guidance still improves by a wide margin. As both are gradient-only, the difference is attributable to the guidance *signal* rather than to VLS's sampling machinery. Conversely, where the base policy is already near-saturated (object and language, 75–95%) there is little gain, and the guidance can even be *detrimental*, perturbing an already-correct trajectory; the relevance gate (Section III-E) mitigates but does not eliminate this. The per-type breakdown is in Appendix I.

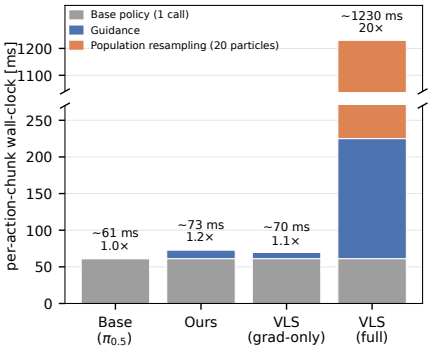


Fig. 3. Per-action-chunk inference overhead on $\pi_{0.5}$ (mean over the four suites, CUDA-synchronized). With our guidance a chunk takes ∼73 ms in total (1.2× the base's ∼61 ms), comparable to gradient-only VLS (∼70 ms total); full VLS costs ∼20× more, scaling with its particle count (20 here).

Generality across base policies. The guidance is independent of the base policy. It improves the average task-and-positional rate of both frozen flow-matching policies (Table II): 26.3 → 37.5 for $\pi_{0.5}$, and 9.9 → 11.8 for GR00T N1.7, whose base is near zero on positional shifts. The gain is larger for $\pi_{0.5}$, whose short action chunk the per-step guidance steers most directly.

Q2: What is the inference-time overhead of guidance? Unlike VLS, which resamples a population of particles at every denoising step, our guidance runs a single denoising chain and adds only a lightweight, JIT-compiled distance-pair loss during the per-step line search (no extra policy forward pass). On $\pi_{0.5}$ this is ∼12 ms over a single base-policy call (∼61 ms), a 1.2× overhead on par with gradient-only VLS (1.1×), whereas full VLS is ∼20× slower because its resampling re-runs the policy across particles (Figure 3). On hardware, perception adds 113 ± 2 ms per inference plus a one-time first-frame setup of 29.5 ± 3.1 s (Appendix L).

Q3: Which components of the guidance mechanism drive the gains? Turning the raw gradient of distance-pair errors into a usable correction relies on two mechanisms: the per-step line search that sets the step size (and rejects steps that increase the loss), and the normalization of the guidance gradient to a unit direction, which decouples the step from the raw gradient scale (Appendix G). We enable each cumulatively on top of the unsteered $\pi_{0.5}$ base (Table III).

Raw fixed-step gradient guidance is slightly worse than the

TABLE II
Generality across base policies (LIBERO-Pro; task and positional success rate %, averaged over suites, and their mean).

Base policy	Steering	Task	Pos	Average
$\pi_{0.5}$ (flow)	Base	26.3	26.3	26.3
	+ Ours	**40.8**	**34.3**	**37.5**
GR00T N1.7 (flow)	Base	19.3	0.5	9.9
	+ Ours	**20.0**	**3.5**	**11.8**

base (23.8 vs. 26.3 average): unnormalized distance gradients are poorly scaled, so a fixed step over- or under-shoots. The per-step line search recovers a large gain (23.8 → 35.0), and gradient normalization yields the full method and the best result (35.0 → 37.5). The geometric signal alone is thus necessary but not sufficient: the line search and normalization are what make it a net-positive correction.

TABLE III
GUIDANCE-MECHANISM ABLATION ($\pi_{0.5}$, LIBERO-PRO; SUCCESS RATE %, AVERAGED OVER THE FOUR SUITES). COMPONENTS ARE ENABLED CUMULATIVELY ON TOP OF THE UNSTEERED BASE. THE TWO INTERMEDIATE VARIANTS USE A 3-TRIAL SWEEP (30 EPISODES/CELL); THE BASE AND FULL METHOD USE THE 10-TRIAL MAIN RUNS.

Variant ($\pi_{0.5}$)	Task	Pos	Average
No guidance	26.3	26.3	26.3
+ gradient guidance (fixed step)	24.2	23.3	23.8
+ line search	34.2	**35.8**	35.0
+ gradient normalization (Ours)	**40.8**	34.3	**37.5**

VI. REAL-WORLD EXPERIMENTS

We deploy the *complete* pipeline on hardware, where grounding, depth, calibration and control are all subject to real noise, and ask whether the guidance still recovers the base policy's OOD failures. All real-world runs use the open-vocabulary perception stack of Appendix A (no privileged labels).

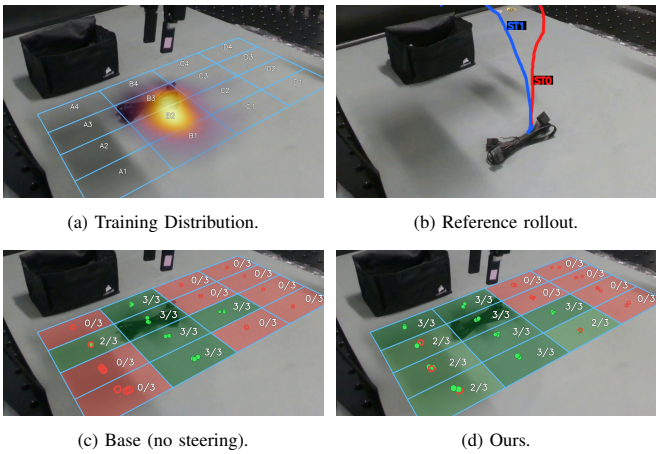
Hardware. A UR5 arm with a Robotiq 2F-140 parallel gripper. Object detection and 3D reconstruction are driven by a single fixed external RealSense D435, calibrated markerlessly with Kalib [34]. Policy inference, the VLM calls and the guidance loop all run on one RTX 5090 (Appendix K).

Task and base policy. We evaluate variations of a *cable bagging* task with a single $\pi_{0.5}$ [2] checkpoint, finetuned on 28 demonstrations of "Put all cables in the bag" and frozen throughout. It takes one wrist and one fixed camera view and predicts joint positions (Figure 4). The demonstration database $\mathcal{D}$ holds a *single successful rollout of that same policy* in the nominal configuration: no human demonstrations, no extra training data and no weight updates, so any OOD gain comes from how that one rollout is abstracted and re-applied.

- **Positional variation.** Using the "cable bagging" policy, we vary the cable placement over a 4×4 grid, 3 trials per cell. Only three of the sixteen cells overlap the positions seen in training; the rest lies between two and six standard deviations away (Section VI).
- **Task variation.** Using the "cable bagging" policy (trained only on cables), we vary the task by changing the name of the target object to be bagged: "cable", "banana", or "spoon". The three elongated objects are placed simultaneously inside an in-distribution region of the table (Section VI).

The guidance improves both axes (35.4 → 54.2 positional, 33.3 → 62.5 task), and the two failures it corrects differ in kind: under positional shift the base policy knows *what* to grasp and fails to reach it; under task shift it reaches confidently, but for the *wrong* object.

Q4: Does the guidance recover positional OOD failures? We run 48 placements per method, three per cell of a 4×4 grid spaced 15 and 7.5 cm, re-placing the cable at the same nominal cell for both methods (Figure 4). To put every placement on one scale, we fit a Gaussian to the 110 cable grasp positions in the 28 training demonstrations (5.6 and 6.2 cm along the two table axes) and report each rollout's measured start position as a Mahalanobis distance in σ. We cut three zones at 2 and 4.5σ; both cuts fall in empty gaps of the measured distances (Appendix M).



(a) Training Distribution. (b) Reference rollout.

(c) Base (no steering). (d) Ours.

Fig. 4. Positional sweep on cable bagging, in the camera view. (a) Training distribution of the 110 cable grasps in the demonstrations. (b) The single reference rollout, with its end-effector path split into sub-trajectories. (c, d) Per-cell outcomes: dots mark the measured cable position of every trial, filled green for success, red rings for failure.

In the near-OOD band the guidance raises the success rate by +33.3 points ($p = 0.028$, Table IV). It neither helps nor hurts in-distribution, and does not recover the far-OOD failures, for a pooled gain of +18.8 points ($p = 0.10$) over the sweep (Figure 4c,d). The per-method logistic fit restates this on a continuous scale (Figure 5): guidance pushes the 50% crossing out by 0.6σ and flattens the decay (Appendix M). Qualitatively, most failures, guided or not, occur in the grasping phase (Section VI-A); in the failed guided rollouts the approach still brought the end-effector over the cable.

Q5: Can the policy be steered toward objects it was not trained on? The base policy was trained to bag a cable and nothing else, so can the same frozen checkpoint be *redirected* to a different object simply by changing the relations the guidance tracks? We place the three objects of Figure 7 in the scene *simultaneously* and name one in the prompt. All three are elongated and share the graspable affordance the policy overfit to, so the setting isolates its tendency to approach whichever object its visual prior favours, regardless of the instruction. Figure 6 shows one such scene with the transferred sub-trajectories and the relations anchored on the named object. A trial is *correct* when the arm grasps the named object and *confused* when it grasps a distractor.

The base policy is confused on 45.8% of trials (11/24): prompted for the spoon it goes for the cable in half the trials, and prompted for the cable it splits evenly between cable

and banana; only the banana, the most visually distinctive, is reliably singled out (88%). Under guidance, confusion falls to 4.2% (1/24) and the named object is reached on 95.8% of trials. Task success rises from 33.3% to 62.5%, by less than the naming rate: steering reliably decides *which* object the frozen policy goes for, while bagging an object it was never trained to handle remains limited by its low-level grasp.

A. Failure modes

We annotate each real-world rollout with the first event that ended it: **wrong target** (grasps an object the prompt did not name), **grasp miss** (reaches the right object but never closes on it), **dropped**, and **other** (mostly perception dropouts and protective stops).

The two studies fail for opposite reasons under the base policy (Figure 8): in the positional sweep all 31 failures are grasp misses, the signature of reaching for a position never seen in training, whereas in the task study 11 of 16 are wrong-target grasps. Guidance corrects both, cutting grasp misses to 13 and leaving a single wrong-target trial, and the residual failures move downstream past contact, into drops during transport. The guidance acts on where the end-effector goes and which object it goes to, not on contact, so grasping and transport become the limiting stage once the approach is corrected, consistent with the far-OOD dead band of Section VI.

TABLE IV
POSITIONAL SWEEP ON CABLE BAGGING, 48 PLACEMENTS PER METHOD, EVERY PLACEMENT RUN BY BOTH. ZONES ARE MAHALANOBIS BANDS CUT AT 2 AND 4.5σ. CELLS GIVE THE SUCCESS RATE (%), SUCCESSES/ROLLOUTS AND THE 95% WILSON INTERVAL; Δ IS THE GAIN IN POINTS AND p A TWO-SIDED FISHER EXACT TEST.

	Success rate (%)			
Zone	Base	Ours	Δ	p
In-distribution ($< 2\sigma$)	100.0 9/9 [70, 100]	100.0 9/9 [70, 100]	0.0	1.00
Near-OOD (2 to 4.5σ)	29.6 8/27 [16, 48]	**63.0** 17/27 [44, 79]	**+33.3**	**0.028**
Far-OOD ($> 4.5\sigma$)	0.0 0/12 [0, 24]	0.0 0/12 [0, 24]	0.0	1.00
Pooled OOD ($> 2\sigma$)	20.5 8/39 [11, 36]	**43.6** 17/39 [29, 59]	+23.1	0.051
Pooled, full sweep	35.4 17/48 [23, 50]	**54.2** 26/48 [40, 67]	+18.8	0.10

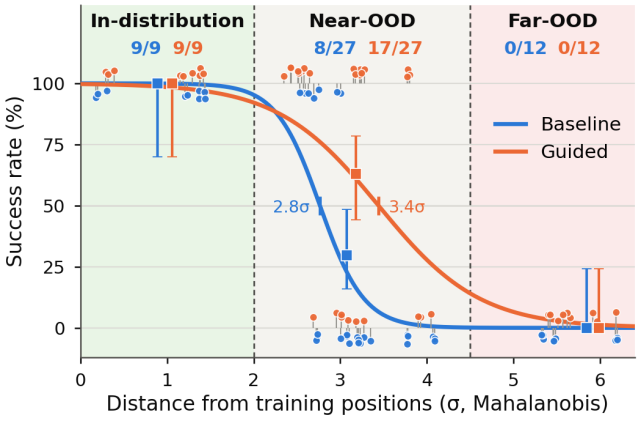


Fig. 5. Positional generalization. Every rollout is plotted at the Mahalanobis distance of its measured cable position (dots near 100 are successes, near 0 failures), with per-zone success rates and 95% Wilson intervals (squares) and a per-method logistic fit, crossing 50% at 2.8σ (base) and 3.4σ (ours).



Fig. 6. Disambiguating the target object in a cluttered scene. In red and blue, the distance pairs used per sub-trajectory (ST0 approach, ST1 transport).

VII. CONCLUSION

We presented a visual-geometric policy steering framework that improves the OOD generalization of frozen flow-matching policies without any weight update. By retrieving task-relevant in-distribution demonstrations and abstracting them into a single guidance signal of object-relational distance pairs, tracked per timestep along VLM-selected axes, our method injects differentiable corrections over position, orientation, and grasp directly into the policy's denoising process. On LIBERO-Pro, it consistently outperforms existing steering baselines under pose and task-goal shifts, and on a physical UR5 (Section VI), under real open-vocabulary perception, it improves over the unsteered base under positional shift and redirects a single-task policy to unseen objects while nearly eliminating object confusion.

Limitations. *(i) Perception.* The guidance is only as reliable as the grounding it is built on, which requires detecting, labeling, and localizing the task-relevant objects in both the

TABLE V
TASK VARIATION. *Named* IS THE RATE AT WHICH THE ARM WENT FOR THE OBJECT THE PROMPT NAMED; *Task SR* ALSO REQUIRES IT TO BE BAGGED; *Confusion* IS THE RATE OF GRASPING A DISTRACTOR. THE POLICY WAS ONLY TRAINED ON THE CABLE, THE BANANA AND THE SPOON ARE UNSEEN.

		Named (%)					
Method	Trials	Cable	Banana	Spoon	All	Task SR (%)	Confusion (%) ↓
Base (no steering)	24	38	88	38	54.2	33.3	45.8
Ours	24	**100**	**100**	**88**	**95.8**	**62.5**	**4.2**

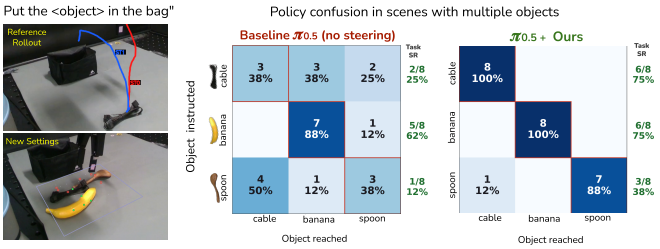
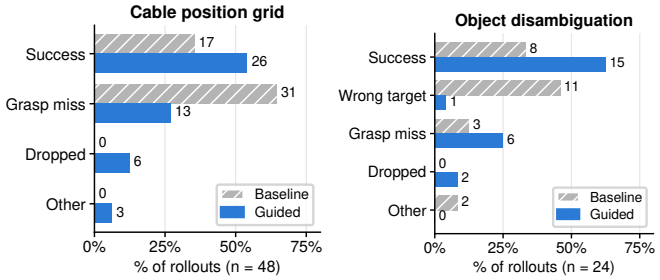


Fig. 7. Task variation on cable bagging. Left: the single reference rollout above the new setting in which cable, banana and spoon share the workspace. Center, right: confusion matrices (instructed object in rows, reached object in columns) for the base and the guided policy.



(a) Positional variation. (b) Task variation.

Fig. 8. Outcome breakdown of the real-world rollouts, base against guided, as a share of each method's rollouts (counts on the bars). Residual failures concentrate in the low-level grasp and transport, which the guidance signal does not address.

demonstration and the live scene. Our simulation experiments use ground-truth segmentation and labels to isolate the steering signal; under real open-vocabulary perception (Appendix J), missed detections, mislabeled instances, and imprecise masks or depth corrupt the distance pairs and substantially degrade performance. This gap in Table I makes perception the dominant source of error. *(ii) Sub-trajectory switching.* Progress depends on the bidirectional, sub-goal-gated phase transitions (Section III-E): when a sub-goal is mis-detected, the active relation set can advance too early, stall, or oscillate, steering toward the wrong relations. *(iii) Balancing guidance and policy behavior.* The single global guidance scale, tempered by the line search and do-no-harm guard, mediates but does not resolve the tension between following the guidance and preserving the policy's learned behavior. In contact-rich interactions such as grasping or insertion, over-steering disrupts the policy's low-level motor skills while under-steering fails to correct the OOD error; an adaptive, interaction-aware schedule could improve this balance.

ACKNOWLEDGMENT

This work was in part supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.RS-2025-25463654).

REFERENCES

[1] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[2] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai *et al.*, "$\pi_{0.5}$: A vision-language-action model with open-world generalization," *arXiv preprint arXiv:2504.16054*, 2025.

[3] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang *et al.*, "GR00T N1: An open foundation model for generalist humanoid robots," *arXiv preprint arXiv:2503.14734*, 2025.

[4] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, "OpenVLA: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[5] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 2024.

[6] J. Guo, Z. Wu, C. Tu, Y. Ma, X. Kong, Z. Liu, J. Ji, S. Zhang, Y. Chen, K. Chen *et al.*, "On robustness of vision-language-action model against multi-modal perturbations," *arXiv preprint arXiv:2510.00037*, 2025.

[7] S. Fei, S. Wang, J. Shi, Z. Dai, J. Cai, P. Qian, L. Ji, X. He, S. Zhang, Z. Fei *et al.*, "LIBERO-Plus: In-depth robustness analysis of vision-language-action models," *arXiv preprint arXiv:2510.13626*, 2025.

[8] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun, "LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization," *arXiv preprint arXiv:2510.03827*, 2025.

[9] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine, "Steering your diffusion policy with latent space reinforcement learning," *arXiv preprint arXiv:2506.15799*, 2025.

[10] A. Bucker, P. Ortega-Kral, J. Francis, and J. Oh, "GRAPPA: Generalizing and adapting robot policies via online agentic guidance," *IEEE Robotics and Automation Letters*, 2026.

[11] J. Kwok, C. Agia, R. Sinha, M. Foutter, S. Li, I. Stoica, A. Mirhoseini, and M. Pavone, "RoboMonkey: Scaling test-time sampling and verification for vision-language-action models," *arXiv preprint arXiv:2506.17811*, 2025.

[12] Y. Wu, R. Tian, G. Swamy, and A. Bajcsy, "From foresight to forethought: VLM-in-the-loop policy steering via latent alignment," *arXiv preprint arXiv:2502.01828*, 2025.

[13] Y. Wu, A. Li, T. Hermans, F. Ramos, A. Bajcsy, and C. Pérez-D'Arpino, "Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification," *arXiv preprint arXiv:2510.16281*, 2025.

[14] Y. Zhang, C. Wang, O. Lu, Y. Zhao, Y. Ge, Z. Sun, X. Li, C. Zhang, C. Bai, and X. Li, "Align-then-steer: Adapting the vision-language action models through unified latent guidance," *arXiv preprint arXiv:2509.02055*, 2025.

[15] M. Du and S. Song, "DynaGuide: Steering diffusion policies with active dynamic guidance," *arXiv preprint arXiv:2506.13922*, 2025.

[16] Y. Wang, L. Wang, Y. Du, B. Sundaralingam, X. Yang, Y.-W. Chao, C. Pérez-D'Arpino, D. Fox, and J. Shah, "Inference-time policy steering through human interactions," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 15 626–15 633.

[17] L. Gao, J. Schulman, and J. Hilton, "Scaling laws for reward model overoptimization," *International Conference on Machine Learning (ICML)*, 2023.

[18] S. Liu, I. S. Singh, Y. Xu, J. Duan, and R. Krishna, "VLS: Steering pretrained robot policies via vision-language models," *arXiv preprint arXiv:2602.03973*, 2026.

[19] Y. Song, L. Le, Y.-H. Park, J. Wang, J. Shi, L. Liu, J. Gu, E. Eaton, D. Jayaraman, and K. Daniilidis, "OmniGuide: Universal guidance fields for enhancing generalist robot policies," *arXiv preprint arXiv:2603.10052*, 2026.

[20] S. Lee, D. Kang, J. Park, and H.-W. Park, "DynaFlow: Dynamics-embedded flow matching for physically consistent motion generation from state-only demonstrations," *arXiv preprint arXiv:2509.19804*, 2025.

[21] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," *arXiv preprint arXiv:2409.01652*, 2024.

[22] T. Wang, E. Y. Puang, M. Lee, W. Jing, and Y. Wu, "End-to-end reinforcement learning of robotic manipulation with robust keypoints representation," in *2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2022, pp. 01–08.

[23] T. Zhang, Y. Hu, H. Cui, H. Zhao, and Y. Gao, "A universal semantic-geometric representation for robotic manipulation," *arXiv preprint arXiv:2306.10474*, 2023.

[24] T. Hoang, H. Le, P. Becker, V. A. Ngo, and G. Neumann, "Geometry-aware RL for manipulation of varying shapes and deformable objects," *arXiv preprint arXiv:2502.07005*, 2025.

[25] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[26] A. Bansal, H.-M. Chu, A. Schwarzschild, S. Sengupta, M. Goldblum, J. Geiping, and T. Goldstein, "Universal guidance for diffusion models," *arXiv preprint arXiv:2302.07121*, 2023.

[27] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with

diffusion for flexible behavior synthesis," *International Conference on Machine Learning (ICML)*, 2022.
[28] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby *et al.*, "DINOv2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.
[29] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," *International Conference on Machine Learning (ICML)*, 2021.
[30] 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.
[31] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.
[32] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "SAM 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.
[33] J. Kiefer, "Sequential minimax search for a maximum," *Proceedings of the American Mathematical Society*, vol. 4, no. 3, pp. 502–506, 1953.
[34] T. Tang, M. Liu, W. Xu, and C. Lu, "Kalib: Easy hand-eye calibration with reference point tracking," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, arXiv:2408.10562.

APPENDIX

This appendix collects the implementation-specific details of the pipeline summarized in Sections III–IV. The values reported below are those used for the experiments unless stated otherwise.

A. Perception pipeline

Object instance masks are produced by an open-vocabulary, grounded pipeline rather than a single segmentation model. For each task-relevant object named in the instruction, a VLM (Gemini 3 Flash [31]) proposes one grounded bounding box; SAM2 [32] converts each box into a precise mask; and the masks are size- and erosion-filtered before centroids and surface points are recovered by unprojecting the depth image and fusing the available camera views. After the first frame, masks are propagated across frames with the SAM2 video tracker so that per-step perception is inexpensive; full VLM–SAM2 detection is rerun only when re-grounding is needed. The same pipeline is applied to both the demonstration frame and the live OOD frame, giving consistent, language-grounded labels across scenes. Segmentation therefore comes from grounded VLM boxes refined by SAM2, rather than from a single concept-level segmentation model.

B. Demonstration analysis and VLM prompting

The entity pairs, relation types, and projections of Section III-D are selected by a separate reasoning VLM (GPT-5.4) queried in a single-turn prompt with the task description and the two annotated images (segmentation overlay and STRAP sub-trajectory overlay). An optional iterative variant (up to three rounds, selecting pairs by alignment consistency across demonstrations) is available but disabled in the reported runs. VLM outputs and live object detections are recomputed per episode (caching disabled) so that results reflect the full online pipeline. Figure 9 walks through the offline abstraction and its online application.

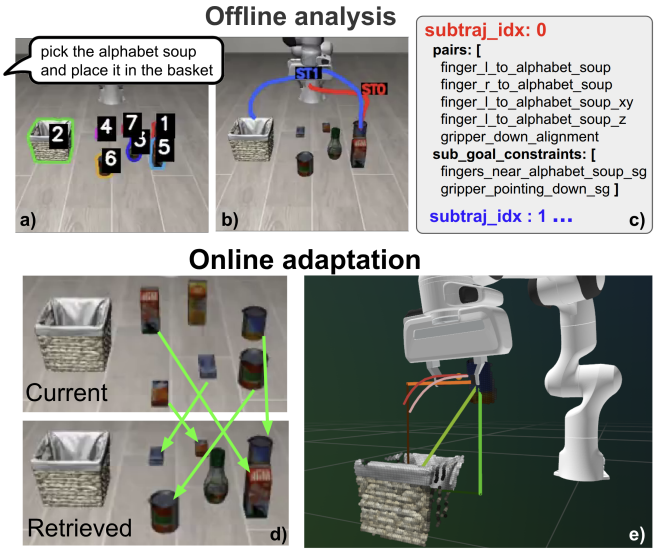


Fig. 9. Visual-geometric guidance via VLM-selected distance pairs. **Offline** (a–c): the demonstration is segmented (Gemini 3 Flash + SAM2) and split into sub-trajectory phases (STRAP); the VLM selects each phase's entity pairs, projections, and sub-goal constraints. **Online** (d–e): the live scene is matched to the demonstration, and the guidance steers the frozen policy so the selected projected distances track the reference.

C. Sub-trajectory segmentation

Following the velocity-based heuristic of STRAP [30], the demonstration is split at low end-effector speed points: we threshold the end-effector speed (computed from the end-effector position trajectory) at 0.005 and enforce a minimum sub-trajectory length of 20 steps to avoid spurious fragmentation.

D. Object-relational relations and per-relation weights

Each relation specifies a *source* and a *distance type*. Sources are the end-effector, the left or right fingertip (obtained by forward kinematics from the end-effector pose and gripper aperture), an object rigidly held by the gripper (placed at a body-frame offset from the end-effector), or, for orientation and gripper relations, the end-effector pose and gripper channel directly. Distance types are: **centroid** (Euclidean), **surface** (minimum distance from the source to a sub-sampled object point cloud), **orientation** (signed cosine of a rotated body-frame vector $R\hat{e}$ with an axis, or its in-plane magnitude w.r.t. a plane), and **gripper** (metric aperture). The gripper channel is mapped from the policy's normalized action range to metric finger separation so that finger-kinematics and aperture relations are all expressed in metres.

The per-relation weight w_n of Eq. 4 combines the VLM-assigned importance of relation n with a *relevance gate*: a relation is linearly ramped from inactive to fully active as the end-effector distance to the relevant object decreases from a far threshold ($0.15\,$m) to a near threshold ($0.05\,$m), so that a relation only steers the policy once the corresponding object is being approached. The reference distance trajectory $d_{t'}^{*n}$ is taken from the retrieved demonstration; when several demonstrations are available, their dynamic-time-warping-aligned

spread yields a per-relation, per-step standard deviation that is used for alignment and diagnostics (it does not reweight the loss in the reported configuration).

E. Trajectory alignment

Alignment runs online in the visual-geometric feature space. In the MIXED mode we use, a shared cursor is first matched to the reference by minimizing the mean squared error over the relevant relations, and then each relation is independently re-aligned within a window of half-width 5 frames around that cursor. The cursor is *not* monotonic: within the active sub-trajectory it may move forward or backward, so the guidance can track retries; per step it advances by at most 1.2 action chunks to avoid large jumps. Distances are normalized per sub-trajectory so matching is driven by relative progress, and the optional temporal-pace term is disabled. Sub-trajectory switching is bidirectional and gated on sub-goal satisfaction, with a persistence of 10 steps to reject transient signals. A softmax over each relation's alignment lag (temperature 5.0, floor 0.05) sets the per-relation temporal weighting that enters w_n.

F. Action roll-out (dynamics model)

The dynamics model $\mathcal{F}$ of Section III-D is a kinematic integrator of the OSC velocity actions. From the current end-effector position p_0 and axis-angle orientation r_0 (with $R_0 = \exp r_0$), it integrates the chunk's per-step linear and angular velocities (v_k, ω_k) with an Euler (cumulative-sum) scheme,

$$p_{t'} = p_0 + \sum_{k \le t'} v_k \, \Delta t, \qquad r_{t'} = r_0 + \sum_{k \le t'} \omega_k \, \Delta t, \qquad R_{t'} = \exp(r_{t'}), \tag{9}$$

where Δt is the per-step duration. The gripper channel is absolute and read directly, mapped to a metric aperture as in Appendix D. Because $\mathcal{F}$ is differentiable, the guidance gradient flows through it back to the action chunk; any other differentiable dynamics model can be substituted without changing the rest of the pipeline. In the real world the policy predicts joint positions, so $\mathcal{F}$ is the analytic UR5 forward kinematics.

G. Guidance injection and line search

Guidance is applied at every replanning call and every denoising step. The gradient is evaluated at the policy's tentative next iterate $A_t^{(k)} + \Delta t \, v$ over the first predicted action chunk, which holds the executed actions. Our $\pi_{0.5}$ checkpoint predicts two consecutive 10-step chunks per call; since it replans every 5 steps, the second, look-ahead chunk is never executed and is left unguided.

Gradient normalization. By default the gradient is ℓ_2-normalized to a unit direction, $\hat{d} = -g/\|g\|$, so the golden-section step size, not the raw gradient scale, sets the correction magnitude. This is the *gradient normalization* component of the ablation (Table III). Its ablated variants use the raw gradient $-g$ instead: the fixed-step row applies it with $c = 1$ and no line search, and the line-search row searches its step over $[0, 100]$ rather than $[0, 1]$, since the raw gradient is an order of magnitude or more smaller than the unit direction. The golden-section search (Algorithm 1) uses 20 narrowing iterations and expands its bracket by at most five doublings; the do-no-harm guard returns a zero step whenever no positive step reduces the loss.

Algorithm 1 Per-step golden-section line search for the guidance scale $\alpha^\star$.

Require: chunk $A_t^{(k)}$, unit direction $\hat{d}$, loss $\mathcal{L}_{guide}$, bound $\alpha_{\max}$, iterations n

define $f(\alpha) \equiv \mathcal{L}_{guide}(A_t^{(k)} + \alpha\,\hat{d})$, $\varphi \leftarrow (\sqrt{5}+1)/2$
$a \leftarrow 0$, $b \leftarrow \alpha_{\max}$
while $f(2b) < f(b)$ **do** ▷ expand bracket (capped at 5 doublings)
  $b \leftarrow 2b$
for n iterations **do** ▷ golden-section narrowing
  $u \leftarrow b - (b-a)/\varphi$, $w \leftarrow a + (b-a)/\varphi$
  if $f(u) < f(w)$ **then**
    $b \leftarrow w$
  else
    $a \leftarrow u$
$\alpha \leftarrow (a+b)/2$
return α **if** $f(\alpha) < f(0)$ **else** 0 ▷ do-no-harm

An optional multi-step variant decomposes the correction across relations or chunk timesteps: *sequential* (greedy coordinate descent over relations, line-searched against the total loss), *sum/weighted* (independent per-relation steps, optionally scaled by w_n), and *per-timestep* (the timestep analog of sequential). These are disabled in the reported runs, which use a single shared descent direction.

H. Hyperparameters and base policies

The global guidance scale is $c = 1.0$, and the relevance thresholds are $(0.05, 0.15)$ m across all benchmarks and base policies; only the line-search bound $\alpha_{\max}$ is rescaled per policy to match its action parameterization (velocity field versus score). Guidance is recomputed every 5 environment steps (the policy replanning interval). We evaluate $\pi_{0.5}$ (OpenPI checkpoint `pi05_libero_grounded`; 224×224 inputs) and GR00T N1.7 (`GR00T-N1.7-LIBERO`; 256×256 inputs), keeping each policy frozen. On LIBERO-Pro we run 10 trials per task across the object, spatial, language, and goal/task perturbation conditions. The retrieval encoders are DINOv2 (`dinov2-base`) for vision and CLIP (`clip-vit-base-patch16`) for language.

I. Full results breakdown

Table VI reports the per-perturbation-type success rates behind the task-and-positional summary in the main text, including the near-saturated object and language perturbations. The base policies already reach 75–95% on object and language shifts, so steering has little headroom there and can slightly perturb an already-correct trajectory; the gains concentrate on the task and positional shifts.

TABLE VI

PER-PERTURBATION-TYPE SUCCESS RATE (%) ON LIBERO-PRO, AVERAGED OVER THE FOUR SUITES. THE BASE IS NEAR-SATURATED ON OBJECT AND LANGUAGE, SO THE OVERALL AVERAGE UNDERSTATES THE TASK/POSITIONAL GAINS. "GT LABELS" / "GEMINI 3 FLASH" DENOTE THE PERCEPTION USED FOR OUR METHOD.

Method	Task	Pos	Object	Lang.	Overall
$\pi_{0.5}$					
No steering	26.3	26.3	83.0	92.3	56.9
VLS (reproduced, full)	32.0	27.6	45.8	61.5	41.7
VLS (grad-only)	17.3	21.5	18.3	26.8	20.9
Ours (GT labels)	40.8	34.3	65.8	75.0	53.9
Ours (Gemini 3 Flash)	24.8	24.5	47.5	55.3	38.0
GR00T N1.7					
No steering	19.3	0.5	75.5	90.5	46.4
Ours (GT labels)	20.0	3.5	60.8	75.8	40.0
Ours (Gemini 3 Flash)	14.5	3.5	44.0	57.0	29.8

TABLE VII

OPEN-VOCABULARY GROUNDING-BACKBONE ABLATION FOR $\pi_{0.5}$+OURS UNDER REAL PERCEPTION (SUCCESS RATE %; "POS" DENOTES THE POSITIONAL/SWAP PERTURBATION). GEMINI 3 FLASH OUTPERFORMS GEMINI ROBOTICS-ER 1.6 ON THE TASK AND POSITIONAL SUITES.

Suite	Pert.	ER-1.6	Gemini 3 Flash
Goal	Task	9.0	11.0
Goal	Pos	23.0	25.0
Spatial	Task	40.0	49.0
Spatial	Pos	33.0	32.0
libero_10	Task	5.0	6.0
libero_10	Pos	9.0	8.0
Object	Task	23.0	33.0
Object	Pos	27.0	33.0
Avg (task+pos)		21.1	**24.7**

J. Real open-vocabulary perception

Under real open-vocabulary perception (grounded VLM detection followed by SAM2 masks, with no simulator labels, unlike the ground-truth grounding used in the main results), $\pi_{0.5}$+Ours reaches 24.8 / 24.5 on the task / positional perturbations (Table VI), well below the 40.8 / 34.3 obtained with ground-truth grounding. This difference quantifies how much current performance is bottlenecked by perception rather than by the steering itself. Within real perception, the choice of grounding backbone matters: Table VII compares Gemini Robotics-ER 1.6 and Gemini 3 Flash, with Gemini 3 Flash performing best (24.7 vs. 21.1 averaged over the task and positional suites); we use it throughout.

K. Real-world platform details

Robot and control. A UR5 6-DoF arm with a Robotiq 2F-140 parallel gripper (0–140 mm stroke). The arm is commanded in joint space + gripper, matching the action space the base policies emit; the policy emits 50-step chunks replanned every 25 steps. The gripper aperture is mapped from the policy's normalized channel to metric finger separation as in Appendix D.

Cameras and calibration. All object detection and 3D reconstruction for the guidance are performed from a *single fixed external* Intel RealSense D435, mounted on the right side of the arm. Intrinsics are taken from the camera's factory calibration. Extrinsics are obtained *markerlessly* with Kalib [34], which tracks a reference point on the robot with a visual foundation model and recovers the camera-to-robot transform with a PnP solve from forward kinematics; no fiducial markers, mesh models or network training are required, and a full calibration takes seconds. This matters in practice because the guidance consumes metric object centroids and surface points unprojected from depth: a calibration error translates directly into a bias on every distance pair. Because only one view is available, the multi-view fusion described in Appendix A reduces to a single view, so object centroids are recovered from the visible surface alone and self-occluded geometry is not observed.

Base policy and reference rollout. The base policy is $\pi_{0.5}$ [2] (base), kept frozen. The demonstration database holds a single successful rollout of that same policy, recorded in the nominal configuration with the cable (Section VI). Guidance hyperparameters are unchanged from Appendix H.

L. Real-world perception timing

The simulation profiling of Figure 3 assumes privileged object states. On hardware, the guidance also pays for real open-vocabulary perception (Appendix A). We report it in two parts: a once-per-episode first-frame setup, and the per-replan cost inside the inference loop.

First-frame setup. Table VIII breaks down the setup over 6 runs on the same scene with a fresh VLM cache each run. The six VLM round-trips sum to 20.6 ± 3.1 s, so roughly 70% of the 29.5 ± 3.1 s total is API latency; live-object inference is the noisiest call. Process startup before the episode is separate (retrieval encoders 1.6 ± 0.6 s, perception clients 3.4 ± 0.4 s).

TABLE VIII

FIRST-FRAME SETUP, MEAN ± SD OVER 6 RUNS.

Stage	Wall time
Retrieval (DINOv2 + CLIP embed and match)	0.43 ± 0.01 s
Demo seed frame: object names (VLM)	2.1 ± 0.3 s
Demo seed frame: box detection (VLM)	2.1 ± 0.4 s
Demo seed frame: SAM2 masks	0.16 ± 0.14 s
Pair and constraint selection (VLM)	9.4 ± 0.6 s
Demo analysis total	14.9 ± 0.8 s
Reference distance trajectory (57 frames)	3.2 ± 0.4 s
Live-object inference (VLM)	3.0 ± 1.7 s
Live seed-frame detection (boxes + SAM2)	2.6 ± 0.5 s
Equivalence mapping (VLM)	3.1 ± 0.4 s
Total	$\mathbf{29.5 \pm 3.1}$ **s**

Inference loop. We replayed detect-then-track over the primary view of six recorded guided rollouts at the 25-step replan cadence, using each rollout's own object names; every object was found on every frame (Table IX). Full detection runs once per episode; afterwards the loop pays only mask propagation and back-projection, 113 ± 2 ms per replan.

M. Positional OOD analysis

Training distribution. The 28 training demonstrations of "Put all cables in the bag." yield 110 cable positions, one per

TABLE IX
PER-REPLAN PERCEPTION COST IN THE INFERENCE LOOP.

Measurement	Value
Detection, once per episode (VLM boxes + SAM2)	2.6 ± 0.6 s
Track call per replan (replay, 63 frames)	46 ± 4 ms
Tracker step alone (replay)	37.5 ± 3.5 ms
Tracker step alone (in the loop, 80 rollouts)	48 ± 14 ms
Perception block in the live loop	$\mathbf{113 \pm 2}$ **ms**

gripper closure near the table, read from the demonstrations' proprioception in the robot base frame; these are grasp points and therefore a proxy for where the cable lay. On the table they form a weakly correlated Gaussian ($\rho = -0.07$) centered in cell B2 of the 4×4 evaluation grid, with a standard deviation of 6.2 cm across columns and 5.6 cm across rows. Each experiment rollout's cable position, measured from its first frame with the perception pipeline of Appendix A, is expressed as a Mahalanobis distance from that Gaussian, in standard deviations. The two sets of positions therefore reach us through different channels, demonstration proprioception at gripper closure against perception on the first frame, so the distances carry the offset between them. A systematic offset would shift all reported distances by a common amount; it would not change the ordering of the cells or their zone assignment, which follow from the grid geometry.

Statistics. We analyze success against that distance in two ways: binned into three zones, with 95% Wilson score intervals per method and a two-sided Fisher exact test between methods per zone (Table IV), and as a logistic regression of success on distance fitted per method, whose 50% crossing we report (Table XII). Intervals on the difference between methods use Newcombe's hybrid score method: $[+7, +54]$ points in the near-OOD band, $[+2, +41]$ pooled over OOD and $[-1, +37]$ over the full sweep. Fisher's exact test is conservative on small, discrete samples, which is why the pooled OOD comparison gives $p = 0.051$ while the corresponding interval excludes zero. Both procedures also treat the 48 rollouts per method as independent, which they are not: each placement is run by both methods, a paired design these unpaired tests ignore, and the three trials in a cell share a nominal placement. We therefore report all of these as descriptive summaries rather than as inference over independent trials.

Zones. We assign each rollout to a zone from its own measured position rather than from its nominal cell (Table X). Both boundaries lie in gaps of the distance distribution: no rollout falls between 1.4 and 2.3σ or between 4.1 and 5.3σ, so any cut placed inside a gap yields the same assignment, and the two rollouts of a placement are assigned together unless their measured positions differ by more than the 0.9σ gap width. Table XI lists the distance of every cell center, and Figure 4 shows the per-cell outcomes projected into the camera view.

TABLE X
ZONE DEFINITION FOR THE POSITIONAL SWEEP, BY MAHALANOBIS DISTANCE OF THE MEASURED CABLE POSITION FROM THE TRAINING POSITIONS.

Zone	Distance	Cells	Rollouts per method
In-distribution	below 2.0σ	B1, B2, B3	9
Near-OOD	2.0 to 4.5σ	A1 to A4, B4, C1 to C4	27
Far-OOD	above 4.5σ	D1 to D4	12

TABLE XI
CELL CENTERS OF THE 4×4 GRID IN THE TRAINING DISTRIBUTION'S STANDARDIZED COORDINATES AND AS A MAHALANOBIS DISTANCE, SORTED BY DISTANCE. THE DISTANCE USES THE FULL COVARIANCE, WHICH IS WEAKLY CORRELATED, SO IT IS NOT THE EUCLIDEAN NORM OF THE TWO z COLUMNS. CENTERS ARE LISTED FOR REFERENCE ONLY; ZONE ASSIGNMENT USES EACH ROLLOUT'S OWN MEASURED POSITION.

Cell	z columns	z rows	Mahalanobis	Zone
B2	-0.13	-0.26	0.30	in-distribution
B3	1.09	-0.26	1.10	in-distribution
B1	-1.35	-0.26	1.39	in-distribution
B4	2.30	-0.26	2.30	near-OOD
C2	-0.13	2.44	2.44	near-OOD
C1	-1.35	2.44	2.71	near-OOD
C3	1.09	2.44	2.74	near-OOD
A2	-0.13	-2.95	2.97	near-OOD
A3	1.09	-2.95	3.08	near-OOD
A1	-1.35	-2.95	3.33	near-OOD
C4	2.30	2.44	3.47	near-OOD
A4	2.30	-2.95	3.62	near-OOD
D1 to D4	-1.35 to 2.30	5.13	5.1 to 5.8	far-OOD

N. VLS reproduction

The VLS paper reports 38.5 / 35.1 (36.8 average) on the task / positional perturbations; our reproduced full run reaches 32.0 / 27.6. The configuration matches the paper's hyperparameters (batch 20, 50 denoising steps, 200 episodes per suite, guidance scale, and resampling/diversity settings) and the same model families (GPT-5.1 guidance, Gemini 3 Flash grounding), so the residual gap is attributable to (i) a different, re-exported base checkpoint (`pi05_libero_finetuned_v044`); (ii) routing all VLM calls through OpenRouter rather than the native OpenAI/-Google APIs; and (iii) single-run stochasticity (guidance code sampled at temperature 1.0, live stage decisions, unpinned seeds) together with provider model drift since the paper's runs.

The *grad-only* VLS row instead runs the public repository defaults, which set `sample_batch_size=` 1. With a single particle, VLS's Feynman–Kac resampling and RBF diversity initialization are both disabled, so the method degenerates to pure gradient steering (the paper's "w/o FKD, w/o RBF" ablation), and it uses 10 denoising steps. Because it shares the same grounding and a comparable compute budget, this row isolates the gradient-steering signal and is the appropriate point of comparison for our single-chain gradient guidance.

TABLE XII

LOGISTIC FIT OF SUCCESS AGAINST MAHALANOBIS DISTANCE, PER METHOD. THE 50% CROSSING IS ANNOTATED IN FIGURE 5; THE SLOPE IS THE LOGIT DECREASE PER STANDARD DEVIATION. BOTH FITS ARE CONSTRAINED ALMOST ENTIRELY BY THE NEAR-OOD TRIALS, SINCE THE OTHER TWO ZONES ARE SATURATED FOR BOTH METHODS, AND NEITHER IS SUPPORTED BY DATA BEYOND 4.5σ.

Logistic fit	Base (no steering)	Ours
50% crossing	2.8σ	3.4σ
Slope	3.9 per σ	1.7 per σ

O. Worked example: offline demonstration analysis

This subsection reproduces a complete offline demonstration-analysis query end to end, for the task *"pick up the alphabet soup and place it in the basket"*. The reasoning VLM (Section B) is given the system prompt below, together with the user message: the task description, the object list, the STRAP sub-trajectory split, and the three annotated images of Figure 10. It returns the structured set of entity pairs, projections, and per-sub-trajectory sub-goal constraints shown in the assistant response, which the pipeline then compiles into the visual-geometric relations used for steering. The full message blocks are reproduced verbatim below.

a) System message.:

```
You are a robotics expert. Given a manipulation task description, a scene image (original RGB), a
     segmentation-overlay image that labels every object with its numerical instance ID, a sub-trajectory overlay
     image showing the robot end-effector path split into numbered sub-trajectories (each drawn in a distinct bright
     colour and labelled ST0, ST1, ...), and a list of objects (segmentation ID + name), select the most informative
     pairs of entities whose pairwise distance serves as a useful heuristic for guiding task execution.

The special entity "ee" refers to the robot end-effector. The entities "gripper_l" and "gripper_r" refer to the left
     and right gripper finger tips respectively. Prefer "gripper_l" and "gripper_r" for any task that involves
     grasping, manipulating, or releasing objects with the fingers --- they capture finger-level proximity (e.g. each
     finger to the target object) and gripper aperture ("gripper_l" to "gripper_r"). Only fall back to "ee" for tasks
     that do not involve the gripper fingers (e.g. pushing, pointing, or tool use where the end-effector as a whole
     is the relevant contact point).

For each pair choose the most appropriate distance type:
  - "centroid": Euclidean distance between the 3-D centroids of the two entities.
  - "surface": Shortest distance between the surfaces of the two entities (useful when contact/proximity matters).
  - "orientation": Projection of a body-frame vector of the end-effector onto a world-space reference direction. By
     default the tracked vector is the gripper's body z-axis (its "approach direction"), but you can set "ee_vector"
     to any body-frame direction (e.g. [1,0,0] for the gripper x-axis). Use when the orientation of the end-effector
     matters --- e.g. ensuring the gripper approaches a target from above rather than from the side. Both
     "from_entity" and "to_entity" must be "ee" (the pair self-references). Requires either "axis" or "plane" (see
     below).
  - "gripper_aperture": Scalar gripper opening (metres). Both entities must be the gripper fingers --- set
     "from_entity": "gripper_l" and "to_entity": "gripper_r".

Additionally, any pair can include at most ONE of the following optional component selectors to restrict the
     measurement to a subspace:
  - "axis" [x,y,z]: For positional pairs (centroid/surface), signed 1-D projection of the distance vector onto this
     direction --- e.g. axis=[0,0,1] for vertical clearance during a lift, axis=[1,0,0] for progress along X during
     an approach. For "orientation" pairs, signed cosine of the angle between the tracked body vector and this
     direction (in [-1, 1]) --- e.g. axis=[0,0,-1] gives +1 when the body vector points straight down ("approach from
     above") and -1 when it points straight up.
  - "plane" [nx,ny,nz]: For positional pairs, magnitude of the in-plane component after removing the contribution
     along this normal (2-D). Use for lateral alignment --- e.g. plane=[0,0,1] gives XY-plane distance. For
     "orientation" pairs, magnitude of the tracked body vector projected into the plane perpendicular to the normal
     (in [0, 1]) --- e.g. plane=[0,0,1] is 0 when the body vector is exactly vertical and 1 when it is fully
     horizontal.
  - "ee_vector" [x,y,z]: Only for "orientation" pairs. Specifies which body-frame direction of the end-effector to
     track. Defaults to [0,0,1] (z-axis, the "approach direction"). Set to [1,0,0] to track the gripper x-axis,
     [0,1,0] for the y-axis, or any arbitrary body-frame direction. Useful when the task cares about a sideways
     alignment rather than approach direction --- e.g. pouring, where the gripper's x-axis must point away from the
     target container, or wiping, where the gripper's y-axis must remain aligned with the travel direction.

Set both "axis" and "plane" to null when the full 3-D distance is the right signal (positional pairs only ---
     "orientation" pairs always need one).

Group your output by sub-trajectory. For each sub-trajectory phase, provide a single "reasoning" string (1-2
     sentences) that describes the goal of that phase and why the selected pairs are informative for it. Use
     subtraj_idx null for pairs that are equally relevant across all sub-trajectories.

For each sub-trajectory you must also define sub-goal constraints:

SUB-GOAL CONSTRAINTS ("sub_goal"):
  Conditions that must be satisfied at the END of the sub-trajectory. They determine when to transition to the next
     sub-trajectory. Example: the end-effector must be within 3 cm of the mug before the grasp phase ends.

  IMPORTANT: Sub-goal constraints must depend ONLY on the CURRENT state of the objects and end-effector --- they may
     compare positions of entities visible in the scene at the moment of evaluation, but must NOT reference past
     positions, initial positions, or any temporal history. If you need a height reference, use a currently visible
     entity (e.g. the table, a container, or another scene object), not the object's own past position.

Each constraint is expressed as a Python lambda string with the signature:
  lambda ee, keypoints, ee_R: <scalar expression>
where "ee" is a numpy array of shape (3,) with the end-effector XYZ position, "keypoints" is a dict mapping entity
     IDs (int or "ee") to numpy arrays of shape (3,) with their 3-D centroid positions, and "ee_R" is the
     world-from-EE rotation matrix as a numpy array of shape (3, 3). Within "ee_R", the third column "ee_R[:, 2]" is
     the gripper's body z-axis ("approach direction") expressed in world coordinates --- use it to write orientation
     constraints such as "the gripper must point downward at end of approach". "ee_R[:, 0]" and "ee_R[:, 1]" are the
     other two body axes if a yaw-style constraint is needed. The constraint is SATISFIED when the returned scalar is
     <= 0. Use np for numpy operations. Entity IDs in keypoints match the segmentation IDs in the object list; "ee"
     maps to the end-effector centroid.
```

```
The lambda is evaluated in a restricted sandbox where only the "np" (numpy) module is available. Python built-ins
     such as max/min/abs/sum are NOT accessible --- use numpy equivalents (np.maximum, np.minimum, np.abs, np.sum)
     instead. The lambda must return a scalar (a Python float or 0-D numpy value).

Example 1 --- pick up a mug (2 sub-trajectories: approach ST0, lift ST1):
{
  "subtraj_idx": 0,
  "reasoning": "During the approach phase the fingers must close in on the mug, so the surface distances from each
     finger to the mug are the primary signals. The gripper must also approach from above (z-axis pointing down),
     maintain vertical clearance, and keep the fingers horizontally aligned with the handle.",
  "pairs": [
    {"from_entity": "gripper_l", "to_entity": 2,    "label": "finger_l_to_mug",       "distance_type": "surface",
     "axis": null,        "plane": null},
    {"from_entity": "gripper_r", "to_entity": 2,    "label": "finger_r_to_mug",       "distance_type": "surface",
     "axis": null,        "plane": null},
    {"from_entity": "gripper_l", "to_entity": 2,    "label": "finger_l_to_mug_z",     "distance_type": "centroid",
     "axis": [0, 0, 1],   "plane": null},
    {"from_entity": "ee",        "to_entity": "ee", "label": "gripper_down_alignment", "distance_type":
     "orientation", "axis": [0, 0, -1], "plane": null, "ee_vector": null},
    {"from_entity": "ee",        "to_entity": "ee", "label": "gripper_side_alignment", "distance_type":
     "orientation", "axis": [0, 0, 1],  "plane": null, "ee_vector": [0, 1, 0]}
  ],
  "sub_goal_constraints": [
    {
      "label": "fingers_near_mug_sg",
      "description": "Both finger tips must be within 8 cm of the mug centroid at the end of approach.",
      "expression": "lambda ee, keypoints, ee_R: np.maximum(np.linalg.norm(keypoints['gripper_l'] - keypoints[2]),
     np.linalg.norm(keypoints['gripper_r'] - keypoints[2])) - 0.08",
      "reasoning": "Ensures the fingers have reached grasping distance before transitioning to the lift phase."
    },
    {
      "label": "gripper_pointing_down_sg",
      "description": "Gripper z-axis must be within ~30° of straight-down at the end of approach (cos(30°) ≈ 0.85).",
      "expression": "lambda ee, keypoints, ee_R: 0.85 - np.dot(ee_R[:, 2], np.array([0.0, 0.0, -1.0]))",
      "reasoning": "Confirms the gripper is well-aligned to a top-down grasp before the lift phase begins."
    }
  ]
},
{
  "subtraj_idx": 1,
  "reasoning": "The lift phase raises the grasped mug off the table; the Z-axis projection captures vertical progress
     while ignoring lateral offset.",
  "pairs": [
    {"from_entity": 2, "to_entity": 1, "label": "mug_table_z", "distance_type": "centroid", "axis": [0, 0, 1],
     "plane": null}
  ],
  "sub_goal_constraints": [
    {
      "label": "mug_lifted_sg",
      "description": "Mug centroid must be at least 10 cm above the table centroid.",
      "expression": "lambda ee, keypoints, ee_R: 0.10 - (keypoints[2][2] - keypoints[1][2])",
      "reasoning": "Confirms the mug has been successfully lifted before the episode ends."
    }
  ]
}

Example 2 --- Pick the object and place it in the container (2 sub-trajectories: approach ST0, transport ST1):
{
  "subtraj_idx": 0,
  "reasoning": "ST0 is the approach-and-grasp phase for the object. The most informative signals are the
     finger-to-object surface distances and maintaining horizontal alignment between the fingers and the object.",
  "pairs": [
    {"from_entity": "gripper_l", "to_entity": 1,           "label": "finger_l_to_object",    "distance_type":
     "surface",  "axis": null, "plane": null},
    {"from_entity": "gripper_r", "to_entity": 1,           "label": "finger_r_to_object",    "distance_type":
     "surface",  "axis": null, "plane": null},
    {"from_entity": "gripper_l", "to_entity": 1,           "label": "finger_l_to_object_xy", "distance_type":
     "centroid", "axis": null, "plane": [0, 0, 1]}
  ],
  "sub_goal_constraints": [
    {
      "label": "fingers_near_object_sg",
      "description": "Both finger tips must be within 7 cm of the object centroid at the end of the approach phase.",
      "expression": "lambda ee, keypoints, ee_R: np.maximum(np.linalg.norm(keypoints['gripper_l'] - keypoints[1]),
     np.linalg.norm(keypoints['gripper_r'] - keypoints[1])) - 0.07",
      "reasoning": "This indicates the fingers have reached grasping proximity to the object before transitioning to
     transport."
    }
  ]
},
{
  "subtraj_idx": 1,
  "reasoning": "ST1 transports the grasped object and lowers it into the container. The key signals are horizontal
     alignment between object and container and then proximity to the container interior while maintaining enough
     height to avoid premature contact with the container rim.",
```

```
  "pairs": [
    {"from_entity": 1, "to_entity": 2, "label": "object_to_container_xy", "distance_type": "centroid", "axis": null,
        "plane": [0, 0, 1]},
    {"from_entity": 1, "to_entity": 2, "label": "object_to_container",    "distance_type": "centroid", "axis": null,
        "plane": null},
    {"from_entity": 1, "to_entity": 2, "label": "object_to_container_z",  "distance_type": "centroid", "axis": [0, 0,
     1], "plane": null}
  ],
  "sub_goal_constraints": [
    {
      "label": "object_in_container_xy_sg",
      "description": "Object centroid must be horizontally within 6 cm of the container centroid at the end of
     placement.",
      "expression": "lambda ee, keypoints, ee_R: np.linalg.norm((keypoints[1] - keypoints[2])[:2]) - 0.06",
      "reasoning": "This ensures the object is laterally aligned with the container opening at the end of the
     placement motion."
    },
    {
      "label": "object_lowered_into_container_sg",
      "description": "Object centroid must be no more than 8 cm above the container centroid at the end of
     placement.",
      "expression": "lambda ee, keypoints, ee_R: (keypoints[1][2] - keypoints[2][2]) - 0.08",
      "reasoning": "This confirms the object has been lowered into the container region rather than remaining high
     above it."
    }
  ]
}

Example 3 --- Grasp a small object and transport it to a target zone (3 sub-trajectories: approach ST0,
     grasp-and-lift ST1, transport ST2):
{
  "subtraj_idx": 0,
  "reasoning": "ST0 is the approach phase: the fingers must close in on the object with good horizontal alignment
     before grasping.",
  "pairs": [
    {"from_entity": "gripper_l", "to_entity": 1, "label": "finger_l_to_object",    "distance_type": "surface",
     "axis": null, "plane": null},
    {"from_entity": "gripper_r", "to_entity": 1, "label": "finger_r_to_object",    "distance_type": "surface",
     "axis": null, "plane": null},
    {"from_entity": "gripper_l", "to_entity": 1, "label": "finger_l_to_object_xy", "distance_type": "centroid",
     "axis": null, "plane": [0, 0, 1]}
  ],
  "sub_goal_constraints": [
    {
      "label": "fingers_near_object_sg",
      "description": "Both finger tips must be within 7 cm of the object centroid.",
      "expression": "lambda ee, keypoints, ee_R: np.maximum(np.linalg.norm(keypoints['gripper_l'] - keypoints[1]),
     np.linalg.norm(keypoints['gripper_r'] - keypoints[1])) - 0.07",
      "reasoning": "The fingers have reached grasping proximity."
    }
  ]
},
{
  "subtraj_idx": 1,
  "reasoning": "ST1 is the grasp-and-lift phase. The gripper fingers must close around the object and then lift it.
     Monitoring finger-to-object surface distances confirms the fingers are positioned around the object; the gripper
     aperture tracks finger closure.",
  "pairs": [
    {"from_entity": "gripper_l", "to_entity": 1,           "label": "finger_l_to_object",  "distance_type":
     "surface",  "axis": null, "plane": null},
    {"from_entity": "gripper_r", "to_entity": 1,           "label": "finger_r_to_object",  "distance_type":
     "surface",  "axis": null, "plane": null},
    {"from_entity": "gripper_l", "to_entity": "gripper_r", "label": "gripper_aperture",    "distance_type":
     "centroid", "axis": null, "plane": null}
  ],
  "sub_goal_constraints": [
    {
      "label": "object_lifted_sg",
      "description": "Object centroid must be at least 5 cm above the table (entity 3) centroid.",
      "expression": "lambda ee, keypoints, ee_R: 0.05 - (keypoints[1][2] - keypoints[3][2])",
      "reasoning": "Confirms the object has been lifted off the table by comparing against the current table height."
    }
  ]
},
{
  "subtraj_idx": 2,
  "reasoning": "ST2 transports the grasped object to the target zone. The gripper must remain closed throughout
     transport to avoid dropping the object.",
  "pairs": [
    {"from_entity": 1, "to_entity": 2, "label": "object_to_target_xy", "distance_type": "centroid", "axis": null,
     "plane": [0, 0, 1]},
    {"from_entity": 1, "to_entity": 2, "label": "object_to_target",    "distance_type": "centroid", "axis": null,
     "plane": null}
  ],
  "sub_goal_constraints": [
```

```
    {
      "label": "object_at_target_sg",
      "description": "Object centroid must be within 5 cm of the target zone.",
      "expression": "lambda ee, keypoints, ee_R: np.linalg.norm(keypoints[1] - keypoints[2]) - 0.05",
      "reasoning": "Object has reached the target location."
    }
  ]
}

Return JSON with the following schema:
{
  "subtrajectories": [
    {
      "subtraj_idx": <int or null>,
      "reasoning": <string>,
      "pairs": [
        {
          "from_entity": <int or "ee" or "gripper_l" or "gripper_r">,
          "to_entity": <int or "ee" or "gripper_l" or "gripper_r">,
          "label": <string>,
          "distance_type": <"centroid"|"surface"|"orientation"|"gripper_aperture">,
          "axis": <[x,y,z] or null>,
          "plane": <[nx,ny,nz] or null>,
          "ee_vector": <[x,y,z] or null>
        },
        ...
      ],
      "sub_goal_constraints": [
        {
          "label": <string>,
          "description": <string>,
          "expression": <python lambda string>,
          "reasoning": <string>
        },
        ...
      ]
    },
    ...
  ]
}

Only include clearly relevant pairs and constraints for each specified sub-trajectory.
```

b) User message.:

```
Task: pick up the alphabet soup and place it in the basket

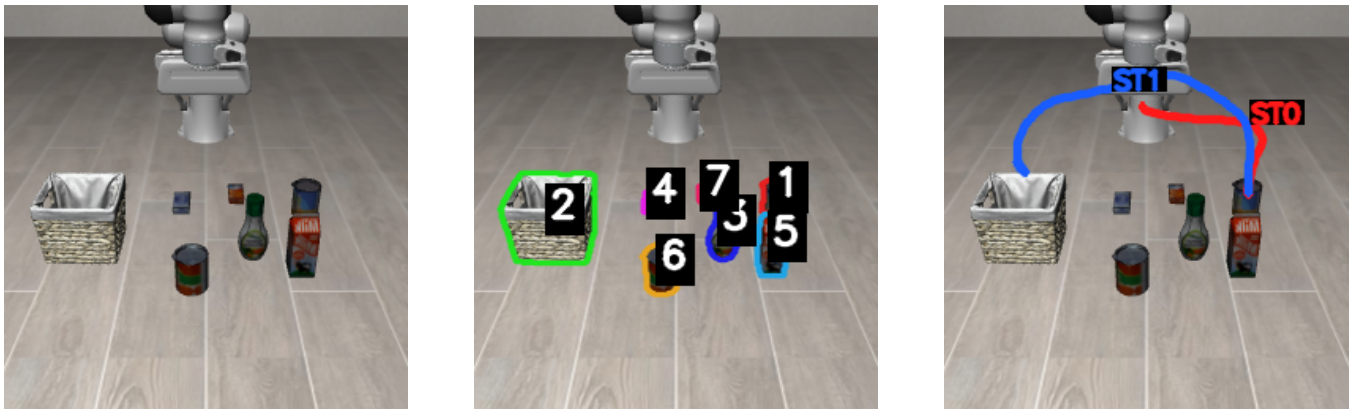
Available objects (id: name):
  1: alphabet_soup_1
  2: basket_1
  3: salad_dressing_1
  4: cream_cheese_1
  5: milk_1
  6: tomato_sauce_1
  7: butter_1

Number of sub-trajectories: 2
  ST0: timesteps 0-61
  ST1: timesteps 62-163
```

The three images below accompany the user message above.

c) Assistant response.:

```
{
  "subtrajectories": [
    {
      "subtraj_idx": 0,
      "reasoning": "ST0 is the approach-and-grasp phase for the alphabet soup can. The most informative signals are
     the distances from each gripper finger to the can, the lateral alignment of the gripper with the can, and the
     gripper orientation for a top-down grasp.",
      "pairs": [
        {
          "from_entity": "gripper_l",
          "to_entity": 1,
          "label": "finger_l_to_alphabet_soup",
          "distance_type": "surface",
          "axis": null,
          "plane": null,
          "ee_vector": null
        },
        {
          "from_entity": "gripper_r",
          "to_entity": 1,
          "label": "finger_r_to_alphabet_soup",
          "distance_type": "surface",
          "axis": null,
```



(a) Original RGB scene.

(b) Segmentation overlay with numeric instance IDs.

(c) STRAP sub-trajectory overlay (ST0 approach, ST1 transport).

Fig. 10. The three images provided to the reasoning VLM for the worked example of Appendix O: (a) the raw RGB observation, (b) the segmentation overlay that labels every object with its instance ID, and (c) the end-effector path split into color-coded numbered sub-trajectories.


```
      "plane": null,
      "ee_vector": null
    },
    {
      "from_entity": "gripper_l",
      "to_entity": 1,
      "label": "finger_l_to_alphabet_soup_xy",
      "distance_type": "centroid",
      "axis": null,
      "plane": [
        0,
        0,
        1
      ],
      "ee_vector": null
    },
    {
      "from_entity": "gripper_l",
      "to_entity": 1,
      "label": "finger_l_to_alphabet_soup_z",
      "distance_type": "centroid",
      "axis": [
        0,
        0,
        1
      ],
      "plane": null,
      "ee_vector": null
    },
    {
      "from_entity": "ee",
      "to_entity": "ee",
      "label": "gripper_down_alignment",
      "distance_type": "orientation",
      "axis": [
        0,
        0,
        -1
      ],
      "plane": null,
      "ee_vector": null
    }
  ],
  "sub_goal_constraints": [
    {
      "label": "fingers_near_alphabet_soup_sg",
      "description": "Both finger tips must be within 8 cm of the alphabet soup centroid at the end of the grasp
approach.",
      "expression": "lambda ee, keypoints, ee_R: np.maximum(np.linalg.norm(keypoints['gripper_l'] -
keypoints[1]), np.linalg.norm(keypoints['gripper_r'] - keypoints[1])) - 0.08",
      "reasoning": "This ensures the gripper has reached grasping proximity to the can before transitioning to
transport."
    },
    {
      "label": "gripper_pointing_down_sg",
      "description": "The gripper approach direction must be within about 30 degrees of straight down at the end
of ST0.",
```

```
          "expression": "lambda ee, keypoints, ee_R: 0.85 - np.dot(ee_R[:, 2], np.array([0.0, 0.0, -1.0]))",
          "reasoning": "A downward-facing gripper is the appropriate orientation for securely grasping the upright
     can from above."
        }
      ]
    },
    {
      "subtraj_idx": 1,
      "reasoning": "ST1 transports the grasped alphabet soup to the basket and lowers it into the basket region. The
     key signals are the can-to-basket lateral alignment, overall proximity to the basket, vertical relation to the
     basket, and maintaining a closed grasp during transport.",
      "pairs": [
        {
          "from_entity": 1,
          "to_entity": 2,
          "label": "alphabet_soup_to_basket_xy",
          "distance_type": "centroid",
          "axis": null,
          "plane": [
            0,
            0,
            1
          ],
          "ee_vector": null
        },
        {
          "from_entity": 1,
          "to_entity": 2,
          "label": "alphabet_soup_to_basket",
          "distance_type": "centroid",
          "axis": null,
          "plane": null,
          "ee_vector": null
        },
        {
          "from_entity": 1,
          "to_entity": 2,
          "label": "alphabet_soup_to_basket_z",
          "distance_type": "centroid",
          "axis": [
            0,
            0,
            1
          ],
          "plane": null,
          "ee_vector": null
        },
        {
          "from_entity": "gripper_l",
          "to_entity": "gripper_r",
          "label": "gripper_aperture",
          "distance_type": "gripper_aperture",
          "axis": null,
          "plane": null,
          "ee_vector": null
        }
      ],
      "sub_goal_constraints": [
        {
          "label": "alphabet_soup_aligned_with_basket_sg",
          "description": "The alphabet soup centroid must be horizontally within 10 cm of the basket centroid at the
     end of placement.",
          "expression": "lambda ee, keypoints, ee_R: np.linalg.norm((keypoints[1] - keypoints[2])[:2]) - 0.10",
          "reasoning": "This ensures the can is positioned over the basket opening rather than beside it."
        },
        {
          "label": "alphabet_soup_lowered_to_basket_sg",
          "description": "The alphabet soup centroid must be no more than 12 cm above the basket centroid at the end
     of placement.",
          "expression": "lambda ee, keypoints, ee_R: (keypoints[1][2] - keypoints[2][2]) - 0.12",
          "reasoning": "This confirms the can has been lowered into the basket region instead of remaining high above
     it."
        }
      ]
    }
  ]
}
```