Think Proprioceptively: State-Grounded Visual Token Selection for VLA Policies

Fangyuan Wang[1,2] Peng Zhou[3] Jiaming Qi[4] Shipeng Lyu[1,2]
Chengyang He[5] David Navarro-Alarcon[1,*] Guodong Guo[2,*]
[1]The Hong Kong Polytechnic University [2]Eastern Institute of Technology
[3]Great Bay University [4]Northeast Forestry University [5]National University of Singapore

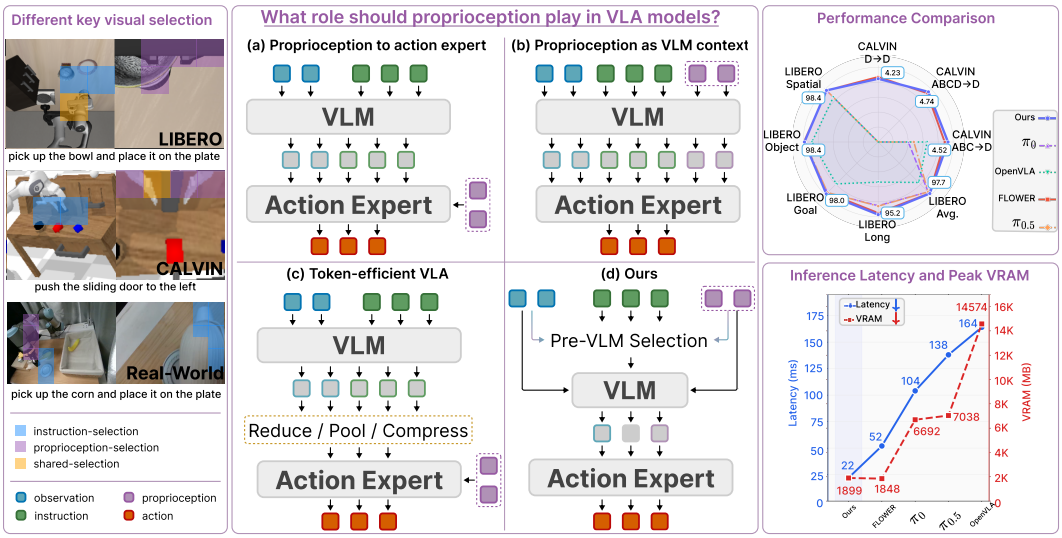


Figure 1: ThinkProprio investigates what role proprioception should play in VLA models. **Left:** Across simulation, and real-world tasks, instruction selection (blue) and state selection (purple) highlight complementary visual evidence. **Middle:** Four representative VLA designs: (a) proprioception as late input to the action expert, (b) proprioception as VLM context tokens, (c) generic token compression before the action expert, and (d) ThinkProprio uses both language and proprioception to gate visual tokens *before* VLM. **Right:** ThinkProprio matches strong baselines across CALVIN and LIBERO while achieving the lowest inference latency with a compact visual-token budget.

Abstract: Vision-language-action (VLA) models typically inject proprioception only as a late conditioning signal, preventing robot state from grounding instruction understanding or directing visual attention. We introduce ThinkProprio, which discretizes proprioception into VLM-vocabulary tokens and uses them jointly with the instruction to gate visual patches before VLM computation, steering the model toward action-relevant evidence while discarding redundant tokens early. We find that proprioception added as a passive conditioning signal leaves performance essentially unchanged; its value emerges when token-form state acts as an active query that, with the instruction, selects which visual patches the VLM processes. Systematic ablations show that VLM-vocabulary tokens outperform learned projectors as the state encoding, and that retaining only about 12 % of the visual tokens surpasses on CALVIN ABC→D. Across CALVIN, LIBERO, and real-world manipulation, ThinkProprio reduces end-to-end inference latency while improving the matched full-token baseline.

Keywords: Robot manipulation, Vision-language-action models

*Corresponding authors.

1 Introduction

Vision-language-action (VLA) models translate visual observations and language instructions into executable actions through large-scale pretraining [1, 2, 3]. Yet contact-rich manipulation depends not only on what is visible and what is requested, but also on the robot's embodiment, including its joint configuration and motion. Most current VLA pipelines treat proprioception as a late conditioning signal for the action head, couple it only weakly to perception, or omit it altogether. This raises a concrete design question: *should robot state be a late conditioning signal for action generation, or an active participant in instruction grounding and visual attention?*

Holding the backbone, action head, data, and training budget fixed, we vary only how proprioception is encoded and where it enters, finetuning on CALVIN ABC→D (Table 7). Passive late conditioning at the action head matches omitting state entirely, and an MLP projection into the VLM hurts; only VLM-vocabulary tokens routed through the VLM improve over the no-proprioception baseline.

In published systems, proprioceptive design is typically entangled with vision-language aggregation and the action conditioning mechanism (see Table 1), making the functional role of proprioception hard to attribute. We therefore vary these axes systematically. Prior work treats efficient visual-token selection [11, 12] and routing proprioception through the VLM token interface [4] as separate developments; once proprioception lives in this token space, it can also serve as a query signal for visual evidence, complementary to the instruction.

Table 1: Taxonomy of representative VLA designs. Subheaders specify the order of slash-separated fields. Tok. = tokenized and Comp. = compressed.

Model	**VL** repr. / act. cond.	**Proprio.** enc. / entry / act. cond.
π_0 [3]	Dense / Cross-attn	MLP / ACT / Cross-attn
$\pi_{0.5}$ [4]	Dense / Cross-attn	Tok. / VLM / Cross-attn
SmolVLA [5]	Dense / Cross-attn	MLP / VLM / Cross-attn
FLOWER [6]	Dense / Cross-attn	MLP / ACT / AdaLN
Dita [7]	Dense / In-context	MLP / ACT / In-context
CogACT [8]	Comp. / In-context	N/A
OTTER [9]	Pooled / Cross-attn	MLP / ACT / Cross-attn
DiT-Blocks [10]	Pooled / AdaLN	MLP / VLM / Cross-attn

We instantiate this idea as ThinkProprio, which gates visual patches before VLM computation using separate instruction and proprioception branches. Our contributions are:

- A systematic study isolating proprioceptive encoding and entry point from vision-language aggregation and action conditioning, showing that encoding proprioception as VLM-vocabulary tokens both preserves baseline performance and exposes state to downstream visual reasoning.
- ThinkProprio, a state-grounded visual-token gating mechanism that uses language and proprioception as complementary query branches before VLM computation.
- Empirical evaluation on CALVIN, LIBERO, and a 14-task real-world benchmark, where ThinkProprio matches or surpasses strong baselines at a fraction of the visual tokens and the lowest per-step latency.

2 Related Work

Vision-Language Feature Extraction. VLA systems differ in how vision-language backbone outputs are presented to the action head. Some pass dense token sets directly, as in π_0 [3], $\pi_{0.5}$ [4], and FLOWER [6], preserving fine-grained information but increasing computation. Others aggregate tokens before control: CogACT [8] compresses the set, while DiT-Block [10], OTTER [9], and LightVLA [11] apply pooling-style reductions that inherit from the broader token-reduction literature [12, 13]. These methods span a spectrum from indiscriminate aggregation, which reduces tokens without regard to task content, to guided selection that conditions retention on an external signal.

Proprioceptive Encoding and Entry. Prior work integrates proprioception with different encodings and entry points. $\pi_{0.5}$ [4] serializes proprioception as text before VLM input, whereas ThinkProprio maps discretized state bins directly to VLM vocabulary IDs. GR00T-N1 [14] and SmolVLA [5] instead project proprioception with multilayer perceptrons into the VLM feature space, which increases representational flexibility but can introduce mismatch relative to pretrained token features. FLOWER [6] conditions the action head directly on proprioceptive inputs, bypassing the backbone,

and some single-system VLAs such as CogACT [8] omit explicit proprioceptive input entirely. Encoding and entry point co-vary in the literature: MLP-encoded state is almost always fed to the action head, token-form state enters the VLM, and proprio-free designs cluster among single-system policies. This coupling has historically been treated as a fixed architectural package rather than as two independent design axes.

Action Conditioning Mechanisms. VLA action heads adopt three main conditioning mechanisms. *In-context* approaches concatenate vision-language and proprioceptive embeddings with the action sequence, as in Dita [7] and decoder-only systems such as OpenVLA [2]. *AdaLN-style* modulation predicts layer-wise scale and shift from pooled features and is used by DiT-Block [10], FLOWER [6], and MDT [15]. *Cross-attention* retains token-level access at higher cost and is used by π_0 [3], $\pi_{0.5}$ [4], and FLOWER's Flow Transformer [6]. The three families trade expressivity for cost: cross-attention is most expressive but scales with the context-token count, AdaLN is cheapest but collapses tokens into a global modulation, and in-context conditioning falls in between.

Across these methods, proprioception is treated as a passive conditioning signal, but not used to select what the policy looks at. ThinkProprio departs from this convention by using state as an active query against visual tokens, alongside the instruction, before the VLM consumes them.

3 Method

We consider a policy π_θ composed of a vision-language backbone and a separate action head. At timestep t, the policy receives an observation o_t comprising n RGB images $(I_t^1, \ldots, I_t^n)$, a language instruction ℓ, and a proprioceptive state $\boldsymbol{q}_t$ that encodes the robot's current configuration, including joint angles and end-effector pose. We represent each input stream as a token sequence with shared embedding dimension D: vision tokens $H_v \in \mathbb{R}^{N_v \times D}$, language tokens $H_l \in \mathbb{R}^{N_l \times D}$, and proprioceptive tokens $H_q \in \mathbb{R}^{N_q \times D}$, where N_v, N_l, and N_q denote the corresponding token counts. As shown in Figure 2, ThinkProprio instantiates this dual-system policy by discretizing proprioception into VLM-vocabulary tokens and using instruction and proprioceptive tokens in two separate guidance branches to select visual patches before VLM computation. The backbone f_{VLM} maps the resulting compact multimodal sequence to conditioning features C, and the action head f_{ACT} predicts a continuous action chunk $\boldsymbol{a}_{t:t+\mathcal{H}}$ conditioned on C.

3.1 Proprioceptive State Encoding

The proprioceptive state $\boldsymbol{q}_t$ contains d_q scalar values, so $N_q = d_q$. We discretize each value with uniform binning over a clipped range, where $q_{\min}$ and $q_{\max}$ are shared scalar clipping bounds applied to each dimension and B is the number of bins. For each state element $q_{t,k}$ with $k \in \{1, \ldots, d_q\}$, we compute the bin index

$$b_{t,k} = \left\lfloor \frac{\text{clip}(q_{t,k}, q_{\min}, q_{\max}) - q_{\min}}{q_{\max} - q_{\min}} \cdot (B-1) \right\rfloor. \tag{1}$$

We map each bin index to a proprioceptive token ID using the reverse mapping $\tau_{t,k} = V - 1 - b_{t,k}$, where V is the VLM vocabulary size. These IDs are vocabulary aliases for discretized state values, not the output of the language tokenizer. Reusing the existing embedding table adds no new embedding parameters and uses the same lookup pathway as text inputs; the reverse indexing keeps the proprioceptive token range disjoint from the tokens produced by the language tokenizer for our prompts, so the two streams do not collide. We then obtain the corresponding embeddings from the VLM token embedding table, $H_q = \text{Embed}(\boldsymbol{\tau}_t)$, where $\boldsymbol{\tau}_t = [\tau_{t,1}, \ldots, \tau_{t,d_q}]$.

3.2 Embodied Visual Token Gating

We gate visual tokens before they enter the VLM so that subsequent VLM computation operates on a compact set of action-relevant patches. The selector uses two complementary guidance branches, one driven by the instruction and one by proprioception. The instruction branch emphasizes task-semantic

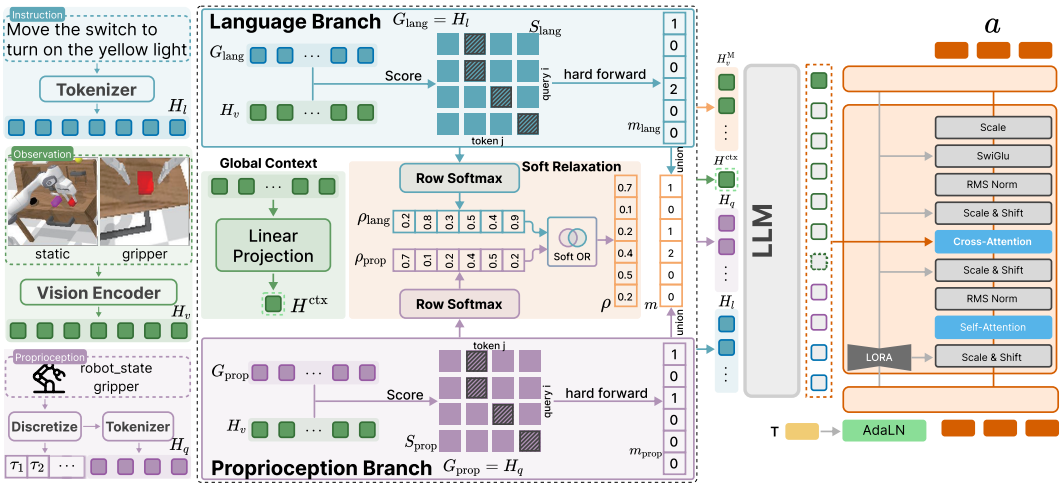


Figure 2: Overview of ThinkProprio. Proprioception is used with the instruction in two guidance branches to select task-relevant visual patches before VLM computation, alongside a global context token. The action head attends to the resulting features to generate actions.

evidence such as objects and goals, while the proprioception branch emphasizes configuration-dependent evidence such as the gripper and contact regions.

Branch-specific scoring. Let $\Omega = \{\text{lang}, \text{prop}\}$ index the two branches, with guidance tokens $G_{\text{lang}} = H_l$ and $G_{\text{prop}} = H_q$. We normalize the visual and guidance tokens as $\tilde{H}_v = \text{RMSNorm}(H_v)$ and $\tilde{G}_\omega = \text{RMSNorm}(G_\omega)$ for each $\omega \in \Omega$. Rather than scoring visual tokens directly from the guidance tokens, we use a vote-based construction that conditions each vote on a visual token, so the selector forms visual-context-dependent votes instead of a single top-down saliency map. For branch ω, we compute

$$Q_\omega = \text{softmax}\left(\frac{\tilde{H}_v \tilde{G}_\omega^\top}{\sqrt{D}}\right)\tilde{G}_\omega, \qquad S_\omega = \frac{\text{RMSNorm}(Q_\omega)\tilde{H}_v^\top}{\sqrt{D}}. \tag{2}$$

Each row of Q_ω is obtained by letting visual token i attend to the branch guidance tokens and extract the guidance information most relevant to it. The resulting guidance-conditioned query then scores all visual tokens as candidates for retention. The matrix $S_\omega \in \mathbb{R}^{N_v \times N_v}$ is therefore a branch-specific vote matrix in which row index i denotes the visual token that issues the query and column index j denotes a candidate visual token for retention; entry $S_\omega[i, j]$ measures how strongly guidance-conditioned visual token i votes for retaining visual token j.

Vote-based gating with straight-through relaxation. Each branch converts its score matrix into a hard selection mask for the forward pass and a soft relaxation for gradient propagation. During training, we perturb each branch score matrix as $\hat{S}_\omega = S_\omega + \alpha\Gamma_\omega$, where Γ_ω is sampled elementwise from the standard Gumbel distribution and α is cosine-annealed from α_{start} to α_{end}. This perturbation encourages exploration early in training and approaches deterministic selection near convergence. At inference time, we remove the perturbation by setting $\hat{S}_\omega = S_\omega$.

In the hard forward path, each row i casts one vote for a candidate visual token, $z_\omega[i] = \arg\max_{j \in \{1,\ldots,N_v\}} \hat{S}_\omega[i, j]$. Branch ω selects token j if it receives at least one vote, $m_\omega[j] = \mathbf{1}[\exists\, i :\, z_\omega[i] = j]$, and we merge the two branch masks by union,

$$m[j] = m_{\text{lang}}[j] \vee m_{\text{prop}}[j], \qquad M = \{j : m[j] = 1\}. \tag{3}$$

In parallel, we compute the soft relaxation used for backpropagation. The row-wise probabilities $P_\omega = \text{softmax}(\hat{S}_\omega)$ give $P_\omega[i, j]$ as the relaxed probability that row i votes for token j. Treating per-row votes as conditionally independent, the probability that token j receives at least one vote within branch ω is the *noisy-OR* [16] over rows (see Appendix Section A), $\rho_\omega[j] = 1 - \prod_{i=1}^{N_v}(1 - P_\omega[i, j])$.

We aggregate the two branches by an analogous noisy-OR over branches:

$$\rho[j] = 1 - \big(1 - \rho_{\text{lang}}[j]\big)\big(1 - \rho_{\text{prop}}[j]\big). \tag{4}$$

The straight-through gate combines the hard union mask and the soft union probability $w[j] = m[j] + \rho[j] - \text{sg}(\rho[j])$, where $\text{sg}(\cdot)$ denotes stop-gradient. Thus, $w[j]$ equals the hard mask $m[j]$ in the forward pass, while gradients through $w[j]$ follow the soft probability $\rho[j]$. During training, each retained token $H_v[j]$ with $j \in M$ is multiplied by $w[j]$ before being packed into H_v^M. At inference time, we use the hard mask directly.

Diversity regularization. To prevent the instruction and proprioception branches from selecting identical token sets, we aggregate the unperturbed scores over voting rows, $r_\omega[j] = \sum_{i=1}^{N_v} S_\omega[i, j]$, and penalize agreement between the resulting normalized selection distributions:

$$\mathcal{L}_{\text{div}} = \big[\,\text{sim}_{\cos}\big(\,\text{softmax}(r_{\text{lang}}),\, \text{softmax}(r_{\text{prop}})\big) - \gamma\big]_+^2, \tag{5}$$

where $\text{sim}_{\cos}(\cdot, \cdot)$ denotes cosine similarity and γ is the diversity margin. This encourages complementary instruction and proprioception guidance while still allowing the union mask M to retain evidence supported by both branches.

Global context token. Aggressive token selection can remove scene-level information that receives little support from individual votes but remains useful for action generation. To preserve coarse visual context, we append a learned global token computed from the full pre-selection visual sequence,

$$H^{\text{ctx}} = W_c \left(\frac{1}{N_v} \sum_{j=1}^{N_v} H_v[j] \right), \tag{6}$$

where W_c is a learned linear projection.

3.3 Training Objective

After visual token gating, the VLM maps the compact multimodal sequence to conditioning features, $C = f_{\text{VLM}}([H_v^M; H^{\text{ctx}}; H_q; H_l])$. The selector, VLM backbone, and action head are trained end-to-end. We train the action head with flow matching [17]. We sample a continuous flow time $T \sim \text{Unif}(0, 1)$ and construct a noisy action chunk $\boldsymbol{a}_{t:t+\mathcal{H}}^T = (1-T)\boldsymbol{a}_{t:t+\mathcal{H}} + T\epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Following FLOWER, we embed T and inject it into the action Transformer through global AdaLN-style modulation, while the action tokens cross-attend to C. Under this linear interpolation, the target velocity field is $\frac{d}{dT}\boldsymbol{a}_{t:t+\mathcal{H}}^T = \epsilon - \boldsymbol{a}_{t:t+\mathcal{H}}$. The action head predicts $v_\theta = f_{\text{ACT}}(\boldsymbol{a}_{t:t+\mathcal{H}}^T, T, C)$, and we optimize $\mathcal{L}_{\text{fm}} = \mathbb{E}_{T,\epsilon}\Big[\|v_\theta - (\epsilon - \boldsymbol{a}_{t:t+\mathcal{H}})\|^2\Big]$. The final training objective combines the flow matching loss with the branch diversity regularizer,

$$\mathcal{L} = \mathcal{L}_{\text{fm}} + \lambda_{\text{div}}\, \mathcal{L}_{\text{div}}. \tag{7}$$

where λ_{div} controls the strength of the diversity regularizer.

4 Experiments

We address the following research questions: **RQ1**: How should proprioception be represented and incorporated into a VLA policy? **RQ2**: Can proprioception guide visual-token selection to retain task- and state-relevant evidence while reducing inference cost? **RQ3**: Do the resulting design choices improve manipulation performance and efficiency across simulation and real-world tasks?

4.1 Experiments Setup

Baselines. On CALVIN, we compare against single-system VLAs such as OpenVLA [2], which represents actions as discrete tokens, and methods with dedicated continuous-control action heads, including GR-1 [18], RoboFlamingo [19], π_0 [3], $\pi_{0.5}$ [4], and FLOWER [6]. We report π_0 and

Table 2: Simulation benchmark results. CALVIN ABC→D reports success rate (%) at each chain length and average completed chain length. LIBERO reports success rate (%) across 10 tasks per suite. *Finetuned by us. Best results are in **bold**; second-best are underlined.

(a) CALVIN ABC→D

Method	LH-1 ↑	LH-2 ↑	LH-3 ↑	LH-4 ↑	LH-5 ↑	Avg. ↑
OpenVLA	91.3	77.8	62.0	52.1	43.5	3.27
GR-1	85.4	71.2	59.6	49.7	40.1	3.06
RoboFlamingo	82.4	61.9	46.6	33.1	23.5	2.47
π_0^*	70.0	48.0	37.0	28.0	18.0	2.01
$\pi_{0.5}^*$	71.0	56.0	45.0	37.0	29.0	2.38
SuSIE	87.0	69.0	49.0	38.0	26.0	2.69
VPP	95.7	91.2	86.3	81.0	75.0	4.29
Seer	96.3	91.6	86.1	80.3	74.0	4.29
FLOWER	**99.3**	**96.0**	90.3	82.3	75.5	4.44
ThinkProprio	98.9	95.4	**91.6**	**86.4**	**79.1**	**4.52**

(b) LIBERO

Method	Spa. ↑	Obj. ↑	Goal ↑	Long ↑	Avg. ↑
OpenVLA	84.7	88.4	79.2	53.7	76.5
WorldVLA	85.6	89.0	82.6	59.0	79.1
SmolVLA	93.0	94.0	91.0	77.0	88.8
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1
COA-VLA	85.3	93.1	85.8	55.0	79.8
π_0	96.8	98.8	95.8	85.2	94.2
$\pi_{0.5}$	98.0	97.8	95.6	85.8	94.3
FLOWER	97.5	99.1	96.1	94.9	96.9
LightVLA	**98.4**	98.4	**98.2**	94.6	97.4
ThinkProprio	**98.4**	**99.2**	98.0	**95.2**	**97.7**

Table 3: Real-world task success rates. Left: pick-place tasks. Right: drawer tasks.

Pick-place	Banana		Corn		Tape	
	FLOWER	ThinkProprio	FLOWER	ThinkProprio	FLOWER	ThinkProprio
Basket → Plate	16/20	17/20	17/20	19/20	16/20	19/20
Basket → Table	16/20	18/20	15/20	18/20	17/20	19/20
Plate → Basket	19/20	17/20	15/20	16/20	16/20	16/20
Table → Basket	16/20	18/20	15/20	17/20	16/20	19/20

Drawer	FLOWER	ThinkProprio
Open	16/20	18/20
Close	16/20	18/20

$\pi_{0.5}$ results from our own fine-tuning runs, marked with *. We further compare with visual planning methods, including SuSIE [20], VPP [21], and Seer [22]. On LIBERO, we compare with strong OpenVLA variants and recent VLA baselines, including OpenVLA-OFT [23], COA-VLA [24], LightVLA [11], WorldVLA [25] and SmolVLA [5]. Unless otherwise noted, all reported results are means over 5 seeds.

Simulation & Real-world setup. We evaluate ThinkProprio in both simulation and the real world. **CALVIN** [26] evaluates long-horizon control by requiring policies to complete chains of five tasks. **LIBERO** [27] evaluates generalization across four suites, Spatial, Object, Goal, and Long. For real-world evaluation, we use a UR3 arm with a parallel gripper and two RGB cameras: a fixed third-person view and a wrist-mounted view, as shown in Figure 3. We collect approximately 120 minutes of teleoperated demonstrations and fine-tune from a pretrained checkpoint. The real-world experiments contains 14 tasks: 12 pick-and-place tasks across three objects (banana, corn, and tape), and two drawer tasks (open and close). The Gumbel exploration noise annealed from 1.0 to 0.01 and a diversity regularizer using weight 5×10^{-4}, margin $\gamma = 0.75$. Full implementation details are in Appendix Section B.

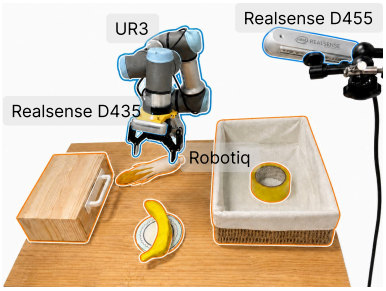


Figure 3: Real-world task setup.

4.2 Performance and Efficiency

Simulation performance. On CALVIN ABC→D (Table 2a), success drops with chain length, consistent with compounding errors. ThinkProprio achieves the best Avg. Len. of 4.52, exceeding FLOWER's 4.44. FLOWER is stronger at LH-1 and LH-2, while ThinkProprio leads from LH-3 to LH-5, consistent with state-conditioned gating helping most when configuration drift accumulates over longer chains. The gains are strongest at longer horizons: ThinkProprio reaches 86.4 % at LH-4 and 79.1 % at LH-5, reducing FLOWER's LH-5 failure rate by roughly 15 %. This supports the intuition that state- and instruction-guided token selection helps retain key visual evidence as the configuration evolves. On LIBERO (Table 2b), ThinkProprio achieves the best overall success, ties on LIBERO-Spatial, and leads on LIBERO-Object and LIBERO-Long.

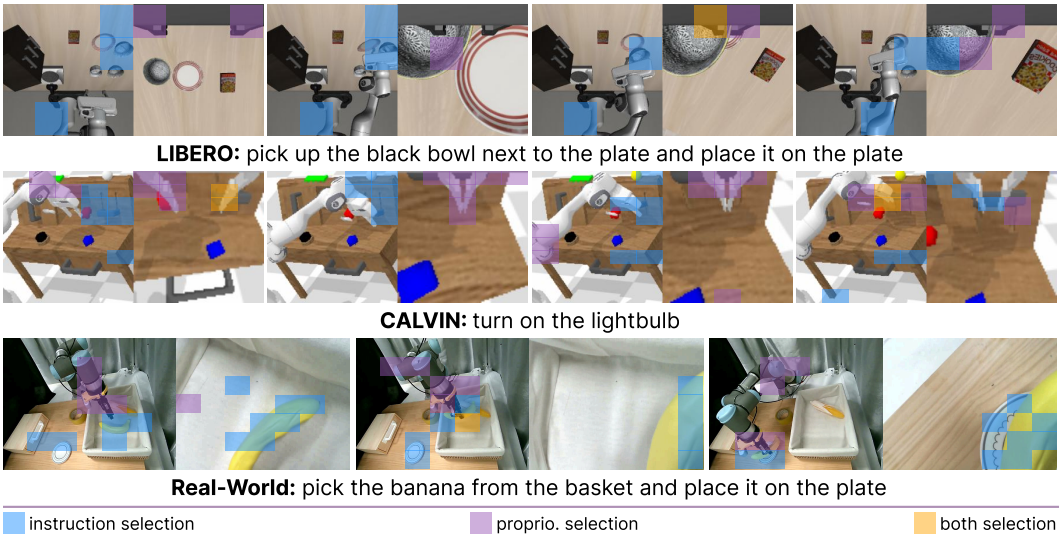


Figure 4: Qualitative token-selection results on LIBERO, CALVIN, and the real-world setup. Across time, the retained tokens track task-relevant objects, targets, and robot–object interaction regions.

Real-world robot evaluation. We compare against FLOWER as the strongest baseline for real world experiments. Table 3 reports successful trials out of 20 for each task. ThinkProprio achieves an overall success rate of $88.9\,\%$ against $80.7\,\%$ for FLOWER. Gains are consistent across objects and task categories: ThinkProprio matches or exceeds FLOWER on 13 of 14 tasks, with the single exception being Banana Plate→Basket (17/20 vs. 19/20). Drawer manipulation shows a uniform improvement of 18/20 versus 16/20 on both open and close. These results confirm that selected visual tokens provide consistent gains across diverse task configurations.

Computational efficiency. Table 4 compares inference cost on CALVIN ABC→D. ThinkProprio keeps only about 12 of 100 visual tokens per step on average, shortening the sequence processed by both the VLM and the action head. The selector gathers the kept tokens, pads to the per-batch max, and supplies an attention mask. Despite this overhead, ThinkProprio reaches lower end-to-end latency than FLOWER ($22\,\mathrm{ms}$ vs. $52\,\mathrm{ms}$). Peak VRAM grows marginally over FLOWER from selector parameters, still far below OpenVLA.

Table 4: Inference efficiency on CALVIN ABC→D. Latency is per timestep in ms; VRAM is MB.

Method	Tokens	Latency	VRAM	Avg. Len.
OpenVLA	256	164	14574	3.27
π_0	256	104	6692	2.01
$\pi_{0.5}$	256	138	7038	2.38
FLOWER	100	52	1848	4.44
ThinkProprio	**12**	**22**	**1899**	**4.52**

4.3 Token Selection Analysis

Qualitative behavior. Figure 4 visualizes retained tokens across simulation and real-world rollouts. The two streams play complementary roles: instruction selected tokens cover task-referenced objects and target regions, while proprioception selected tokens track the gripper and its immediate contact area, with overlap concentrated at the moment of interaction.

Complementary query sources. Table 5 decomposes retained tokens by query source. The overlap stays below one token across all benchmarks, an order of magnitude smaller than either stream alone, so the proprioception branch contributes genuinely complementary evidence rather than re-selecting language-grounded objects. The retained set thus extends beyond a language-conditioned object mask to include configuration-dependent interaction cues.

Table 5: Retained-token counts by query source. Both gives their overlap.

Benchmark	lang	prop	Both
CALVIN	8.60 ± 0.36	3.52 ± 0.39	0.25 ± 0.10
LIBERO-Spa.	4.78 ± 0.83	3.37 ± 0.57	0.35 ± 0.08
LIBERO-Obj.	4.48 ± 0.44	2.58 ± 0.27	0.18 ± 0.14
LIBERO-Goal	3.43 ± 0.48	2.60 ± 0.50	0.10 ± 0.08
LIBERO-Long	4.17 ± 0.49	3.47 ± 0.17	0.17 ± 0.08
Real-World	9.36 ± 0.42	5.87 ± 0.28	0.61 ± 0.13

4.4 Ablation Studies

We isolate the design choices behind ThinkProprio with others fixed, ablating each component's contribution and the entry point for proprioception driving visual-token selection.

Component ablation. In Table 6, VLM-vocabulary proprioceptive tokens give a small but consistent gain across CALVIN and LIBERO-Spatial. Visual token selection alone reduces the visual-token budget but slightly hurts performance, suggesting that local selection can discard scene-level evidence needed for action generation. Adding the global context token H^{ctx} recovers this loss, and the diversity loss gives the best result on both benchmarks by encouraging selected tokens to cover complementary evidence.

Table 6: Ablations on CALVIN ABC→D and LIBERO-Spatial. Tokens denotes the retained visual-token budget.

	CALVIN ABC→D		LIBERO-Spatial	
Method variant	**Tokens**	**Avg. Len. ↑**	**Tokens**	**SR ↑**
FLOWER	All	4.44±0.03	All	97.5±0.08
+Proprio tokens	All	4.45±0.02	All	97.8±0.05
+Token selection	∼12 %	4.35±0.04	∼23 %	97.2±0.06
$+H^{\text{ctx}}$	∼12 %	4.48±0.03	∼23 %	98.2±0.04
+Div. loss	∼12 %	**4.52±0.02**	∼23 %	**98.4±0.03**
$\Omega = \{\text{lang}\}$	∼8 %	3.40±0.06	∼14 %	96.6±0.05
$\Omega = \{\text{prop}\}$	∼3 %	3.12±0.10	∼10 %	96.0±0.04
Mean pooling	1 %	4.20±0.02	1 %	90.3±0.04
Max pooling	25 %	4.33±0.03	25 %	92.8±0.02
Random	50 %	4.24±0.02	50 %	87.4±0.08

Query signal for token retention. The lower panel of Table 6 compares practical retention strategies. Task-agnostic pooling and random baselines retain much of FLOWER's performance, indicating substantial visual redundancy in CALVIN. Query-guided retention is sensitive to the guidance signal: instruction-only captures task-referenced objects but lacks embodiment context, proprioception-only is too narrow without task semantics, and only their combination keeps both object-centric and configuration-dependent cues. The 1.1 Avg. Len. gap on CALVIN exceeds any plausible budget effect, and the near-zero cross-branch overlap indicates the streams are complementary rather than redundant.

Proprioceptive encoding and entry point. Table 7 isolates proprioceptive integration methods. Where state enters the policy matters as much as whether it is present: late action-head modulation leaves the baseline unchanged, while projecting continuous state into the VLM input hurts performance. VLM-vocabulary proprioceptive tokens avoid this projection mismatch by using the backbone's native embedding lookup rather than a learned feature projector. The standalone gain is modest, but this is the only entry point that both preserves performance and exposes proprioceptive tokens to the downstream visual selector.

Table 7: Proprioceptive integration ablation on CALVIN ABC→D.

Encoding	**Entry**	**ACT cond.**	**Avg. Len. ↑**
None	–	–	4.44
MLP	ACT	AdaLN	4.44
MLP	VLM	Cross-attn	4.15
VLM-vocab	VLM	Cross-attn	**4.48**

5 Conclusion

We presented ThinkProprio, a VLA policy that exposes proprioception as VLM-vocabulary tokens and pairs it with language to guide visual-token retention. Across simulation and real-world experiments, ThinkProprio improves strong baselines while substantially reducing inference latency. The ablations show that proprioception is most effective when used as an active signal for visual selection.

Limitations

Our real-world evaluation is confined to a setup with rigid objects, broader embodiments, object categories, and baselines remaining future work. The pre-VLM selector forms a vote matrix, incurring $O(N_v^2 D)$ cost that may bottleneck higher visual resolutions. Finally, although the selector retains human-interpretable evidence, it occasionally preserves uninformative patches such as background, as it is supervised by the action objective rather than explicit relevance signals; tightening selection toward interpretable evidence is a promising direction for future work.

References

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pages 2679–2713. PMLR, 2025.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[4] P. Intelligence, K. Black, N. Brown, J. Darpinian, K. Dhabalia, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, M. Y. Galliker, D. Ghosh, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, D. LeBlanc, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, A. Z. Ren, L. X. Shi, L. Smith, J. T. Springenberg, K. Stachowicz, J. Tanner, Q. Vuong, H. Walke, A. Walling, H. Wang, L. Yu, and U. Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

[5] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[6] M. Reuss, H. Zhou, M. Rühle, O. E. Yağmurlu, F. Otto, and R. Lioutikov. Flower: Democratizing generalist robot policies with efficient vision-language-flow models. In J. Lim, S. Song, and H.-W. Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 3736–3761. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/reuss25a.html.

[7] Z. Hou, T. Zhang, Y. Xiong, H. Duan, H. Pu, R. Tong, C. Zhao, X. Zhu, Y. Qiao, J. Dai, and Y. Chen. Dita: Scaling diffusion transformer for generalist vision-language-action policy. *arXiv preprint arXiv:2503.19757*, 2025.

[8] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[9] H. Huang, F. Liu, L. Fu, T. Wu, M. Mukadam, J. Malik, K. Goldberg, and P. Abbeel. Otter: A vision-language-action model with text-aware visual feature extraction. *arXiv preprint arXiv:2503.03734*, 2025.

[10] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

[11] T. Jiang, X. Jiang, Y. Ma, X. Wen, B. Li, K. Zhan, P. Jia, Y. Liu, S. Sun, and X. Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

[12] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

[13] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021.

[14] 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.

[15] M. Reuss, Ö. E. Yağmurlu, F. Wenzel, and R. Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. *arXiv preprint arXiv:2407.05996*, 2024.

[16] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[17] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

[18] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=NxoFmGgWC9.

[19] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, H. Li, and T. Kong. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[20] K. Black, M. Nakamoto, P. Atreya, H. R. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=c0chJTSbci.

[21] Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Forty-second International Conference on Machine Learning*, 2025.

[22] Y. Tian, S. Yang, J. Zeng, P. Wang, D. Lin, H. Dong, and J. Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. *arXiv preprint arXiv:2412.15109*, 2024.

[23] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

[24] J. Li, Y. Zhu, Z. Tang, J. Wen, M. Zhu, X. Liu, C. Li, R. Cheng, Y. Peng, Y. Peng, and F. Feng. Coa-vla: Improving vision-language-action models via visual-textual chain-of-affordance, 2025. URL https://arxiv.org/abs/2412.20451.

[25] J. Cen, C. Yu, H. Yuan, Y. Jiang, S. Huang, J. Guo, X. Li, Y. Song, H. Luo, F. Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[26] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[27] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36: 44776–44791, 2023.

[28] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2024. URL https://arxiv.org/abs/2309.13037.

[29] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

[30] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

A Noisy-OR Relaxation for Mask Union

This appendix expands the noisy-OR construction inlined in Section 3.2: how it arises as the expected hard mask under a natural row-wise voting model, why it is preferred to simpler differentiable surrogates, and how it interacts with the diversity regularizer. The instantiations used by the selector — across voting rows within a branch and across branches — are stated in the main text and are not repeated here.

Boolean union and its noisy-OR relaxation. The Boolean union of events $b_1, \ldots, b_n \in \{0, 1\}$,

$$b_\vee = \bigvee_{i=1}^{n} b_i = \mathbf{1}[\textstyle\sum_{i=1}^{n} b_i > 0], \tag{8}$$

is non-differentiable. Replacing each b_i by a Bernoulli probability $p_i \in [0, 1]$ and assuming the events are conditionally independent gives the differentiable surrogate

$$\rho_\vee(p_1, \ldots, p_n) = \Pr\left(\bigvee_{i=1}^{n} b_i = 1\right) = 1 - \prod_{i=1}^{n}(1 - p_i), \tag{9}$$

known as the *noisy-OR* [16]. The term $\prod_i(1 - p_i)$ is the probability that no event occurs, so subtracting it from one gives the probability that at least one does.

Probabilistic interpretation in the selector. The noisy-OR is not introduced from the outside as a smooth approximation of $\vee$. Treating each row's softmax $P_\omega[i, \cdot]$ as a categorical distribution over which token row i votes for, and the rows as conditionally independent, the probability that token j receives at least one vote across the N_v rows is exactly

$$\mathbb{E}[m_\omega[j]] \;=\; \Pr(\exists\, i :\; z_\omega[i] = j) \;=\; 1 - \prod_{i=1}^{N_v}(1 - P_\omega[i, j]) \;=\; \rho_\omega[j], \tag{10}$$

where $m_\omega[j] = \mathbf{1}[\exists\, i :\; z_\omega[i] = j]$ is the hard branch mask. The same identity applies across branches when branch decisions are taken as independent. The soft scores $\rho_\omega[j]$ and $\rho[j]$ used in Section 3.2 are therefore the expected hard masks under the row-wise and cross-branch surrogate voting models, not arbitrary smooth surrogates.

Properties relied on by the selector.

- **(P1) Boundary agreement with hard OR.** $\rho_\vee = 0$ iff every $p_i = 0$, and $\rho_\vee \to 1$ if any $p_i \to 1$. The forward-pass mask and the soft expectation agree at the corners of $[0, 1]^n$.
- **(P2) Strict monotonicity with non-degenerate gradient.** $\partial\rho_\vee/\partial p_i = \prod_{k \neq i}(1 - p_k) \geq 0$, with equality only when some other $p_k = 1$. Every input receives a gradient unless the union is already saturated by another event.
- **(P3) Bounded without renormalization.** $\rho_\vee \in [0, 1]$ for any n, so evidence aggregated over N_v rows or two branches needs no division by n.
- **(P4) Union-style accumulation without dilution or overshoot.** Two independent moderate votes reinforce (e.g. $p_1 = p_2 = 0.5$ give $\rho_\vee = 0.75$), a single confident vote keeps the union near 1 regardless of how many low-confidence votes accompany it, and the value never exceeds 1.

Table 8: Differentiable surrogates for the Boolean union of n Bernoulli events. Only the noisy-OR satisfies all four properties.

Surrogate	P1	P2	P3	P4
$\max_i p_i$	Yes	Partial	Yes	No
$\sum_i p_i$	No	Yes	No	No
$\frac{1}{n}\sum_i p_i$	No	Yes	Yes	No
Noisy-OR $1 - \prod_i (1 - p_i)$	Yes	Yes	Yes	Yes

Comparison with simpler surrogates. Table 8 contrasts noisy-OR with three natural alternatives. The maximum $\max_i p_i$ satisfies P1 and P3 but is gradient-zero away from the $\arg\max$ (P2 partial) and does not accumulate: two votes at $p = 0.5$ stay at 0.5 rather than reinforcing each other (P4 violated). The sum $\sum_i p_i$ violates P1 and P3 by exceeding 1. The normalized mean $\frac{1}{n}\sum_i p_i$ restores boundedness but breaks P1, and *dilutes* one confident vote among many low-confidence ones down toward $1/n$ — exactly the regime the selector operates in, where only a small fraction of rows vote confidently for any given column.

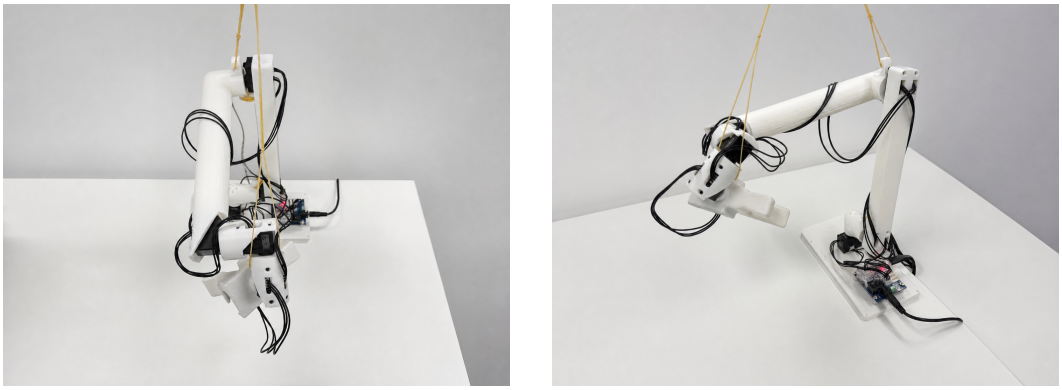
Independence assumption and the role of the diversity regularizer. The expected-mask identity above holds exactly when the underlying votes are conditionally independent; otherwise the noisy-OR is an approximation. In the selector this assumption appears at two scales.

Across voting rows within a branch. The rows share a single score matrix and are not strictly independent, but each row casts at most one effective vote and the guidance-conditioned queries separate row distributions, so we expect the residual dependence to be limited.

Across branches. We do not attempt to enforce probabilistic independence between the language and proprioception branches. The diversity regularizer $\mathcal{L}_{\text{div}}$ instead discourages branch collapse by penalizing cosine similarity between the row-aggregated selection distributions $\mathrm{softmax}(r_{\text{lang}})$ and $\mathrm{softmax}(r_{\text{prop}})$, which reduces empirical overlap and keeps branch behavior aligned with the complementary-branch intent of the architecture. This narrows the regime in which the cross-branch noisy-OR is loosest — heavily overlapping branches — without claiming the approximation is exact.

Role in the forward pass. The noisy-OR never enters the forward computation. The selector uses the hard Boolean union $m[j] = m_{\text{lang}}[j] \vee m_{\text{prop}}[j]$ both at inference and at the forward step of training, and the straight-through gate routes gradients through the soft $\rho[j]$ while leaving the forward value equal to $m[j]$. The relaxation therefore only shapes the learning signal; deployed model behavior is identical to that of a deterministic argmax-union selector.

Gradient flow and token exploration. The straight-through gate passes gradient only to tokens in the current selection M: tokens with $m[j] = 0$ are dropped from H_v^M before the VLM, so their gate value $w[j]$ never enters the loss and they receive no direct gradient in that step. Coverage of the full token set is instead provided over training by the Gumbel perturbation $\hat{S}_\omega = S_\omega + \alpha\Gamma_\omega$. Early in training, the large exploration scale α makes the per-row argmax votes stochastic, so the realized selection M varies from step to step and, across steps, most visual tokens are eventually selected and updated at least once. A dropped token also continues to shape learning indirectly, since it participates in the scores of other tokens (as a voting row in $\tilde{H}_v$ and through the row-wise normalization) and in the aggregated selection distributions r_ω used by $\mathcal{L}_{\text{div}}$. As α anneals from 1.0 to 0.01, selection sharpens toward the deterministic argmax-union used at inference, so exploration is concentrated early and the gate becomes effectively hard by convergence. This is distinct from property (P2), which concerns the gradient of the noisy-OR surrogate with respect to its probability inputs rather than the realization of the discrete forward selection.



(a) Overhead view. (b) Side view.

Figure 5: Customized GELLO-style leader-side teleoperation interface used for collecting real-world demonstrations.

B Experiment Details

B.1 Observations and Proprioceptive State

Image preprocessing follows the dataset transform configs. For CALVIN, the static and gripper-camera views are resized to 224×224. During training, we apply RandomShiftsAug with padding 10 for the static view and padding 4 for the gripper view, then scale images to $[0, 1]$ and normalize with CLIP statistics. Validation disables RandomShiftsAug but keeps resize and normalization. The proprioceptive state is the 15D robot observation vector, normalized using dataset statistics and with additional normalization of orientation entries. Actions are 7D relative end-effector commands scaled to $[-1, 1]$.

For LIBERO, we use the same augmentation and normalization structure, resizing images to 112×112. We form a 9D proprioceptive state by concatenating seven joint values with the 2D gripper state. Actions use the same 7D relative command parameterization and are scaled to $[-1, 1]$.

For real-world evaluation, we use the same two-stream policy interface with a fixed third-person camera and a wrist-mounted camera. Both views are resized to 224×224 and normalized with Florence/CLIP-style image statistics; training additionally uses RandomShiftsAug with padding 10 for the static view, padding 4 for the wrist view, and color jitter. The UR3 proprioceptive state is a 16D vector containing end-effector position (3), end-effector quaternion (4), gripper state (1), six joint values (6), and two zero-padded dimensions, normalized using statistics computed from training episodes. Actions contain six relative pose deltas and one gripper command; the motion dimensions are normalized and the gripper target is binarized. Across CALVIN, LIBERO, and real-world runs, unless otherwise specified, normalized proprioceptive values are clipped to $[-3, 3]$, uniformly discretized into $B = 256$ bins, and mapped to VLM-vocabulary token IDs as in Section 3.1; Table 14 studies this bin and clip choice. The real-world dataset contains approximately 120 minutes of teleoperated demonstrations over 14 tasks: 12 pick-and-place tasks across banana, corn, and tape using Basket→Plate, Basket→Table, Plate→Basket, and Table→Basket routes, plus two drawer tasks, open and close.

Teleoperation interface. The real-world demonstrations were collected with the customized GELLO-style leader-side teleoperation interface shown in Figure 5 [28]. The operator moves a lightweight linkage and gripper handle, and the adapted software maps these inputs to the UR3 during data collection. The policy observations remain the fixed and wrist-mounted RGB streams together with the robot proprioceptive state described above.

Table 9: Training and optimization hyperparameters used across experiments.

Setting	Value
Optimizer	AdamW
Learning rate	2×10^{-5}
AdamW betas	$(0.9, 0.95)$
Weight decay	0.05 (no decay on norm/bias)
Per-GPU batch size	8
Number of GPUs	1
Total steps	$50{,}000$
LR schedule	tri-stage (warmup $\rightarrow$ hold $\rightarrow$ cosine)
Warmup / hold / decay	$0.05/0.10/0.85$
Warmup start lr	$0.1 \times \mathrm{lr}$
Final lr	$0.5 \times \mathrm{lr}$
Precision	bf16-mixed
EMA decay	0.999
VLM token dropout	0.1
Gumbel scale	$1.0 \rightarrow 0.01$
Diversity loss	$\lambda_{\mathrm{div}} = 5 \times 10^{-4}$, margin $\gamma = 0.75$

B.2 Model Architecture

We build on FLOWER and use Florence-2-Large (`microsoft/Florence-2-large`) as the vision-language backbone. Following FLOWER's recipe, we fine-tune Florence rather than freezing it, so comparisons use the same backbone adaptation regime. A special token `<Flow>` is embedded and inserted to mark the conditioning boundary, and we apply token dropout with probability 0.1 to the VLM encoder outputs during training. The pre-VLM selector uses the branch-specific vote-matrix scoring described in Section 3.2 for the instruction and proprioception branches, and its global context token uses the learned linear projection W_c to compute H^{ctx} from the mean pre-selection visual feature. The action generator is a rectified-flow / Diffusion Transformer with hidden size 1024, 18 transformer layers, and 16 attention heads, using dropout 0.1 in attention, residual, and MLP blocks. The policy predicts an action chunk of length 10, executed with chunked replanning every 10 environment steps.

B.3 Optimization Hyperparameters

We train end-to-end with AdamW using learning rate 2×10^{-5}, betas $(0.9, 0.95)$, and weight decay 0.05 applied to non-normalization and non-bias parameters (norm/bias parameters use zero weight decay). Training uses bf16-mixed precision with per-GPU batch size 8 on one GPU. We use a tri-stage learning-rate schedule over 50k steps (matching `max_epochs=50` and `limit_train_batches=1000`): (i) linear warmup for 5% of steps from $0.1 \times \mathrm{lr}$ to lr, (ii) constant hold for 10% of steps, and (iii) cosine decay for the remaining 85% to a final learning rate of $0.5 \times \mathrm{lr}$. Selector training anneals the Gumbel exploration scale from 1.0 to 0.01 and uses the branch diversity coefficient $\lambda_{\mathrm{div}} = 5 \times 10^{-4}$ from the objective $\mathcal{L} = \mathcal{L}_{\mathrm{fm}} + \lambda_{\mathrm{div}}\mathcal{L}_{\mathrm{div}}$, with margin $\gamma = 0.75$. We additionally maintain an exponential moving average (EMA) of parameters with decay 0.999 and use EMA weights for evaluation.

B.4 Inference Settings and Measurement Protocol

At inference time, we run rectified-flow sampling with 4 steps (`num_sampling_steps=4`) per action chunk. The model predicts a 10-step chunk and replans every 10 environment steps. Inference is performed in bf16 and always uses both camera views.

CALVIN long-horizon evaluation follows the standard 5-subtask instruction-chain protocol: each subtask is given up to 360 environment steps, and we report success rates for completing 1 through 5

subtasks as well as the average successful sequence length. LIBERO evaluation uses 50 trials per task with a maximum horizon of 520 environment steps; we report per-task success and suite averages.

For latency and VRAM profiling, we measure per-environment-step end-to-end inference time including vision encoding, the pre-VLM selector, the Florence encoder forward pass, and all diffusion sampling steps. Although the policy predicts 10-action chunks, the latency values in Tables 4 and 13 are reported per environment step rather than as a total per action chunk. We compute latency with CUDA synchronization to avoid asynchronous kernel overlap artifacts, and report the mean over a fixed number of inference steps (1000 steps in our efficiency tables). In Table 13, the Vision column includes encoding both camera views. Peak VRAM is reported as the maximum allocated GPU memory during inference. All profiling numbers in the paper are collected on a single RTX 4090 GPU under the same bf16 and two-view settings as evaluation.

B.5 Baseline Details

For the extended tables, we keep only the architecture attributes needed to interpret efficiency: scale, backbone, and visual-token count; "–" denotes not applicable or not reported.

FLOWER. FLOWER [6] is the closest dual-system baseline to our implementation. It uses Florence-2-Large as the vision-language backbone and conditions a flow-based action generator through dense cross-attention. We compare against FLOWER under the same benchmark settings used in the main simulation table.

π_0 and $\pi_{0.5}$. π_0 [3] and $\pi_{0.5}$ [4] pair a PaliGemma VLM with a flow-based action generator. We fine-tune the open variants on CALVIN ABC→D, using them to compare continuous proprioceptive action-head conditioning against token-form proprioceptive input. These are the *-marked π_0 and $\pi_{0.5}$ rows in the main CALVIN table.

OpenVLA and OpenVLA-OFT. OpenVLA [2] is a single-system VLA that discretizes actions as language-model tokens. OpenVLA-OFT [23] extends this family with an action expert and improved fine-tuning recipe, and is included for LIBERO comparisons.

GR-1. GR-1 [18] is a generative robot policy that combines visual encoders with a sequence model for language-conditioned manipulation. We include it as a compact VLA-style baseline on CALVIN.

RoboFlamingo and DeerVLA. RoboFlamingo [19] adapts a Flamingo-style vision-language model to robot control. DeerVLA [29] builds on this family with improved data and training choices, providing additional single-system VLA comparisons on CALVIN.

LightVLA. LightVLA [11] is the closest efficiency-oriented baseline. We distinguish it from ThinkProprio along four axes:

- **Target.** LightVLA targets visual-token computation reduction through instruction-guided pruning, whereas ThinkProprio asks how proprioception should participate in visual grounding.
- **Guidance signal.** LightVLA uses the instruction as the pruning signal; ThinkProprio uses separate language and proprioception branches so task semantics and robot-state cues can make complementary votes.
- **Selection mechanism.** ThinkProprio merges branch masks with a hard union in the forward pass and a noisy-OR relaxation for gradients.
- **Regularization and context.** ThinkProprio uses a diversity loss to discourage the two branches from collapsing to the same selection pattern, and keeps a global context token so aggressive local selection does not discard coarse scene information.

Diff-P-CNN and MDT. Diff-P-CNN [30] and MDT [15] are diffusion-style manipulation policies without a large VLM backbone. They provide non-VLM references for CALVIN, especially on the D→D split.

SuSIE, VPP, and Seer. SuSIE [20], VPP [21], and Seer [22] are visual planning or predictive representation baselines. They are included to compare against methods that rely on planning-oriented visual abstractions rather than a VLM-action-head decomposition.

RoboUniView. RoboUniView is included in the CALVIN D→D comparison as a prior multi-view robot learning baseline. We report it only where the corresponding split result is available.

C Additional Results

C.1 CALVIN Long-Horizon Results

Table 10 extends the main CALVIN ABC→D comparison in Table 2a with architecture details, while Tables 11 and 12 report the additional ABCD→D and D→D splits. In these tables, the Tokens column is the average number of retained visual tokens per timestep out of 100 input visual tokens. The split-specific means for ThinkProprio are 12, 15, and 14 tokens; the "around 15 %" statement in the main text is a rounded summary across CALVIN settings.

Table 10: CALVIN ABC→D long-horizon success with architecture details.

Method	**Architecture**			**Performance**					
	Scale (B)	**Backbone**	**Tokens ↓**	**LH-1 ↑**	**LH-2 ↑**	**LH-3 ↑**	**LH-4 ↑**	**LH-5 ↑**	**Avg. Len. ↑**
SuSIE	–	–	–	87.0	69.0	49.0	38.0	26.0	2.69
VPP	1.5	SVD+CLIP	–	95.7	91.2	86.3	81.0	75.0	4.29
Seer	0.3	ViT+CLIP	–	96.3	91.6	86.1	80.3	74.0	4.29
OpenVLA	7.7	Llama-2 DINOv2+SigLIP CLIP	256	91.3	77.8	62.0	52.1	43.5	3.27
GR-1	0.195	MAE-ViT OpenFlamingo	–	85.4	71.2	59.6	49.7	40.1	3.06
RoboFlamingo	3	ViT	128	82.4	61.9	46.6	33.1	23.5	2.47
π_0^*	3.3	PaliGemma	512	70.0	48.0	37.0	28.0	18.0	2.01
$\pi_{0.5}^*$	3.3	PaliGemma	512	71.0	56.0	45.0	37.0	29.0	2.38
FLOWER	0.95	Florence-2-L	100	**99.3**	**96.0**	90.3	82.3	75.5	4.44
ThinkProprio	0.95	Florence-2-L	**12**	98.9	95.4	**91.6**	**86.4**	**79.1**	**4.52**

Table 11: CALVIN ABCD→D long-horizon success with architecture details.

Method	**Architecture**			**Performance**					
	Scale (B)	**Backbone**	**Tokens ↓**	**LH-1 ↑**	**LH-2 ↑**	**LH-3 ↑**	**LH-4 ↑**	**LH-5 ↑**	**Avg. Len. ↑**
Diff-P-CNN	0.32	–	–	86.3	72.7	60.1	51.2	41.7	3.16
RoboFlamingo	3.0	OpenFlamingo ViT	128	96.4	89.6	82.4	74.0	66.0	4.09
DeerVLA	3.0	OpenFlamingo MAE-ViT	128	99.1	93.3	82.1	74.6	63.8	4.13
GR-1	0.195	GPT	–	94.9	89.6	84.4	78.9	73.1	4.21
FLOWER	0.95	Florence-2-L	100	98.9	96.7	93.9	90.2	85.5	4.62
ThinkProprio	0.95	Florence-2-L	**15**	**99.5**	**97.2**	**96.6**	**92.3**	**88.5**	**4.74**

Table 12: CALVIN D→D long-horizon success with architecture details.

Method	**Architecture**			**Performance**					
	Scale (B)	**Backbone**	**Tokens ↓**	**LH-1 ↑**	**LH-2 ↑**	**LH-3 ↑**	**LH-4 ↑**	**LH-5 ↑**	**Avg. Len. ↑**
MDT	–	–	–	93.7	84.5	74.1	64.4	55.6	3.72
RoboUniView	–	–	–	96.2	88.8	77.6	66.6	56.3	3.85
ThinkProprio	0.95	Florence-2-L	**14**	**96.9**	**89.8**	**83.6**	**80.5**	**72.7**	**4.23**

LH-k is the success rate (%) of completing k consecutive subtasks in the five-subtask evaluation chain, and Avg. Len. is the mean number of consecutively completed subtasks. The architecture columns provide model scale, backbone, and visual-token count when available. In the ABCD→D setting, ThinkProprio achieves the best Avg. Len. (4.74) and the best LH-5 result (88.5). In D→D, ThinkProprio exceeds the remaining prior baselines reported in Table 12.

C.2 CALVIN Subtask Breakdown

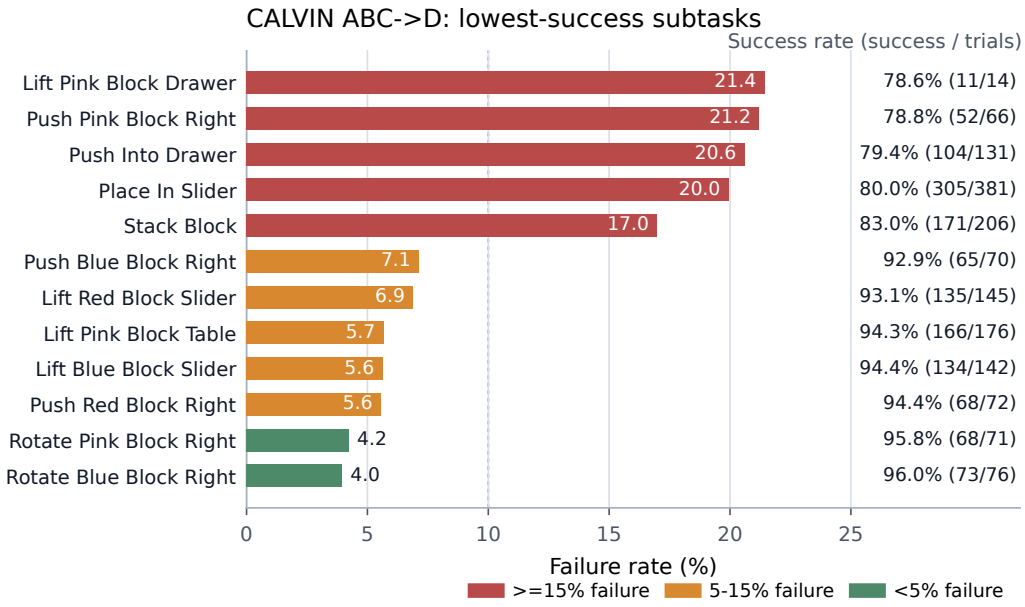


Figure 6: Subtask-level breakdown of ThinkProprio on CALVIN ABC→D. We plot the 12 lowest-success subtasks among the 34 CALVIN subtasks, ordered by failure rate. Bars report failure rate; right annotations report success rate with successful/evaluated rollout counts.

Figure 6 complements the chain-level CALVIN metrics by showing where the remaining ABC→D errors occur. Most subtasks are near saturation, so residual failures concentrate in a small set of contact-sensitive or placement-sensitive interactions. Only three subtasks fall below 80% success in this evaluation: *Lift Pink Block Drawer* (78.6%), *Push Pink Block Right* (78.8%), and *Push Into Drawer* (79.4%). The next hardest cases, such as *Place In Slider* and *Stack Block*, also require precise placement or sustained contact. This pattern is consistent with the main CALVIN result: ThinkProprio improves long-horizon chain completion, while the remaining failures are concentrated in interactions where small state-estimation or contact-control errors can accumulate. Section D.5 revisits the same contact-and-approach failure mode in real-world rollouts.

C.3 Long-Horizon Qualitative Rollouts

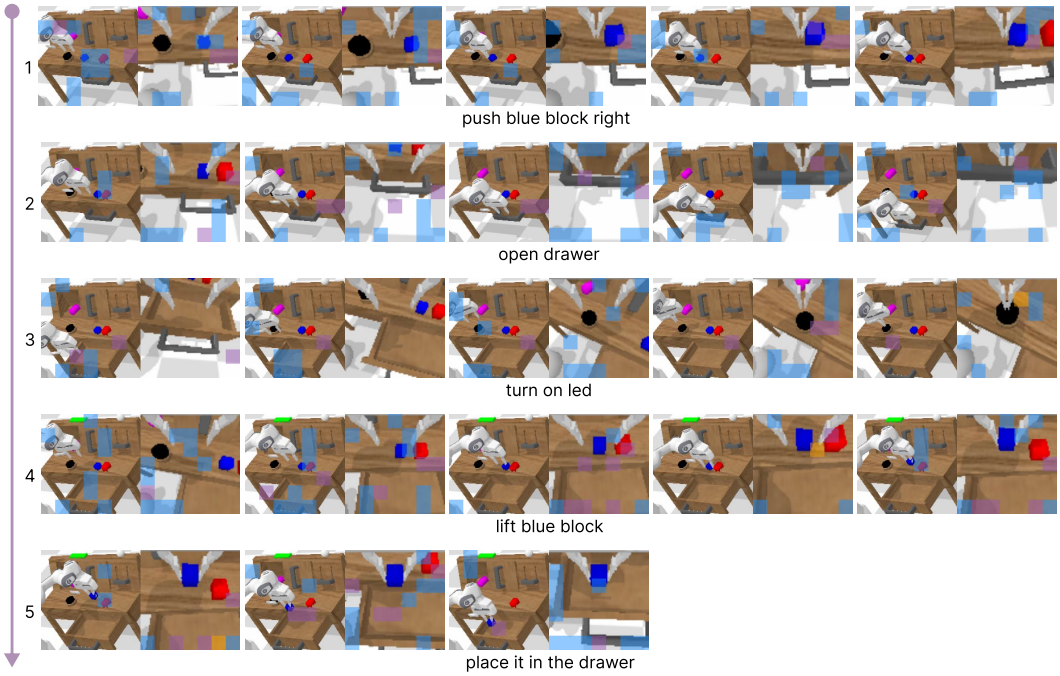


Figure 7: Qualitative CALVIN LH-5 rollout from ThinkProprio. Rows correspond to the five consecutive subtasks in the evaluation chain, and columns show representative timesteps within each subtask. Colored overlays visualize visual tokens retained by the instruction and proprioception guidance branches.

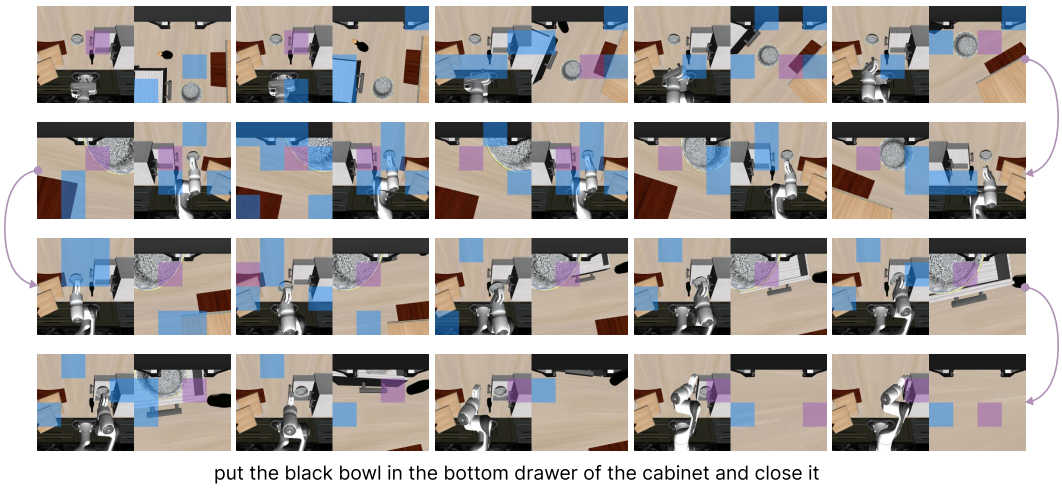


Figure 8: Qualitative LIBERO-Long rollout from ThinkProprio for placing the black bowl in the bottom drawer and closing it. Colored overlays visualize retained visual tokens as the policy transitions from reaching and grasping to drawer interaction and closing.

Figures 7 and 8 provide qualitative examples of the long-horizon settings where proprioceptive visual grounding is most useful. Across both CALVIN and LIBERO-Long, the retained patches move with the task phase: they cover task-referenced objects and target regions during approach, then shift toward the gripper, drawer, and contact regions during manipulation. These examples complement the aggregate long-horizon metrics by showing that token retention remains state-conditioned over multi-stage execution rather than staying fixed on the initial target object.

D Analysis and Ablations

D.1 Inference-Time Breakdown

Table 13: Mean per-environment-step latency in ms on CALVIN and LIBERO. Tokens reports the average retained visual-token count over the total available visual tokens.

Benchmark	Tokens	Vision	Selector	VLM	Action	Total
CALVIN ABC→D	12/100	5.8	0.1	0.9	15.4	22.2
LIBERO	7.8/34	6.5	0.1	1.1	16.1	23.8

Table 13 decomposes the end-to-end inference time used in Table 4. The token denominators reflect the two-view preprocessing resolutions: CALVIN uses 224×224 inputs and produces 100 visual tokens per timestep, whereas LIBERO uses 112×112 inputs and produces 34. The Vision column includes encoding both camera views. The selector adds only about 0.1 ms per step in both benchmarks, while the diffusion action head remains the dominant cost. The total latency is therefore reduced mainly by shortening the sequence processed by the VLM and consumed by the cross-attention action head, rather than by changing the action generator itself.

D.2 Proprioceptive Discretization Sensitivity

Table 14: Auxiliary LIBERO-Spatial sensitivity runs for proprioceptive discretization. We vary the number of bins and clipping range used before mapping state values to VLM-vocabulary token IDs.

Bins	Clip range	Success rate (%) ↑
32	$[-3, 3]$	94.5 ± 0.1
256	$[-3, 3]$	98.4 ± 0.2
512	$[-3, 3]$	98.4 ± 0.3
256	$[-1, 1]$	96.1 ± 0.9
256	$[-5, 5]$	96.2 ± 0.6

Table 14 shows that proprioceptive tokenization is sensitive to both quantization resolution and clipping range. Too few bins coarsen the state representation, and too narrow or too wide clipping ranges reduce useful variation in the resulting token IDs. Increasing from 256 to 512 bins does not improve performance in this setup, so we use 256 bins over $[-3, 3]$ as the default. These are the defaults used by the VLM-vocabulary proprioceptive encoding in Section 3.1 and by the observation/model setup in Sections B.1 and B.2. This supports the view that VLM-vocabulary proprioception is not simply a free replacement for continuous state: the discretization must retain enough resolution for the robot state while remaining stable under the normalized proprioceptive distribution.

D.3 Selection Consistency

Table 15 quantifies whether retained-token masks are structured rather than arbitrary. Each mask records the visual tokens kept in one forward pass, so higher IoU means greater overlap between selected token sets; the matched token budgets are 12/100 for CALVIN ABC→D and 7.8/34 for LIBERO-Spatial. Cross-scene and temporal IoU are both above the random baseline on CALVIN

Table 15: Intersection-over-Union (IoU) between binary retained-token masks. Cross-scene compares the same instruction across different initial states; Temporal compares neighboring rollout steps; Random is the expected IoU from uniform random retention with the matched token budget (12/100 for CALVIN ABC→D, 7.8/34 for LIBERO-Spatial).

IoU	CALVIN ABC→D	LIBERO Spatial
Cross-scene	0.170	0.644
Temporal	0.196	0.418
Random	0.064	0.130

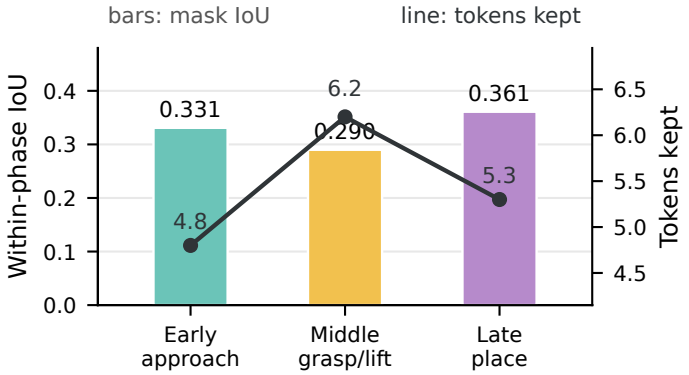


Figure 9: Phase-wise selector behavior on LIBERO-Spatial. Bars show within-phase mask IoU, and the line shows mean tokens kept. Phases are equal thirds of each rollout.

and LIBERO-Spatial, indicating that selection is task- and state-conditioned. The lower CALVIN cross-scene IoU reflects stronger variation across long-horizon configurations, while the higher LIBERO-Spatial IoU is consistent with the more constrained scene layouts in that suite.

D.4 Phase-Wise Selection Adaptation

Figure 9 shows that the selector is not static over an episode. The mean-token line peaks in the middle phase, and the middle-phase IoU bar is the lowest, consistent with grasping and lifting being the most visually dynamic portion of the task. Late-phase masks become more consistent as the robot moves toward the placement target, and the Early–Late IoU bar is approximately 0.29, indicating substantial cross-phase drift. This phase-wise view complements the qualitative rollouts in Figures 7 and 8, where retained tokens shift with objects, gripper motion, and contact regions over execution.

D.5 Qualitative Failure Cases

Figure 10 shows two representative real-world failures. In the tape case, the gripper reaches the correct object but contacts it with a misaligned approach, so the final manipulation fails despite the target being visually identified. In the corn case, the approach trajectory clips a nearby banana before establishing clean contact with the corn. These examples point to control-side and contact-side limitations, complementing the main-paper Limitations section, which focuses on the tabletop evaluation scope and deployment assumptions.

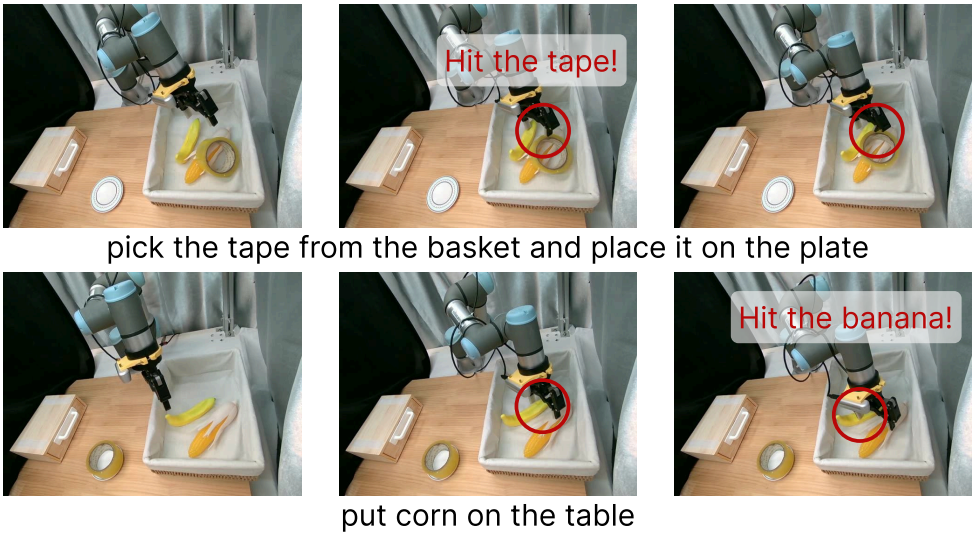


Figure 10: Representative real-world failure cases. Top: the gripper reaches the tape but makes misaligned contact. Bottom: the approach to the corn collides with a nearby banana.