PREMOVER: FAST VISION-LANGUAGE-ACTION CONTROL VIA EARLY EXECUTION DURING INSTRUCTION DELIVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) policies are typically evaluated under the assumption that the robot starts acting only after the user has finished typing or speaking. In real interactions, however, entering an instruction can take several seconds, leaving the policy idle for a substantial fraction of the interaction. Partial instructions may already contain sufficient information to begin acting before the full instruction arrives. We introduce **Premover**, a parameter-lightweight module that reduces interaction latency by overlapping instruction delivery with robot execution while keeping the VLA backbone frozen. Premover uses a learned *vision-language focus map* to identify *where* the current partial instruction refers in the visual scene, and an *action readiness gate* to determine *when* execution can begin. On the SO-101 robot with speech and online transcription, Premover reduces mean wall-clock time by 17.9%, from 60.5 s to 49.7 s, while maintaining task success rates. In simulation on LIBERO with $\pi_{0.5}$ at the speaking rate, Premover reduces mean wall-clock time by 8.3% while maintaining comparable success to waiting for the complete instruction before execution.

1 INTRODUCTION

Recent work on efficient Vision-Language-Action (VLA) models focuses on reducing inference latency once the complete instruction is available (Yue et al., 2024; Liu et al., 2026; Wen et al., 2025; Pertsch et al., 2025). In real-world deployment, however, the instruction takes time to arrive. A user may take several seconds to type or speak a request, during which the policy typically remains idle. This instruction-delivery interval accounts for approximately 54.6% of the wall-clock time across the four LIBERO suites at the speaking rate (Figure 1). Yet standard evaluation protocols (Liu et al., 2023; Zhang et al., 2026) largely ignore this delay by presenting the complete instruction before execution begins.

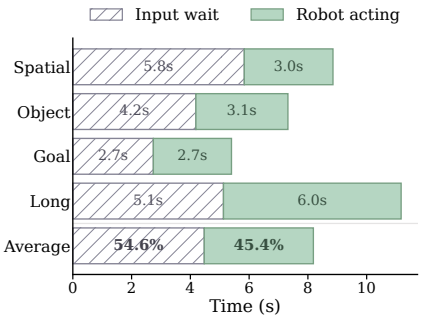


Figure 1: **Breakdown of waiting and action time on LIBERO.** Hatched bars show measured instruction delivery time; solid bars show measured robot action time at the speaking rate.

We ask whether some of this instruction-delivery interval can instead be used for robot execution. In our setting, instruction words arrive incrementally, and we refer to the portion received so far as the *streaming prefix*. Even a streaming prefix such as "Pick the tomato on top of the drawer…" may already contain sufficient information to begin execution, allowing instruction delivery and robot execution to overlap. The challenge is that beginning execution too early can lead the robot to take actions that are not yet supported by the current streaming prefix. This raises two questions. First, *where* does the current streaming prefix refer in the visual scene? Second, *when* does the streaming prefix contain sufficient information to begin execution?

We introduce **Premover**, a lightweight module built on a frozen VLA policy. Figure 2 contrasts Premover with conventional inference, where execution begins only after the complete instruction

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

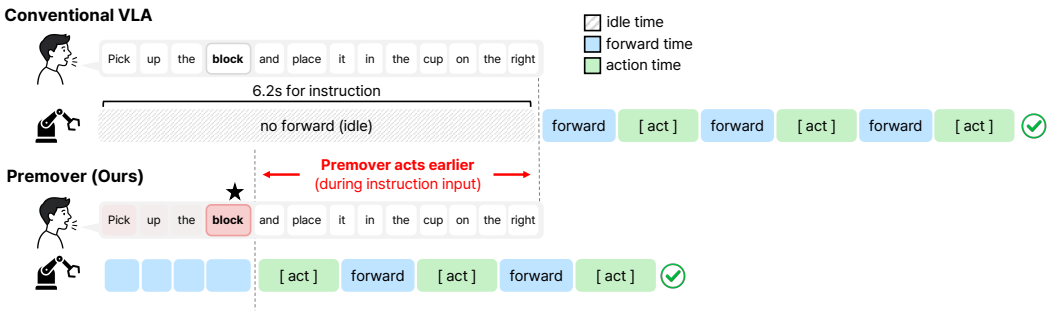


Figure 2: **Conventional VLA inference vs. Premover.** Conventional VLAs wait for the complete instruction before execution. Premover processes the streaming prefix as it arrives, using a vision-language focus map to determine where to focus and an action readiness gate to determine when to begin.

is received. Premover addresses these questions through two components: a *vision-language focus map* that identifies *where* the current streaming prefix refers in the visual scene, and an *action readiness gate* that determines *when* the prefix contains sufficient information to begin execution. Both components are trained jointly on features from the frozen VLA backbone. After execution begins, the instruction continues to arrive, and Premover updates its action prediction as the streaming prefix grows.

We evaluate Premover with two frozen VLA backbones in simulation, $\pi_{0.5}$ (Physical Intelligence et al., 2025) and GR00T-N1.7 (NVIDIA, 2026), on LIBERO (Liu et al., 2023) and VLA-Arena (Zhang et al., 2026). These benchmarks cover different task distributions, scene compositions, and instruction complexity. We evaluate both backbones under two instruction-delivery rates, a typing rate and a speaking rate, using representative rates estimated through the empirical analysis in Appendix D. We further deploy Premover with $\pi_{0.5}$ on an SO-101 robot using a TTS-ASR streaming pipeline at the speaking rate.

On the SO-101 robot at the speaking rate, Premover reduces mean wall-clock time by 17.9% relative to waiting for the complete instruction before execution, with mean success rates of 86.0% and 81.0%, respectively. On LIBERO with $\pi_{0.5}$ at the speaking rate, Premover reduces mean wall-clock time by 8.3%, from 8.17 to 7.49 seconds, while maintaining comparable success (97.6% versus 97.9%). Together, these results show that instruction-delivery time, a previously overlooked source of end-to-end latency, can be reduced by overlapping it with robot execution while preserving task performance.

2 Related Work

Vision-Language-Action (VLA) models adapt pretrained vision-language models to robotic control. Early approaches predict robot actions as autoregressive action-token prediction (Brohan et al., 2023; Kim et al., 2024), while more recent architectures combine vision-language backbones with diffusion- or flow-matching-based action experts to generate continuous action chunks (Chi et al., 2023; Zhao et al., 2023; Black et al., 2024; Physical Intelligence et al., 2025; NVIDIA et al., 2025; Shukor et al., 2025). Other work focuses on preserving or enriching the visual representations available to VLA policies. Knowledge Insulation (Driess et al., 2025) preserves pretrained vision-language representations by blocking gradients from the action expert, while RoboGround (Huang et al., 2025) and VAP (Lee et al., 2025) provide explicit spatial guidance through grounded masks or object-specific visual cues.

Most VLA methods assume that the complete instruction is available before execution begins. Earlier work on incremental language understanding in human-robot interaction instead allows a robot to process and act on an utterance before the speaker has finished (Schlangen & Skantze, 2009; Brick & Scheutz, 2007). These systems, however, rely on task-specific semantic parsers or hand-designed controllers rather than pretrained, frozen VLA backbones. Premover builds on this incremental setting while operating directly with frozen VLA policies.

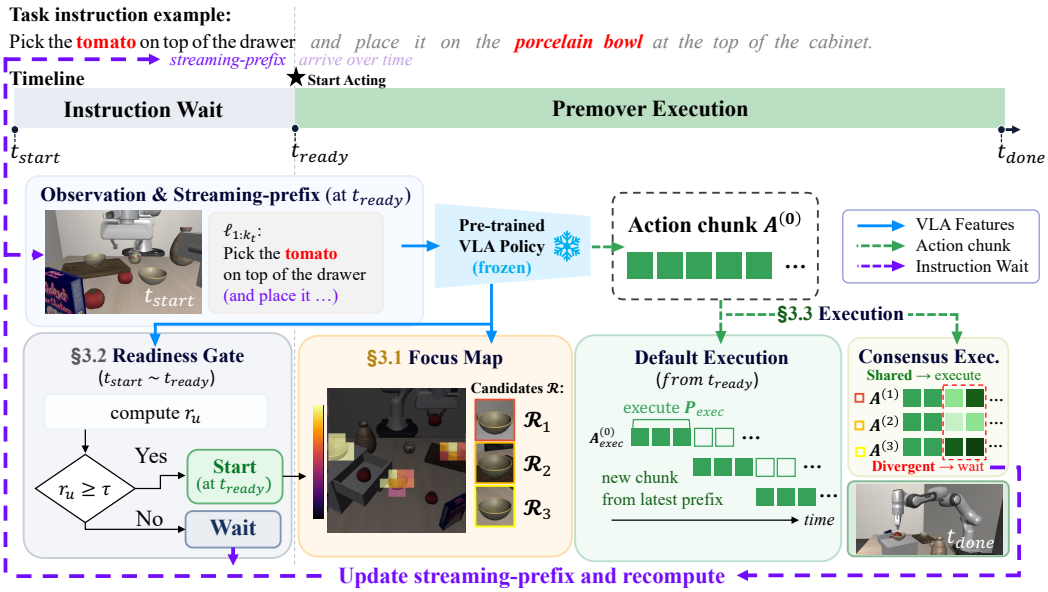


Figure 3: **Overview of Premover.** The vision-language focus map (§3.1) identifies image regions relevant to the current streaming prefix, while the action readiness gate (§3.2) determines when the prefix contains sufficient information to begin execution, defining t_{ready}. As additional words arrive, the policy updates its action prediction using the growing streaming prefix and continues execution as more information becomes available.

A separate line of work reduces VLA inference and execution latency through early exiting (Yue et al., 2024), adaptive visual-token pruning (Pei et al., 2026), compact multimodal backbones (Wen et al., 2025), more efficient action tokenization (Pertsch et al., 2025), or adaptive action-chunk execution (Wang et al., 2026). These approaches improve the efficiency of producing or executing each policy decision, but still primarily operate after the complete instruction is available. In contrast, Premover targets the instruction-delivery interval itself by beginning execution before the full instruction arrives, and can therefore be combined with methods that improve VLA inference or execution efficiency.

3 METHOD

Motivation and challenges. Conventional VLA inference assumes that the complete instruction is available before the robot begins execution. In real-world interaction, however, the robot remains idle for several seconds while the user enters an instruction. On LIBERO-Spatial at speaking rate, waiting for the complete instruction takes an average of 5.8 seconds, accounting for approximately 65.9% of the wall-clock time. Standard VLA evaluation protocols ignore this idle time. This raises a natural question: *can the policy begin execution before the user has finished entering the instruction?* We refer to the partial instruction observed so far as the *streaming prefix*.

Leveraging the streaming prefix requires resolving two challenges. *(i) Where to focus in the visual scene:* a streaming prefix such as "Pick the tomato on top of the drawer…" may already contain sufficient information to begin execution before the complete instruction arrives. However, a frozen VLA backbone does not directly localize the referent of the current streaming prefix, and its vision-language alignment can remain spread across irrelevant image regions (Appendix B). *(ii) When to start acting:* acting on a streaming prefix can cause the robot to grasp the wrong object or place it in the wrong location, leading to task failure. The policy must therefore determine when the current streaming prefix contains sufficient information to begin execution.

Method overview. Figure 3 illustrates Premover, a lightweight module that addresses these two challenges while keeping the VLA backbone frozen. The *vision-language focus map* identifies *where*

the current streaming prefix refers in the visual scene, while the *action readiness gate* determines *when* the streaming prefix contains sufficient information to begin execution. Once the action readiness gate determines that execution can begin, the robot executes the predicted action chunk while the instruction continues to arrive. If subsequent actions still depend on information that has not yet arrived, Premover can temporarily restrict execution by comparing action predictions across candidate regions.

Problem formulation. We consider a VLA setting in which the instruction arrives incrementally rather than being provided in full at the start. At step t, let $\ell_{1:k_t}$ denote the first k_t instruction words received so far. Tokenizing this streaming prefix yields L_t language tokens. Let N denote the total number of image patches across all input camera views. Given the streaming prefix, current image observations, and proprioceptive state, the policy predicts a default action chunk $A^{(0)} = (a_1^{(0)}, \ldots, a_M^{(0)})$ for the next M steps. The robot executes only the first $P_{\text{exec}} \leq M$ actions from each predicted action chunk. The key is to determine both where the streaming prefix refers in the visual scene and when it contains sufficient information to begin execution.

3.1 Vision-Language Focus Map

Beginning execution from a streaming prefix requires identifying which image regions the current streaming prefix refers to. We therefore introduce a supervised vision-language focus map that assigns a relevance score to each image patch based on the current streaming prefix.

Focus Score. Let h^b_{img} and h^b_{lang} denote the hidden states of image patches and language tokens extracted from backbone layer $b = 1, \ldots, B$. We use trainable projection heads f_{img} and f_{lang} to map the image and language features into a shared latent space. We assign a trainable layer-mixing logit γ_b to each backbone layer and obtain $\omega_b = \text{softmax}(\gamma)_b$. The same layer weights ω_b are shared across the image and language modalities:

$$
\begin{aligned}
z_{\text{img}} &= \text{norm}\left(\sum_{b=1}^{B} \omega_b f_{\text{img}}(h^b_{\text{img}})\right), \\
z_{\text{lang}} &= \text{norm}\left(\sum_{b=1}^{B} \omega_b f_{\text{lang}}(h^b_{\text{lang}})\right).
\end{aligned}
\tag{1}
$$

The projection functions and ℓ_2 normalization are applied independently to each image patch and language token.

For each image patch i, we compare its projected feature $z_{\text{img},i}$ with each token feature $z_{\text{lang},j}$, where j indexes the language tokens in the streaming prefix, and retain the strongest match:

$$
p_i = \sigma\left(\beta \max_{1 \leq j \leq L_t} \langle z_{\text{img},i}, z_{\text{lang},j} \rangle\right), \tag{2}
$$

where σ is the sigmoid function and β is a fixed similarity scale. The resulting $p = (p_1, \ldots, p_N)$ is the vision-language focus map. Each p_i, which we call the *focus score*, measures the strongest vision-language match between image patch i and the tokens in the current streaming prefix.

To condition the action prediction on a region selected by the focus map, we convert each selected region R into soft visual weights, $w_i = \alpha + (1 - \alpha)\mathbf{1}[i \in R]$. Patches inside R receive full weight, while patches outside are suppressed to weight $\alpha \in [0, 1)$ rather than discarded. The policy then predicts the action chunk using these reweighted visual features.

The focus map is supervised using two objectives. The focus-map loss ($\mathcal{L}_{\text{focus}}$) matches individual patch focus scores to segmentation-mask labels. The region-contrastive loss ($\mathcal{L}_{\text{contrast}}$) assigns higher mean focus scores to prefix-relevant regions than to other visible regions. Further details are provided in Appendix A.

3.2 Action Readiness Gate

The action readiness gate determines whether the current streaming prefix contains sufficient information to begin execution. Let $e_u \in \{0, 1\}$ denote the binary readiness decision at streaming prefix update u. Here, $e_u = 1$ means that the current streaming prefix is sufficiently informative to support an initial action, whereas $e_u = 0$ indicates that more information is needed before execution can begin.

Readiness score. Let x_u denote a representation of the current observation and streaming prefix, built from backbone hidden states. A two-layer readiness predictor f_{ready} maps x_u to a scalar logit, from which we compute:

$$r_u = \sigma(f_{\text{ready}}(x_u)), \qquad e_u = \mathbf{1}[r_u \geq \tau], \tag{3}$$

where σ is the sigmoid function and τ is a learned threshold. Execution begins when $e_u{=}1$ at two consecutive streaming-prefix updates with distinct word counts. We jointly train f_{ready} and τ using the readiness loss, $\mathcal{L}_{\text{ready}}$, a weighted binary cross-entropy objective with prefix-level labels derived from a *readiness cutoff* annotated by a language model, defined as the earliest word sufficient to identify the target and support the initial action. Backbone-specific feature aggregation, label construction, and the detailed training objective are provided in Appendix A.

3.3 Execution

Once the action readiness gate determines that the current streaming prefix contains sufficient information to begin execution, the robot begins acting from the default action chunk $A^{(0)}$ before the complete instruction arrives. As additional words arrive, the policy updates its action prediction using the current observation and updated streaming prefix.

Default action execution. By default, the robot executes a fixed number of actions from each predicted action chunk. Given the default action chunk $A^{(0)} = (a_1^{(0)}, \ldots, a_M^{(0)})$, we define the default execution length as

$$m_{\text{exec}} = \min(P_{\text{exec}}, M) = P_{\text{exec}}, \tag{4}$$

where the final equality follows from $P_{\text{exec}} \leq M$. The robot then executes the first m_{exec} actions,

$$A_{\text{exec}}^{(0)} = A_{1:m_{\text{exec}}}^{(0)} = \left(a_1^{(0)}, \ldots, a_{m_{\text{exec}}}^{(0)}\right). \tag{5}$$

After executing these actions, the policy predicts a new action chunk using the current observation and the latest streaming prefix. Under default execution, each prediction therefore provides up to P_{exec} actions before replanning. If the current streaming prefix does not yet provide enough information for later actions, Premover can execute fewer than P_{exec} actions and wait for additional instruction. We handle this case with consensus-guided execution, described below.

Consensus-guided execution. Consensus-guided execution is used when later actions still depend on information that has not yet arrived. For example, "pick the tomato on top of the drawer and ..." identifies the object but not its destination (Figure 8). Premover extracts J candidate regions from the focus map, obtains a region-conditioned action chunk $A^{(d)}$ for each using the reweighting in Section 3.1, and executes only the initial actions shared across these chunks. Let $S(\mathcal{A})$ denote the length of this shared action prefix, and let $m_{\text{exec}}^{\text{cons}}$ denote the number of actions executed under consensus-guided execution:

$$m_{\text{exec}}^{\text{cons}} = \min\left(S\left(A^{(1)}, \ldots, A^{(J)}\right), C, P_{\text{exec}}\right), \tag{6}$$

where C limits the number of actions executed under consensus-guided execution. Further details are provided in Appendix C.

3.4 Training Objective

We jointly train the image and language projection heads, shared layer-mixing logits γ, and the action readiness gate using

$$\mathcal{L} = \mathcal{L}_{\text{focus}} + \mathcal{L}_{\text{contrast}} + \mathcal{L}_{\text{ready}}, \tag{7}$$

Table 1: **Real-robot evaluation with spoken instructions**. On five single-object tasks, we report full-task success rate (SR; %), while on three disambiguation tasks, we report partial success score (%). We also report mean wall-clock time (Wall; s), with 20 trials per task and method and a 900-step limit. Avg. reports the mean across tasks within each group.

	Single-object						Disambiguation			
Method	**Block→Cup**	**Drawer**	**Cat→Cup**	**Cup→Cup**	**Push Cup**	**Avg.**	**On Cup**	**Gray Cat**	**Orange Cat**	**Avg.**
(a) Success rate (%) ↑										
Full-prompt	65.0	70.0	95.0	90.0	85.0	81.0	52.5	55.0	65.0	57.5
Right-away	50.0	55.0	85.0	70.0	70.0	66.0	25.0	45.0	35.0	35.0
Noun-triggered	40.0	55.0	65.0	40.0	75.0	55.0	32.5	30.0	40.0	34.2
Premover (Ours)	75.0	80.0	90.0	95.0	90.0	86.0	57.5	90.0	67.5	71.7
(b) Wall-clock time (s) ↓										
Full-prompt	66.6	79.1	48.4	53.5	54.9	60.5	83.6	79.3	101.8	88.2
Right-away	60.7	101.4	48.6	54.0	63.0	65.5	83.8	80.0	101.3	88.4
Noun-triggered	75.2	97.2	57.7	88.4	65.7	76.8	96.8	137.2	99.0	111.0
Premover (Ours)	49.0	73.2	38.8	32.9	54.7	49.7	77.8	71.0	81.2	76.7

where $\mathcal{L}_{\text{focus}}$ provides patch-level focus supervision, $\mathcal{L}_{\text{contrast}}$ separates prefix-relevant regions from other visible regions, and $\mathcal{L}_{\text{ready}}$ supervises the readiness decision. The vision-language focus map and action readiness gate are trained jointly, while the VLA backbone remains frozen. Training settings, loss details, data splits, and optimization hyperparameters are provided in Appendix A.

4 Experiments

4.1 Experimental Setup

Evaluation settings. We evaluate Premover in both simulation and on a real robot with streaming instructions. Under matched configurations, we compare *Premover* with *Full-prompt*, which begins execution after the complete instruction arrives; *Right-away*, which begins execution immediately; and *Noun-triggered*, which begins execution when the target noun arrives. For the speaking condition, both simulation and real-robot experiments use TTS-generated speech with ASR transcripts. We report success rate (SR) for simulation and real-robot single-object tasks, partial success scores for real-robot disambiguation tasks, and mean wall-clock time over all episodes (Wall). Protocol and timing details are provided in Appendix D.

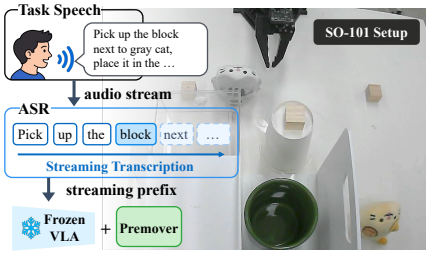


Figure 4: **Real-robot system pipeline.** An ASR model streams partial transcripts, which Premover combines with robot observations to produce action chunks before the full instruction arrives.

Real-robot environments. We evaluate the frozen $\pi_{0.5}$ backbone (Physical Intelligence et al., 2025) on an SO-101 robot (Figure 4). The evaluation comprises five single-object tasks and three disambiguation variants of a block-to-drawer task. Single-object tasks require moving one object to a destination, while disambiguation tasks require selecting the instructed object from multiple identical objects and moving it to a destination. A frozen Segment Anything Model (SAM) (Kirillov et al., 2023) provides offline mask supervision for the vision-language focus map. Details are provided in Appendix F.

4.2 Real-Robot Results

Table 1 shows real-robot results at the speaking rate. Across the five single-object tasks, Premover reduces mean wall-clock time by 17.9%, from 60.5 to 49.7 s; the corresponding mean success rates are 86.0% for Premover and 81.0% for Full-prompt. Disambiguation tasks require distinguishing the target from multiple identical objects using spatial descriptions that arrive after the object noun. On these tasks, Premover reduces mean wall-clock time from 88.2 to 76.7 s while maintaining task performance, with a mean success score of 71.7% compared with 57.5%. Premover achieves lower wall-clock times and higher success scores than all baselines on every disambiguation task.

Table 2: **Main results on LIBERO with $\pi_{0.5}$.** Success rate (SR; %) and wall-clock time (Wall; s) across the four LIBERO suites at both instruction-delivery rates. The two rates correspond to representative typing and speaking speeds (Appendix D). Entries are means over three evaluation seeds, with seed-level standard deviations reported for both SR and Wall.

	Spatial		Object		Goal		Long-10		Mean	
Method	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓
Typing rate										
Full-prompt	99.1±0.3	20.27±0.02	99.3±0.3	14.70±0.02	97.7±0.3	10.14±0.01	94.3±0.9	20.59±0.05	97.6±0.4	16.42±0.02
Right-away	70.9±1.5	9.46±0.35	73.2±1.3	7.60±0.23	56.4±2.2	10.08±0.21	85.2±1.3	9.34±0.23	71.4±0.7	9.12±0.12
Noun-triggered	86.0±1.4	10.18±0.22	99.6±0.2	7.56±0.02	79.9±1.0	8.34±0.08	95.0±1.2	10.87±0.14	90.1±0.3	9.24±0.03
Premover	97.6±1.0	16.44±0.16	98.8±0.4	11.83±0.13	98.0±0.9	9.88±0.06	95.1±1.9	16.61±0.17	97.4±0.5	13.69±0.02
Speaking rate										
Full-prompt	98.7±0.3	8.84±0.01	99.6±0.4	7.31±0.02	97.9±1.3	5.45±0.06	95.4±1.0	11.07±0.09	97.9±0.1	8.17±0.02
Right-away	77.0±1.2	6.10±0.07	65.0±1.1	6.62±0.05	64.6±1.8	6.36±0.09	86.0±0.5	8.39±0.03	73.1±0.4	6.87±0.02
Noun-triggered	84.4±0.7	6.89±0.06	99.6±0.3	5.64±0.02	90.3±1.0	4.96±0.08	95.7±0.3	8.58±0.04	92.5±0.1	6.52±0.02
Premover	98.2±0.6	7.78±0.04	99.2±0.2	6.54±0.02	97.6±1.2	5.45±0.07	95.2±2.1	10.19±0.08	97.6±0.5	7.49±0.02

Table 3: **Evaluation across backbone-benchmark combinations at the speaking rate.** Success rate (SR; %) and wall-clock time (Wall; s) for the remaining backbone-benchmark combinations at speaking rate. Table 2 covers $\pi_{0.5}$ on LIBERO. Entries are means over three evaluation seeds, with seed-level standard deviations reported for both SR and Wall. Complete results across both instruction-delivery rates and all benchmark suites are provided in Appendix E.

	GR00T-N1.7 / LIBERO		$\pi_{0.5}$ / VLA-Arena		GR00T-N1.7 / VLA-Arena	
Method	**SR ↑**	**Wall ↓**	**SR ↑**	**Wall ↓**	**SR ↑**	**Wall ↓**
Full-prompt	94.0±0.3	8.60±0.03	64.3±0.8	10.37±0.05	48.6±1.1	8.12±0.05
Right-away	57.7±0.9	8.58±0.05	60.7±0.7	8.93±0.06	40.0±0.3	7.27±0.01
Noun-triggered	77.2±0.8	7.95±0.06	63.5±0.7	8.97±0.04	44.4±0.8	7.18±0.05
Premover	93.8±0.5	8.07±0.05	63.7±1.3	9.63±0.08	48.3±0.7	7.64±0.03

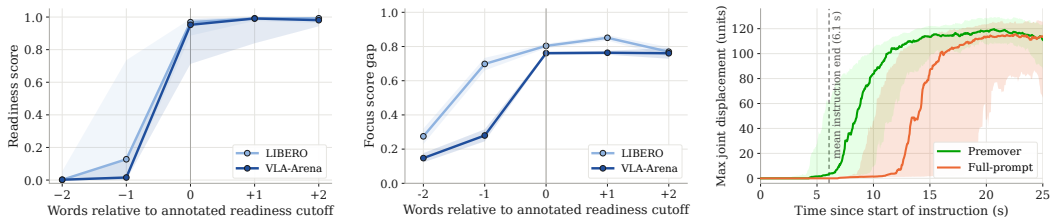
The real-robot results further highlight the role of the action readiness gate. With online transcription and noisier visual observations, delaying execution until the streaming prefix contains sufficient information can reduce the risk of acting on incomplete or misrecognized instructions, helping avoid the large success drop observed with Right-away. After execution begins, the focus map can influence subsequent action predictions and execution trajectories, which may also contribute to the higher success rates.

4.3 Simulation Results

Across all four backbone-benchmark combinations at the speaking rate, Premover reduces mean wall-clock time by 5.9-8.3% relative to Full-prompt while keeping success within 0.6 percentage points of the corresponding baseline (Tables 2 and 3). The consistent reduction across two VLA backbones and two benchmarks shows that the latency benefit of streaming execution extends beyond a single model or task distribution. Although Right-away and Noun-triggered can further reduce wall-clock time in some settings, they often incur substantial drops in success, highlighting the importance of determining when execution can safely begin. Full results for both instruction-delivery rates are provided in Appendix E.

4.4 Analysis

We analyze how Premover's action readiness gate and vision-language focus map behave around the annotated readiness cutoff. Figure 5(a) shows readiness scores on LIBERO and VLA-Arena, averaged over $\pi_{0.5}$ and GR00T-N1.7. The scores remain low before the cutoff and rise sharply around it, indicating that the gate responds to when the streaming prefix becomes sufficient to support the initial action rather than to a fixed amount of observed language. Figure 5(b) examines the vision-language focus map using the gap between the mean focus scores of the top-K patches ($K = 16$) and the remaining patches. The gap increases sharply around the readiness cutoff on both benchmarks,



(a) Readiness gate (b) Focus score gap (c) Physical motion, real robot

Figure 5: **Analysis of early execution.** (a) Readiness scores, averaged over $\pi_{0.5}$ and GR00T-N1.7, increase around the annotated readiness cutoff on LIBERO and VLA-Arena. (b) Mean focus-score gap between the top-16 and remaining patches, averaged over both backbones. The gap widens around the cutoff. (c) Averaged across all real-world tasks, Premover begins measurable joint motion before Full-prompt, including before instruction delivery finishes.

showing that the focus map becomes more spatially concentrated as the instruction provides sufficient information to act.

We then examine whether early execution translates into physical behavior on the real robot. Fig. 5(c) shows the maximum joint displacement from the starting configuration, averaged across all real-world tasks. Premover begins moving substantially earlier than Full-prompt, including before instruction delivery is complete, showing that early execution translates into earlier physical motion. The same trend holds for $K \in \{8, 16, 32\}$, with the sensitivity analysis and the definition of the joint-displacement metric provided in Appendix G.

4.5 Ablation Studies

4.5.1 Effect of Instruction-Delivery Rate

We vary the instruction-delivery rate from 25 to 200 WPM at 25-WPM intervals using $\pi_{0.5}$ on LIBERO, with streaming prefixes arriving according to each delivery rate. Wall-clock time includes both instruction delivery and task execution, accounting for any overlap between them.

Figure 6 shows that Premover consistently maintains Full-prompt-level success across all tested instruction-delivery rates, with success rates between 96.8-98.1%. At the same time, Premover reduces wall-clock time relative to Full-prompt at every delivery rate, by 13-17%. These results show that Premover preserves task performance while consistently reducing end-to-end interaction time across a wide range of instruction-delivery rates.

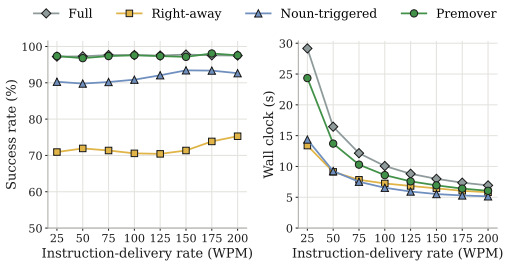


Figure 6: **Effect of instruction-delivery rate.** Premover maintains high success rates and reduces wall-clock time across instruction-delivery rates.

4.5.2 Component Ablation

To isolate the contribution of each component, we ablate the action readiness gate and vision-language focus map using GR00T-N1.7 on LIBERO at the speaking rate. Table 4 summarizes the resulting changes in success rate and wall-clock time. Starting from the streaming-only baseline, adding the action readiness gate raises success from 58.4% to 92.2%, accounting for most of the improvement over Right-away.

Table 4: **Component ablation.** Success rate (SR; %) and wall-clock time (Wall; s) as the action readiness gate and vision-language focus map are added to the streaming baseline.

Streaming	Ready	Focus	SR ↑	Wall ↓
✓			58.4	8.60
✓	✓		92.2	8.01
✓	✓	✓	93.8	8.12

Adding the vision-language focus map further increases success to 93.8%. Overall, the results show that the action readiness gate contributes most of the improvement in success, while the focus map provides an additional gain that recovers nearly all of the remaining gap to Full-prompt.

4.5.3 FIXED-FRACTION EXECUTION BASELINES

Table 5 shows the fixed-fraction results using GR00T-N1.7 on LIBERO at the speaking rate. The fixed baselines start execution after receiving 0%, 25%, 50%, 75%, or 100% of the instruction, where Fixed 0% and 100% correspond to *Right-away* and *Full-prompt*, respectively. Because these baselines depend on the fraction of the complete instruction, this controlled comparison uses the known full instruction to determine each execution point.

Table 5: **Fixed-fraction execution baselines.** Success rate (SR; %) and wall-clock time (Wall; s) when execution begins after a fixed fraction of the instruction has been received.

Setting	SR ↑	Wall ↓
Fixed 0% (Right-away)	58.4	8.60
Fixed 25%	69.3	8.07
Fixed 50%	85.8	7.79
Fixed 75%	93.6	8.30
Fixed 100% (Full-prompt)	93.7	8.63
Premover	93.8	8.12

Starting earlier allows more of the instruction-delivery interval to overlap with execution, but can substantially reduce success. A fixed-fraction strategy also requires knowing the full instruction length in advance. Premover, however, determines when the current streaming prefix contains sufficient information to begin execution, and thus reduces wall-clock time while maintaining near-Full-prompt success.

5 DISCUSSION AND FUTURE WORK

The benefit of Premover depends on how much of the instruction-delivery interval can be safely overlapped with robot execution. When the information needed to identify the target and support the initial action appears early in the instruction, Premover has more opportunity to begin execution before the complete instruction arrives. In contrast, the latency benefit can be smaller when the relevant information appears late, when instructions are delivered quickly, or when little execution can be completed before the remaining instruction arrives. Overall, the achievable latency reduction depends on the temporal structure of the instruction and the point at which the streaming prefix contains sufficient information to support execution.

Our current setting assumes that the intended task remains fixed as the instruction is revealed. We do not consider corrections or changes to previously specified targets or destinations after execution has begun, which would require the robot to reconsider actions already taken under an earlier prefix. Extending streaming execution to handle such revisions is an important direction for future work. Premover also relies on offline supervision for the vision-language focus map and action readiness gate, although this supervision is generated automatically using a frozen segmentation model and a language model rather than manual annotation.

6 CONCLUSION

We introduced Premover, a lightweight module that enables frozen VLA policies to begin execution before the complete instruction arrives. Premover uses a vision-language focus map to identify where the current streaming prefix refers in the visual scene and an action readiness gate to determine when the prefix contains sufficient information to begin execution. At the speaking rate, Premover reduces mean wall-clock time by 8.3% on LIBERO with $\pi_{0.5}$ while maintaining comparable success (97.6% versus 97.9%), and by 17.9% on the SO-101 robot while maintaining task success rates (86.0% versus 81.0%). These results show that overlapping instruction delivery with robot execution can reduce end-to-end interaction time while preserving task performance, without retraining the VLA backbone.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, improving the clarity of the manuscript, and code organization and refactoring. All technical content, experimental design, implementation decisions, analysis, and conclusions were developed and verified by the authors. The authors reviewed all AI-assisted edits and take full responsibility for the final manuscript and code.

REPRODUCIBILITY STATEMENT

Section 3 describes the vision-language focus map, the action readiness gate, the execution procedure, and the training objective, and Section 4.1 gives the evaluation settings, baselines, and real-robot setup. The appendices hold the remaining details. Appendix A covers training procedures, hyperparameters, supervision construction, and evaluation seeds. Consensus-guided execution is specified in Appendix C. Appendix D explains how instructions are delivered (rates and streaming protocols) and how wall-clock time is measured, and Appendix F lists the real-robot tasks, prompts, and data collection. All simulation experiments use public benchmarks (LIBERO and VLA-Arena) with frozen backbones ($\pi_{0.5}$ and GR00T-N1.7). We will release the code, trained heads, and streaming evaluation scripts upon acceptance.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. doi: 10.48550/arXiv.2410.24164.

Timothy Brick and Matthias Scheutz. Incremental natural language processing for HRI. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2007. URL `https://hrilab.tufts.edu/publications/brickscheutz07hri.pdf`.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. Observations on typing from 136 million keystrokes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018. doi: 10.1145/3173574.3174220.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. Spotlight.

Haifeng Huang, Xinyi Chen, Yilun Chen, Hao Li, Xiaoshen Han, Zehan Wang, Tai Wang, Jiangmiao Pang, and Zhou Zhao. RoboGround: Robotic manipulation with grounded vision-language priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22540–22550, 2025.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Grace Tin Yan Lam, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *ArXiv*, abs/2406.09246, 2024. URL https://api.semanticscholar.org/CorpusID:270440391.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, 2023. doi: 10.1109/ICCV51070.2023.00371.

Sangoh Lee, Sangwoo Mo, and Wook-Shin Han. Bring my cup! personalizing vision-language-action models with visual attentive prompting. *arXiv preprint arXiv:2512.20014*, 2025. doi: 10.48550/arXiv.2512.20014.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference, 2026. URL https://arxiv.org/abs/2511.16449.

NVIDIA. NVIDIA Isaac GR00T N1.7-3B model card. https://huggingface.co/nvidia/GR00T-N1.7-3B, 2026. Accessed: 2026-08-08.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ea6j8k8Rnw.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Robotics: Science and Systems (RSS)*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. doi: 10.48550/arXiv.2504.16054.

David Schlangen and Gabriel Skantze. A general, abstract model of incremental dialogue processing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2009)*, pp. 710–718, Athens, Greece, 2009. Association for Computational Linguistics. URL https://aclanthology.org/E09-1081/.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025. doi: 10.48550/arXiv.2506.01844.

Irina-Elena Veliche, Zhuangqun Huang, Vineeth Ayyat Kochaniyan, Fuchun Peng, Ozlem Kalinli, and Michael L. Seltzer. Towards measuring fairness in speech recognition: Fair-Speech dataset. *arXiv preprint arXiv:2408.12734*, 2024. doi: 10.48550/arXiv.2408.12734.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Ramana Rao Kompella, and Gaowen Liu. Vla knows its limits: Adaptive execution horizons for robot policies. In *European Conference on Computer Vision*, pp. 607–623. Springer, 2026.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. TinyVLA: Toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10:3988–3995, 2025.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 56619–56643, 2024.

Borong Zhang, Jiahao Li, Jiachen Shen, Yuhao Zhang, Yishuai Cai, Yuanpei Chen, Juntao Dai, Jiaming Ji, and Yaodong Yang. VLA-arena: An open-source framework for benchmarking vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qjeCH95rbF.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems (RSS)*, 2023.

A Implementation Details

A.1 Training Setup

Trainable parameters and optimization. The trainable parameters are the image and language projection heads f_{img} and f_{lang}, the shared layer-mixing logits γ, the readiness predictor f_{ready}, and the readiness threshold τ. The VLA backbone remains frozen throughout training.

The projection heads are two-layer GELU MLPs with hidden width 512 and output width 256, and the readiness predictor f_{ready} is a two-layer MLP with hidden width 256. The similarity scale in Eq. (2) is fixed to $\beta = 6$. The final layer of f_{ready} is initialized to zero, corresponding to an initial readiness score of 0.5, and τ is initialized to 0.1.

We optimize all trainable parameters with AdamW using a learning rate of 10^{-4}, weight decay of 10^{-4}, and global gradient-norm clipping at 1. The focus-map loss $\mathcal{L}_{\text{focus}}$, region-contrastive loss $\mathcal{L}_{\text{contrast}}$, and readiness loss $\mathcal{L}_{\text{ready}}$ in Eq. (7) each have unit weight.

Training data and prefix sampling. For each frozen backbone-benchmark pair, we train a separate set of heads across the selected tasks. Demonstrations 0–9 are used for training, with up to 16 sampled frames per demonstration and up to 8 streaming prefixes per frame. Prefix sampling includes the complete instruction and, when available, the last not-ready and first ready prefixes. Training makes one pass over the sampled examples, with one optimizer update per successfully loaded example.

Demonstrations 10–14 are used for validation, and initial states 15–49 are reserved for rollout evaluation. The evaluated backbones are $\pi_{0.5}$ (Physical Intelligence et al., 2025) and GR00T-N1.7 (NVIDIA et al., 2025; NVIDIA, 2026).

A.2 Vision-Language Focus Map Supervision

Patch-level focus supervision. Simulator instance masks provide offline supervision for the vision-language focus map. Each mask is projected onto the corresponding camera patch grid, and patches with instance coverage greater than 0.02 are included in the binary supervision mask.

The supervised region follows the task phase. The target-object mask is used through the first positive gripper-close command in the demonstration and the following three frames, after which supervision

switches to the destination mask. Only regions identified by the current streaming prefix are included. If the current streaming prefix has not yet identified the region for the corresponding task phase, the supervision mask is all zero.

Let $m^\star \in \{0,1\}^N$ denote the binary patch-level supervision mask. Each label $m_i^\star$ indicates whether image patch i belongs to the supervised target-object or destination region. The focus-map loss is

$$\mathcal{L}_{\text{focus}} = -\frac{\sum_{i=1}^{N} v_i \left[m_i^\star \log p_i + (1 - m_i^\star)\log(1 - p_i)\right]}{\max\left(\sum_{i=1}^{N} v_i, 1\right)}, \qquad v_i = \begin{cases} N_- / \max(N_+, 1), & m_i^\star = 1, \\ 1, & m_i^\star = 0, \end{cases} \tag{8}$$

where $N_+ = \sum_i m_i^\star$ and $N_- = N - N_+$. All-zero masks are retained during training so that the focus map assigns low scores when the current streaming prefix has not yet identified a relevant region.

Region-contrastive supervision. The positive region is the union of the target-object and destination regions identified by the current streaming prefix. Each negative region corresponds to another visible object and includes all patches with nonzero coverage by its segmentation mask. A negative region is excluded when more than half of its patches overlap the positive region.

The region-contrastive loss encourages a higher mean focus score in prefix-relevant regions than in other visible regions. Let μ_0 denote the mean focus score in the positive region and μ_r the mean focus score in negative region r, with J_- denoting the number of remaining negative regions. We use

$$\mathcal{L}_{\text{contrast}} = -\log \frac{\exp(\mu_0/T_c)}{\exp(\mu_0/T_c) + \sum_{r=1}^{J_-} \eta_r \exp(\mu_r/T_c)}, \qquad T_c = 0.1. \tag{9}$$

For each negative region, we set $\eta_r = 4$ if its object is explicitly excluded by the current streaming prefix, $\eta_r = 2$ if the object has not yet been named, and $\eta_r = 1$ otherwise. Explicit exclusion takes precedence when both conditions apply. The loss is set to zero when the positive region is empty or no negative regions remain.

A.3 Action Readiness Gate Details

Readiness representation. The readiness predictor uses frozen backbone hidden states before the image and language projection heads. At streaming-prefix update u, let t denote the corresponding control step.

For $\pi_{0.5}$, we mean-pool the image patches and visible language tokens within each backbone layer, aggregate them using the shared layer weights ω_b from Eq. (1), normalize the image and language representations independently, and concatenate them:

$$x_u = \left[\text{norm}\left(\sum_{b=1}^{B} \omega_b \frac{1}{N} \sum_{i=1}^{N} h_{\text{img},i}^b\right); \text{norm}\left(\sum_{b=1}^{B} \omega_b \frac{1}{L_t} \sum_{j=1}^{L_t} h_{\text{lang},j}^b\right)\right]. \tag{10}$$

Padding tokens are excluded from the language mean.

For GR00T-N1.7, we use the normalized layer-weighted hidden state of the last visible language token:

$$x_u = \text{norm}\left(\sum_{b=1}^{B} \omega_b h_{\text{lang},L_t}^b\right). \tag{11}$$

Both representations are passed to the readiness predictor f_{ready} in Eq. (3).

Readiness supervision. We use Qwen3.5-2B to annotate an offline readiness cutoff from the complete instruction and scene information. The cutoff is the earliest instruction word at which the streaming prefix contains sufficient information to identify the target and support the initial action. When target identification requires a prepositional phrase, the cutoff is placed at the final word of that phrase.

Let c denote the one-based index of the annotated readiness word. At streaming-prefix update u, a prefix containing k_t words receives the readiness label

$$y_u = \mathbf{1}[k_t \geq c]. \tag{12}$$

Prefixes before c are labeled not ready, while prefixes at or beyond c are labeled ready.

To jointly optimize the readiness score and learned threshold, we define

$$q_u^{\text{ready}} = \sigma\left(\frac{r_u - \tau}{0.1}\right), \tag{13}$$

and use the class-weighted binary cross-entropy loss

$$\mathcal{L}_{\text{ready}} = -w_{y_u}\left[y_u \log q_u^{\text{ready}} + (1 - y_u)\log(1 - q_u^{\text{ready}})\right], \qquad w_1 = \frac{1}{2\rho}, \quad w_0 = \frac{1}{2(1-\rho)}, \tag{14}$$

where ρ is the fraction of ready prefixes and is clipped away from zero and one.

Execution start. The binary readiness decision e_u is evaluated at streaming-prefix updates. Execution begins when $e_u = 1$ at two consecutive updates with distinct word counts. Repeated evaluation of the same streaming prefix does not advance this condition. We denote the resulting execution-start time by t_{ready}. Before t_{ready}, the robot holds its pose.

A.4 Simulation Evaluation Setup

Evaluation rollouts. The main simulation results are averaged over three evaluation seeds (0, 1, and 2), while all ablation studies use seed 0. For each evaluation seed, we run 35 rollouts per task from initial states 15–49, corresponding to 350 rollouts per LIBERO suite and 1400 rollouts across the four suites.

Full-prompt, Right-away, Noun-triggered, and Premover use the same seeded scene construction, initial states, and policy-noise reset procedure. Comparisons match episodes by evaluation seed, suite, task, and initial state.

Computing resources. All experiments were conducted on 8 NVIDIA H200 GPUs with 141 GB of memory each. Each set of trainable heads was trained on a single GPU.

B VISION-LANGUAGE FOCUS MAP DETAILS

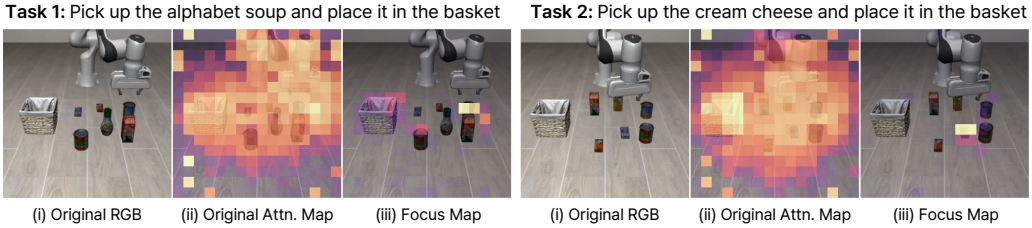


Figure 7: **From dispersed attention to a focus map.** The learned vision-language focus map converts the frozen backbone's dispersed vision-language alignment into a localized relevance signal for the instruction referent. The panels show the original RGB observation, the frozen backbone's implicit alignment, and the learned focus map.

The vision-language focus map processes image patches from each camera in its own coordinate grid. Patches from the available camera views are concatenated to form the full patch representation, but corresponding grid indices in different views are not treated as the same physical location. Region components are therefore extracted separately for each camera before being ranked jointly across views. As a result, different views of the same physical object can produce separate region hypotheses.

The vision-language focus map computes focus scores independently using a sigmoid rather than a softmax over patches. Multiple regions can therefore receive high focus scores at the same time when the visible streaming prefix is compatible with several regions. This property is used by consensus-guided execution to construct multiple region-conditioned action chunks.

The action readiness gate uses backbone features aggregated with the same shared layer weights, with the backbone-specific representations given in Appendix A.

C CONSENSUS-GUIDED EXECUTION DETAILS

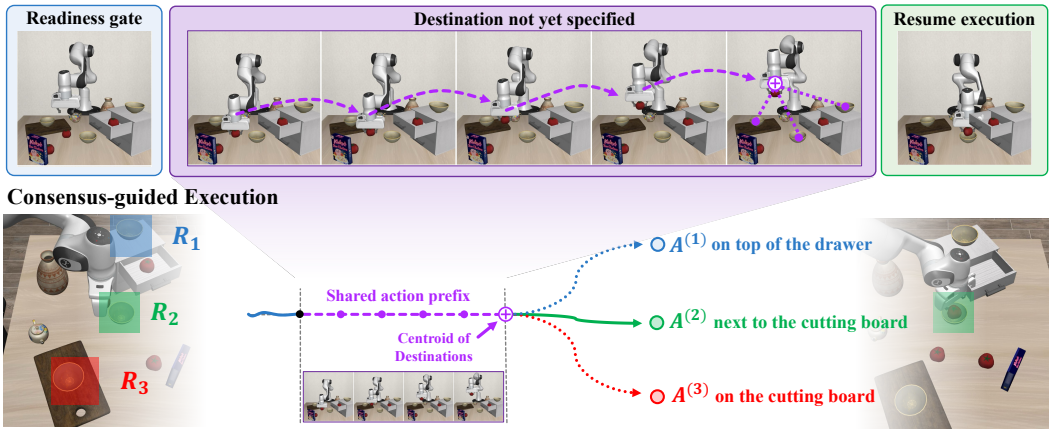


Figure 8: **Consensus-guided execution under destination ambiguity.** After identifying and grasping the target object, Premover executes the shared action prefix of the region-conditioned action chunks, defers execution when they diverge, and resumes default execution once the destination is specified. Dashed trajectories denote alternative predictions that are not executed.

This section specifies region-hypothesis construction, the shared action prefix, and the conditions for resuming default execution used by consensus-guided execution in Section 3.3.

C.1 Region Hypotheses and Convergence

Region hypothesis construction. We extract a ranked list of region hypotheses $\mathcal{R} = (R_1, \ldots, R_J)$ from the cached focus map. For each camera, we retain patches whose focus score is at least 5% of the maximum focus score in that camera's focus map and extract four-connected components. Components from all camera views are ranked by their summed focus scores.

Region-conditioned action chunks. Let $m_i^{(d)}$ indicate whether patch i belongs to region hypothesis R_d, where $d = 1, \ldots, J$. We construct a soft visual weighting

$$w_i^{(d)} = \alpha + (1 - \alpha) m_i^{(d)}. \tag{15}$$

The selected region retains weight one, while visual input outside the region is attenuated by α. We use $\alpha = 0.4$ for LIBERO and $\alpha = 0.8$ for VLA-Arena.

For a fixed observation and streaming prefix, the frozen policy produces the default action chunk $A^{(0)}$ and one region-conditioned action chunk $A^{(d)}$ for each region hypothesis. Predictions within the same planning decision share the same sampling noise or seed. Each region-conditioned action chunk represents the policy's response with that region emphasized.

Cached focus map. Region hypotheses are extracted from the focus map cached at the previous fresh policy query. At the current query, the frozen backbone produces the default action chunk $A^{(0)}$ and features for the current focus map and readiness score. Region-conditioned action chunks use the current observation and streaming prefix with region masks from the cached map; the current focus map is then retained for the next fresh query. While waiting for readiness or for a new word after a shared-action-prefix decision, held control steps reuse the cached state without another policy query. The cache is empty at the beginning of an episode and is cleared at every reset.

Region convergence. Region hypotheses are ranked by their summed focus scores, while convergence is evaluated using their mean focus scores. We apply a softmax with temperature 0.1 to the mean scores and require three conditions: a top-two probability margin of at least 0.5, Jensen-Shannon divergence from the previous region distribution of at most 0.02, and intersection-over-union of at least 0.1 between the current and previous top regions. All conditions must hold at two consecutive updates with distinct word counts. Once reached, convergence remains satisfied for the rest of the episode, and its history is retained after the gripper-close command. A single remaining region hypothesis must also satisfy the temporal stability conditions before default execution resumes.

C.2 Executing a Shared Action Prefix

After the readiness condition is met, the robot executes default action chunks with the default number of actions executed from each predicted action chunk P_{exec} between replanning decisions. In the evaluated implementation, consensus-guided execution is enabled after the first executed gripper-close command, provided that the instruction is incomplete and the region hypotheses have not converged. Any remaining queued actions finish before consensus-guided execution is applied at the next planning decision. At an eligible decision, Premover first checks the conditions for resuming default execution (Appendix C.3); otherwise, it executes only the shared action prefix defined below or waits for additional words.

Shared action prefix. For a set of predicted action chunks $\mathcal{A}$, let M_{common} denote the minimum available chunk length, equal to M when all chunks have the prediction horizon defined in Section 3. For each chunk $A^{(d)} \in \mathcal{A}$, define cumulative translation and gripper command at action step s as

$$x^{(d)}(s) = \sum_{v=1}^{s} a_{v,1:3}^{(d)}, \qquad g^{(d)}(s) = \text{grip}(a_s^{(d)}). \tag{16}$$

The length of the shared action prefix, $S(\mathcal{A})$, is the longest initial interval over which all predicted trajectories remain within the translation and gripper tolerances:

$$S(\mathcal{A}) = \max\Big\{m \in \{0, \dots, M_{\text{common}}\} : \max_{d,d'} \|x^{(d)}(s) - x^{(d')}(s)\|_2 \le \epsilon_x, \\ \max_{d} g^{(d)}(s) - \min_{d} g^{(d)}(s) \le \epsilon_g, \quad \forall s \le m\Big\}. \tag{17}$$

We use $\epsilon_x = 0.15$ in action-delta units and $\epsilon_g = 0.5$. The comparison evaluates cumulative translation and gripper commands along the predicted trajectories.

Execution length. For $J \ge 2$ region hypotheses, when none of the default-execution resumption conditions is satisfied, we compute the execution length as in Eq. (6):

$$S_{\text{cons}} = S\left(\{A^{(1)}, \dots, A^{(J)}\}\right), \qquad m_{\text{exec}}^{\text{cons}} = \min\left(S_{\text{cons}}, C, P_{\text{exec}}\right). \tag{18}$$

We use $C = 3$ and take the first $m_{\text{exec}}^{\text{cons}}$ actions from $A^{(1)}$, the chunk conditioned on the highest-ranked region hypothesis R_1. Each streaming prefix permits at most one execution of a shared action prefix. The robot then holds its pose and preserves the last gripper command until another word arrives, skipping further policy and region-conditioned forwards for the unchanged prefix. If $S_{\text{cons}} = 0$, the robot remains still and waits for the next word before checking again. With fewer than two region hypotheses, no shared action prefix is executed at that decision.

C.3 Resuming default Execution

Default execution resumes when region convergence, action agreement, or instruction completion is reached.

Action agreement. The action-agreement test is first performed at the first planning decision after the first executed gripper-close command for which the instruction is incomplete and the region hypotheses have not converged. It requires at least two valid region-conditioned action chunks and compares them with the default action chunk from the same decision:

$$S\Big(\{A^{(0)}, A^{(1)}, \dots, A^{(J)}\}\Big) > T_{\text{resume}}. \tag{19}$$

The comparison uses the full common prediction horizon without the limit C or the execution horizon P_{exec} and reuses the available chunks without an additional policy forward. If the test fails or fewer than two valid region-conditioned action chunks are available, the robot continues consensus-guided execution. No test is repeated during the unchanged-prefix wait following a shared-action-prefix decision.

We use $T_{\text{resume}} = 6$ for $\pi_{0.5}$ and $T_{\text{resume}} = 10$ for GR00T-N1.7. The prediction horizon is determined by the loaded backbone, and the test uses the full horizon common to the compared chunks.

D Instruction-Delivery Rates and Streaming Protocols

Typing rate. We derive the typing-rate condition from the public keystroke dataset of (Dhakal et al., 2018). WPM is computed using the standard five-character definition of a word. We first average the 15 sentence-level measurements available for each participant and then take the median across participants. The resulting participant-level median is 49.83 WPM; we therefore use the rounded value of 50 WPM as the representative typing rate.

Speaking rate. We derive the speaking-rate condition from Fair-Speech (Veliche et al., 2024), which contains approximately 26.5K voice-command utterances from 593 speakers. Because the original audio durations include leading and trailing silence, we apply voice activity detection and compute WPM over active-speech spans. Of $26{,}470$ utterances, $26{,}466$ yield valid active-speech spans. The active-speech estimate has a median of 149.7 WPM; we use the rounded value of 150 WPM as the representative speaking rate.

Real-time instruction delivery. In simulation, instruction words arrive according to elapsed host time. The scheduling clock t_{sched} starts at the first control tick after the policy-noise and environment resets, before the settling steps, and advances using the monotonic timer `time.perf_counter`. A word with recorded arrival time s becomes visible at the first prefix check for which $t_{\text{sched}} \geq s$. Policy inference, prediction of region-conditioned action chunks, holding, settling, and action execution all consume elapsed time on this clock. When diagnostic state audits are enabled, their measured duration is excluded from the input-arrival clock. Words that arrive during a policy forward or a queued action sequence are incorporated at a subsequent policy query.

Calibration, idle pacing, and warm-up. The calibration procedure measures average host time per rollout step on validation initial states 10 and 11, excluding the first episode for compilation and cache warm-up. All methods in a comparison reuse the same calibration record to set the idle control period Δt and rollout step budget. A holding or settling tick lasts at least Δt, with an explicit wait when simulator computation finishes earlier. Active action steps take their measured computation time without this additional pacing. The calibration period therefore controls idle pacing and the step budget; instruction arrival itself uses elapsed host time rather than $n\Delta t$. Before timed evaluation, the common warm-up procedure uses validation initial state 10 to query the incoming prefixes and region-conditioned action chunks. Warm-up and model loading are outside the reported timing interval.

Incremental-text protocol. For uniform text streams at a rate v WPM, the number of visible instruction words at control step t is

$$k_t = \min\left(W,\, 1 + \left\lfloor \frac{t_{\text{sched}}\, v}{60} \right\rfloor\right). \tag{20}$$

Here, W is the total number of instruction words. At the typing rate of 50 WPM, one word arrives every 1.2 s, with the first word visible at scheduling time zero. The scheduled interval from the first word to instruction completion is $(W-1)60/v$ seconds; each arrival becomes visible at the next prefix check. The tokenizer processes the currently visible words at each policy query. The instruction-delivery-rate ablation in Section 4.5.1 also uses incrementally revealed text, with $v \in \{25, 50, 75, 100, 125, 150, 175, 200\}$ WPM. Its 150-WPM condition uses this uniform text schedule, whereas the speaking-rate results use the ASR replay protocol below.

Speech input and replay. We use XTTS-v2 for speech synthesis and faster-whisper large-v3 for transcription. At the speaking rate of 150 WPM, simulation replays recorded ASR word-arrival timestamps against the elapsed host-time clock t_{sched}. The recorded schedule preserves arrival delays and bursty emissions rather than assuming a uniform 0.4-s interval between words. Speech synthesis and ASR are performed before replay, and the policy receives the prefix available at the current scheduling time. The real-robot results in Section 4.2 use speech and online transcription at the speaking rate; their wall-clock measurements include audio delivery and ASR latency.

Wall-clock measurement. For the real-time simulation evaluation, Wall is the recorded rollout time `rollout_wall_s`. The raw episode timer `wall_s` uses `time.perf_counter`, starts before the policy-noise and environment resets, and stops after the episode loop terminates. The reported metric subtracts the measured reset and settling-computation costs:

$$\text{Wall} = t_{\text{episode}} - t_{\text{reset}} - t_{\text{noise reset}} - t_{\text{settle compute}}. \tag{21}$$

It includes policy inference, additional region-conditioned forwards, readiness and focus computation, execution decisions, and simulator stepping during holding and action execution. Idle-pacing waits, including those during settling, remain included. Model loading, task-scene construction, and separately executed warm-up episodes are outside the raw episode timer.

Wall is averaged over all evaluated episodes, including unsuccessful episodes. It is measured directly, without converting the step count using Δt or adding instruction-delivery time afterward. Relative Wall divides a method's mean by the Full-prompt mean under the same delivery condition, timing definition, and task coverage. Projection-head profiling is reported separately in Appendix H.

Input-waiting and action-time breakdown. Figure 1 reports measured input waiting and robot action times for Full-prompt with $\pi_{0.5}$ on LIBERO at 150 WPM, using 4200 episodes across three

seeds. Input waiting accounts for 65.9%, 57.5%, 50.0%, and 45.9% of the wall-clock time on Spatial, Object, Goal, and Long-10, respectively. The overall share is 54.6%.

E Full Simulation Results

This section reports the complete simulation results for the remaining backbone-benchmark combinations across both instruction-delivery rates.

Table 6: **Main results on VLA-Arena with $\pi_{0.5}$.** Success rate (SR; %) and wall-clock time (Wall; s) across the four VLA-Arena suites at both instruction-delivery rates. Entries are means over three evaluation seeds, with seed-level standard deviations reported for both SR and Wall.

	Safety		Distractor		Extrapolation		Long Horizon		Mean	
Method	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓
Typing rate										
Full-prompt	57.2±1.0	18.45±0.06	78.7±3.9	16.13±0.15	54.5±1.3	24.40±0.10	82.9±2.8	17.22±0.17	64.4±0.2	19.35±0.01
Right-away	59.8±1.1	10.56±0.23	76.3±2.3	7.26±0.36	17.1±0.6	24.53±0.06	79.4±2.1	8.54±0.43	55.2±0.3	13.16±0.05
Noun-triggered	60.3±1.5	12.33±0.21	77.5±1.7	9.14±0.29	35.7±0.3	22.00±0.11	84.8±1.9	10.29±0.24	61.1±0.7	13.88±0.08
Premover	61.2±1.4	13.40±0.22	80.3±0.3	10.61±0.03	41.6±1.3	21.41±0.31	80.3±4.0	12.57±0.54	62.7±0.8	14.80±0.06
Speaking rate										
Full-prompt	57.1±1.6	9.71±0.08	77.8±3.8	8.11±0.18	55.2±1.7	13.46±0.16	82.1±2.3	9.66±0.14	64.3±0.8	10.37±0.05
Right-away	61.4±1.9	7.57±0.21	78.8±1.7	6.19±0.11	32.1±1.4	14.04±0.18	83.3±0.2	7.38±0.07	60.7±0.7	8.93±0.06
Noun-triggered	60.2±0.7	8.00±0.06	78.0±2.1	6.53±0.16	44.7±1.4	13.08±0.14	85.8±2.2	7.67±0.27	63.5±0.7	8.97±0.04
Premover	58.0±1.6	8.75±0.09	77.0±1.9	7.81±0.11	49.9±2.1	12.99±0.18	85.6±1.4	8.56±0.07	63.7±1.3	9.63±0.08

Table 7: **Main results on LIBERO with GR00T-N1.7.** Success rate (SR; %) and wall-clock time (Wall; s) across the four LIBERO suites at both instruction-delivery rates. Entries are means over three evaluation seeds, with seed-level standard deviations reported for both SR and Wall.

	Spatial		Object		Goal		Long-10		Mean	
Method	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓
Typing rate										
Full-prompt	95.0±1.2	20.63±0.06	98.5±0.6	14.87±0.02	95.7±0.8	10.37±0.02	87.6±1.7	21.49±0.09	94.2±0.4	16.84±0.02
Right-away	55.5±1.6	13.11±0.25	50.3±1.2	11.35±0.17	12.9±2.0	14.06±0.14	60.0±2.5	16.22±0.29	44.7±0.9	13.68±0.13
Noun-triggered	55.4±0.9	15.29±0.16	97.4±1.0	8.02±0.06	52.2±0.6	10.83±0.06	71.0±0.7	16.25±0.32	69.0±0.2	12.60±0.06
Premover	95.1±1.3	17.40±0.14	98.2±0.4	12.33±0.02	96.8±0.3	10.35±0.02	86.6±1.3	19.85±0.15	94.2±0.4	14.98±0.03
Speaking rate										
Full-prompt	94.2±0.9	9.21±0.06	97.9±0.3	7.48±0.02	95.5±0.7	5.71±0.06	88.2±1.4	11.99±0.07	94.0±0.3	8.60±0.03
Right-away	56.4±0.9	7.97±0.05	49.6±0.9	7.84±0.08	59.0±1.2	6.97±0.06	65.7±1.2	11.54±0.03	57.7±0.9	8.58±0.05
Noun-triggered	58.4±0.9	8.65±0.04	98.6±0.3	5.84±0.03	76.9±0.9	6.00±0.06	75.0±2.9	11.31±0.23	77.2±0.8	7.95±0.06
Premover	94.3±1.3	8.33±0.04	97.7±1.1	6.60±0.07	95.6±0.4	5.75±0.03	87.4±1.0	11.61±0.12	93.8±0.5	8.07±0.05

Table 8: **Main results on VLA-Arena with GR00T-N1.7.** Success rate (SR; %) and wall-clock time (Wall; s) across the four VLA-Arena suites at both instruction-delivery rates. Entries are means over three evaluation seeds, with seed-level standard deviations reported for both SR and Wall.

	Safety		Distractor		Extrapolation		Long Horizon		Mean	
Method	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓	SR ↑	Wall ↓
Typing rate										
Full-prompt	41.4±1.1	15.92±0.01	66.6±2.3	14.30±0.07	42.9±1.8	21.75±0.08	60.7±4.4	16.13±0.24	49.2±0.6	17.14±0.02
Right-away	39.4±1.5	9.89±0.14	66.0±3.0	6.87±0.51	11.3±1.0	21.80±0.21	61.9±1.6	7.64±0.26	40.6±1.3	11.99±0.20
Noun-triggered	42.7±0.3	11.31±0.14	69.0±1.0	8.36±0.12	31.8±2.3	19.44±0.30	60.4±2.6	10.31±0.26	47.3±0.3	12.68±0.02
Premover	41.2±0.4	13.14±0.10	64.1±2.9	11.30±0.16	33.1±1.2	19.99±0.14	61.3±0.9	11.55±0.09	46.3±1.0	14.28±0.03
Speaking rate										
Full-prompt	41.0±1.2	7.10±0.05	65.5±2.3	6.22±0.03	41.8±0.9	10.86±0.11	60.8±1.7	8.47±0.12	48.6±1.1	8.12±0.05
Right-away	38.0±0.5	5.90±0.00	62.9±1.2	4.80±0.04	15.1±0.7	11.51±0.08	59.6±2.1	6.78±0.07	40.0±0.3	7.27±0.01
Noun-triggered	38.3±1.0	6.03±0.12	66.9±1.6	4.92±0.06	30.5±2.0	10.61±0.13	57.8±1.6	7.15±0.20	44.4±0.8	7.18±0.05
Premover	44.2±2.0	6.53±0.05	65.7±1.2	5.81±0.04	36.9±0.5	10.66±0.05	58.1±1.2	7.74±0.04	48.3±0.7	7.64±0.03

F Real-Robot Implementation Details

The real-robot results in Table 1 use the frozen $\pi_{0.5}$ backbone on an SO-101 robot with speech and online transcription at the speaking rate (150 WPM). Figure 9 illustrates the task setups.

Task definitions and scoring. The five single-object tasks are Block in Drawer, Block in Cup, Cat in Cup, Cup in Cup, and Push Cup. Cat in Cup requires placing a gray cat-shaped object into a cup. All five use binary success. The three disambiguation tasks share the block-to-drawer goal and differ in the referring expression that identifies the target block: in front of the orange cat, on the cup, or next to the gray cat. They report partial success scores with equal subgoal weighting.

Single-object prompts. The five tasks use the following instructions:

- **Block→Cup:** "Pick up the block and place it in the cup on the right."
- **Drawer:** "Pick up the block and put it in the drawer then close it."
- **Cat→Cup:** "Pick up the cat doll and place it in the cup on the right."
- **Cup→Cup:** "Pick up one cup and place it inside the other cup."
- **Push Cup:** "Move the cup leftward by pushing it."

Disambiguation prompts. The three instructions specify different spatial relations to identify the target block:

- **On Cup:** "Pick up the block on the cup, place it in the drawer, and then close the drawer."
- **Gray Cat:** "Pick up the block next to the gray cat, place it in the drawer, and then close the drawer."
- **Orange Cat:** "Pick up the block in front of the orange cat, place it in the drawer, and then close the drawer."

Evaluation protocol. For the single-object results, Wall is averaged over 20 measurements, including unsuccessful episodes. An episode terminates when the final goal state is reached or after a maximum of 900 policy steps. Real-robot Wall includes instruction delivery, online transcription, policy computation, and robot execution, accounting for their overlap. It follows the end-to-end timing definition in Section 4.1; the simulator-specific reset and settling subtractions in Eq. (21) do not apply to the robot. For the disambiguation tasks, we assign 0.5 points for placing the target block in the drawer and 0.5 points for closing the drawer, giving a maximum score of 1.0 per episode. We report the mean episode score as a percentage.

Streaming implementation. Instructions are synthesized through TTS at the speaking rate and transcribed online. Audio capture uses sounddevice v0.5.5, streaming transcription uses faster-whisper v1.2.1, and host-side input events are handled using pynput v1.8.2. Each method receives the same audio and incremental transcript within a given task. Timing improvements are computed relative to the Full-prompt baseline evaluated under the same speaking-rate and transcription condition.

Data collection and supervision. We collect 50 teleoperated expert demonstrations for each task. Observations consist of synchronized third-person and wrist-mounted camera images together with the robot state. A frozen Segment Anything Model (SAM) (Kirillov et al., 2023) generates offline object-mask pseudo-labels for training the vision-language focus map. The VLA backbone remains frozen. At evaluation, the VLA receives camera observations, robot state, and the streaming prefix.

G Sensitivity to the Focus Concentration Set Size

In the main analysis, we measure focus concentration using the gap between the mean focus score of the top-K patches and that of the remaining patches, with $K = 16$. To test whether the observed behavior depends on this choice, we repeat the analysis with $K \in \{8, 16, 32\}$.

As shown in Fig. 10, the same trend holds across all three values of K. Around the annotated readiness cutoff, the top-K patches receive substantially higher focus scores, while the scores of the remaining patches stay comparatively low on both LIBERO and VLA-Arena. Increasing K reduces

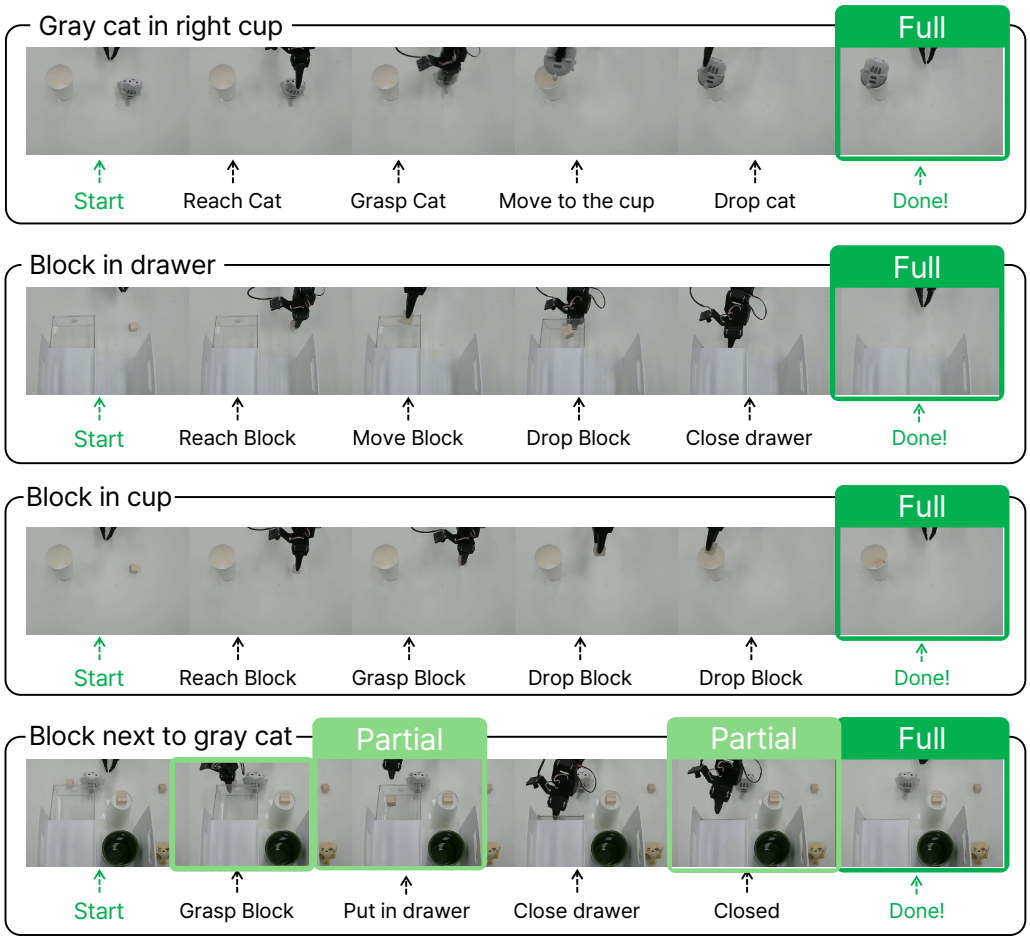


Figure 9: **Real-robot task setups on the SO-101.** The evaluation includes five single-object tasks and three disambiguation tasks with different spatial referring expressions.

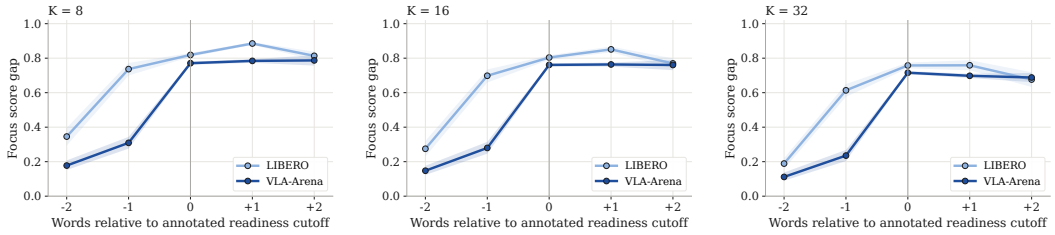


Figure 10: **Sensitivity of focus concentration to K.** Gap between the mean focus scores of the top-K patches and the remaining patches for $K \in \{8, 16, 32\}$, averaged over $\pi_{0.5}$ and GR00T-N1.7. On both LIBERO and VLA-Arena, the gap increases sharply around the annotated readiness cutoff for all three values of K.

the absolute top-K mean, as more lower-scoring patches are included, but does not change the overall behavior. These results indicate that the concentration observed in the main analysis is not specific to the choice of $K = 16$.

Joint-displacement metric. For the real-robot analysis in Fig. 5(c), we measure physical motion relative to the robot's starting configuration. At each time t, we define the maximum joint

displacement as

$$d(t) = \max_j |q_j(t) - q_j(0)|,$$

where $q_j(t)$ denotes the position of joint j at time t. The reported curve averages this quantity across all real-world tasks. This metric indicates when measurable physical motion begins relative to the start of the instruction.

H Projection-Head and Consensus-Guided Execution Overhead

Projection-head measurement. The image and language projection heads contain 2.36M parameters in total, which is small relative to the frozen $\pi_{0.5}$ backbone. In a separate profiling run, the projection heads added 0.232 ms per inference step on LIBERO over an average backbone forward time of 65.84 ms, corresponding to 0.35% of the backbone forward cost. On VLA-Arena, they added 0.239 ms over a 65.98 ms backbone forward, corresponding to 0.36%.

In addition to the projection heads, Premover aggregates backbone features using the shared layer weights, predicts readiness, extracts region hypotheses, and computes region-conditioned action chunks. These costs are included in Wall under the timing definition in Appendix D; the projection-head percentages above do not measure the total overhead of Premover. Each action-agreement test, including repeated tests, reuses chunks from the same planning decision and adds no policy forward. After execution of a shared action prefix, further policy and region-conditioned forwards are skipped until a new word arrives.

I Qualitative Vision-Language Focus Maps

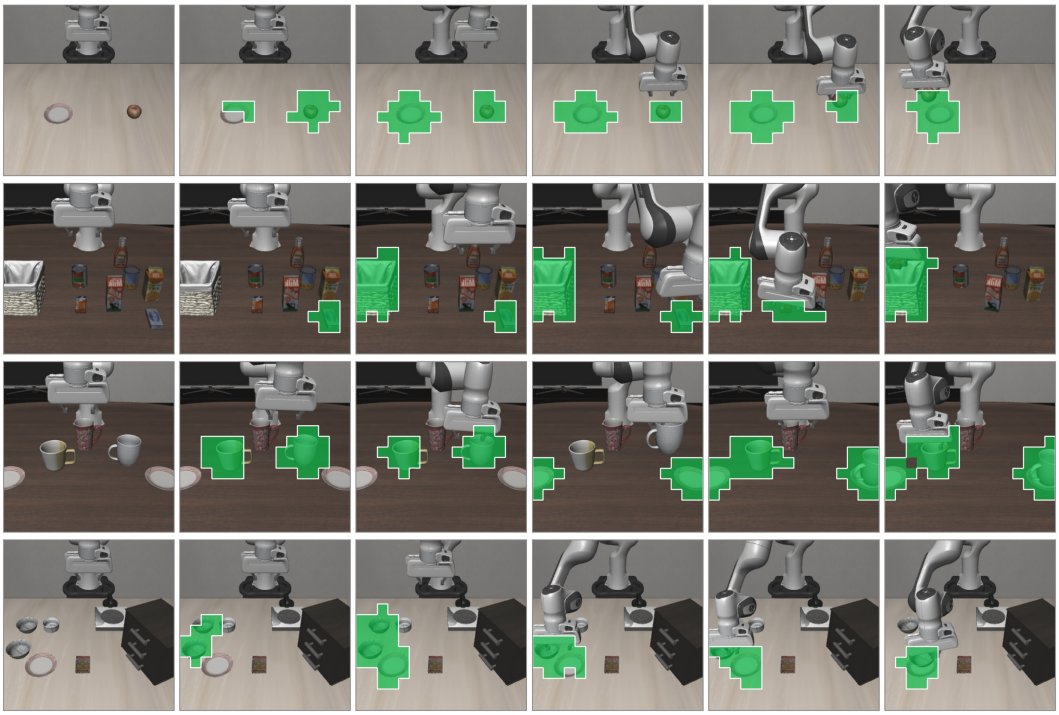


Figure 11: **Qualitative vision-language focus maps in simulation.** Each row shows successive RGB observations with the predicted focus map overlaid. For visualization, only patches with focus score $p_i \geq 0.8$ are displayed; this display threshold is separate from the relative threshold used to construct region hypotheses in Appendix C.1.

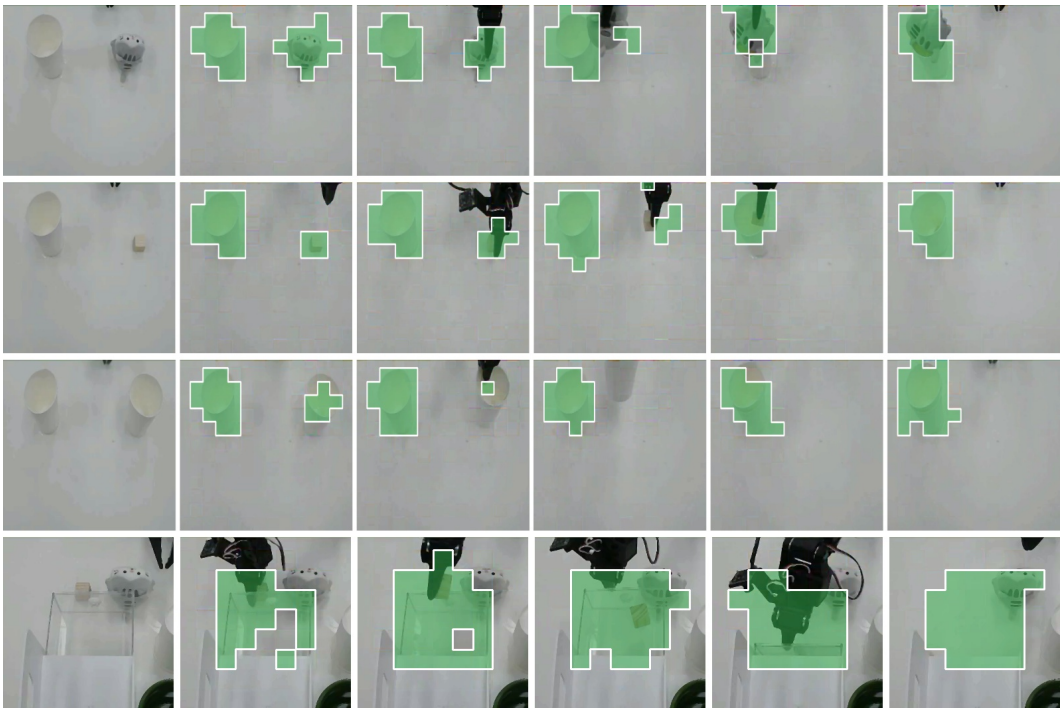


Figure 12: **Qualitative vision-language focus maps on the real robot.** Each row shows successive RGB observations with the predicted focus map overlaid. For visualization, only patches with focus score $p_i \geq 0.8$ are displayed; this display threshold is separate from the relative threshold used to construct region hypotheses in Appendix C.1.

Figures 11 and 12 show focus maps after the streaming prefix has identified the target referent. Across both simulation and real-robot environments, the learned focus map often extends beyond the literal instance-mask boundary to nearby regions such as handles, lids, or contact surfaces.

J Qualitative Streaming Rollout Examples

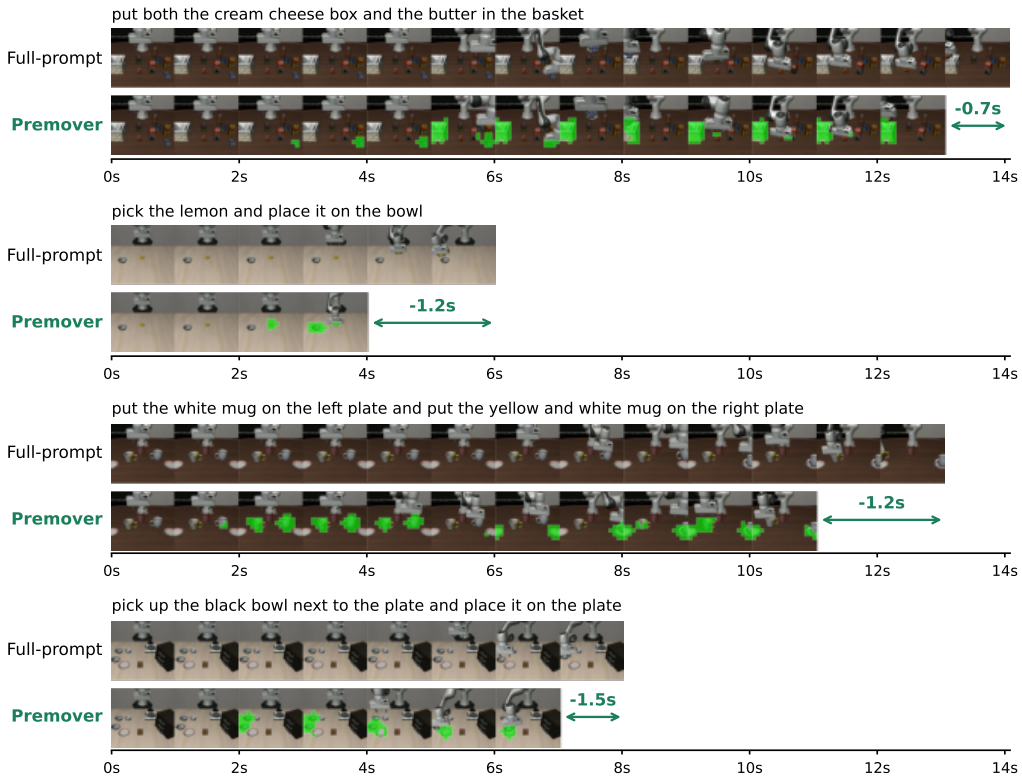


Figure 13: **Qualitative streaming rollout in simulation.** Full-prompt and Premover rollouts are shown on a common time axis aligned to the beginning of instruction delivery. Full-prompt waits for the complete instruction before execution, whereas Premover can begin acting once the current streaming prefix contains sufficient information.

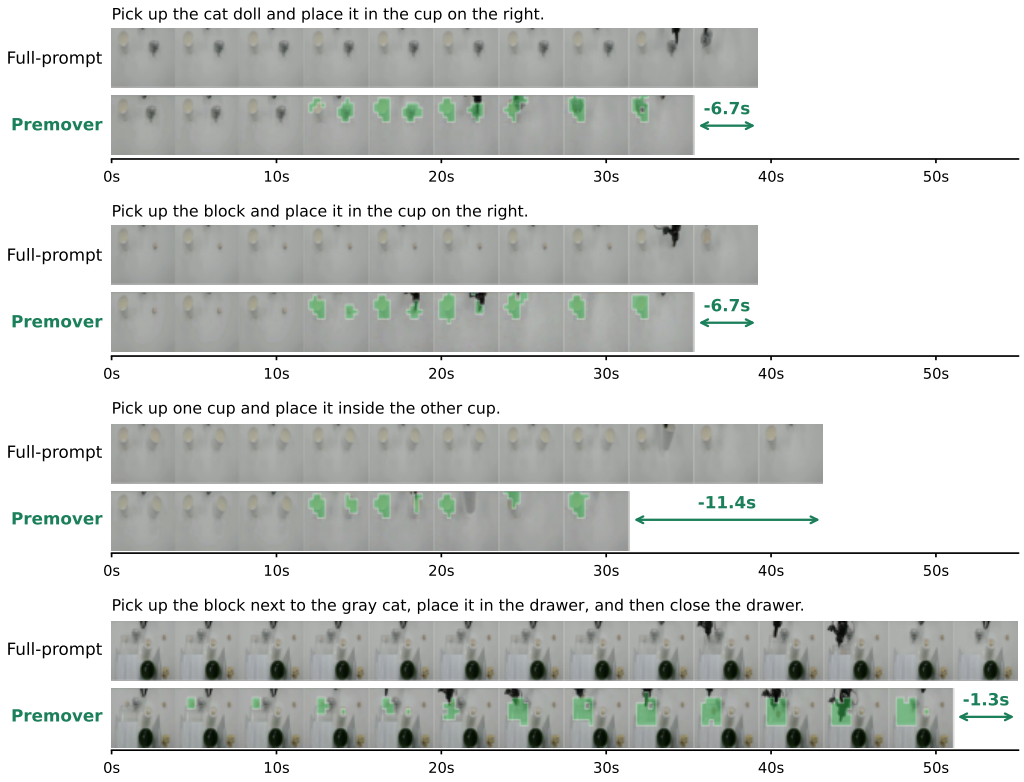


Figure 14: **Qualitative streaming rollout on the real robot.** Full-prompt and Premover rollouts are shown on a common time axis aligned to the beginning of instruction delivery. Premover begins execution from the streaming instruction while Full-prompt waits for the complete instruction.

Figures 13 and 14 illustrate how Premover overlaps instruction delivery with robot execution in simulation and on the real robot. Because Premover can begin execution before the complete instruction arrives, its trajectory can advance while Full-prompt is still waiting. The two methods therefore reach different robot states at the same elapsed time, and subsequent observations need not match frame by frame.