TWILL: A PROACTIVE AGENT THAT DECIDES FROM INTERNAL PROPOSALS AND HUMAN REQUESTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A household robot should recognize from its own observations what a situation requires. Instruction-driven manipulation systems and proactive assistants that anticipate human needs share a blind spot: the robot seldom forms its own judgment of the scene, let alone carries it through to execution. A robot with its own judgment must also arbitrate between that judgment and what the person asks. We build TWILL, a proactive robotic agent that closes the loop from perception to action. Its proposer forms a proposal from the scene alone. Its decision module, SHUTTLE, arbitrates between the proposal and the human request inside a pretrained vision-language model. SHUTTLE reads the scene and each intent through separate attention paths, so that the decision distinguishes the robot's intent from the person's. An off-the-shelf executor then carries out the decision. To evaluate the full loop, we introduce TWINE, which scores understanding and decisions on 360 cases in 180 controlled pairs, each differing in one consequential factor, and scores closed-loop success in simulated kitchens. The proposer reaches 89.88% accuracy in deciding whether and what to propose, 48.61 points above the untuned backbone. SHUTTLE raises the rate of fully correct decisions by 3.70 points over the same fine-tuned backbone without separate reads. In closed loop, TWILL reaches an 81.48% success rate on request-free episodes, against 50.00% for an untuned vision-language model commanding the same executor. TWILL thereby allows a household robot to act on proposals it forms from the scene and to arbitrate between these proposals and human requests according to context.

1 INTRODUCTION

Many household tasks are never requested explicitly, and many of them are time-sensitive: a faucet left running wastes water, and a burner left on wastes energy and poses a safety hazard. Requiring a person to assign every task adds to their cognitive load and lowers the efficiency of both the person and the robot. A robot that decides for itself what needs doing must also account for the requests of the people around it. For example, if someone says "don't turn off the water yet, I'm still using the sink," a robot that is about to turn off the running faucet should wait.

Existing systems rarely address this situation, and few of them let the robot form a proposal from its own observation of the scene. Instruction-driven systems, including task-planning and code-as-policy systems (Ahn et al., 2022; Liang et al., 2023; Singh et al., 2023) and embodied manipulation policies such as vision-language-action (VLA) models (Kim et al., 2024; Black et al., 2025) and world action models (Ye et al., 2026; Li et al., 2026), execute the goal a person states, so tasks that no one requests are not performed; proactive assistance acts without an instruction, but infers the goal from a model of the person, such as predicted intentions, anticipated needs, or learned preferences (Lu et al., 2025; Grannen et al., 2025). Even when both intents are present, existing models have no mechanism for arbitrating between them. When a robot proposal and a person's request are both present, a typical agent harness places both in one context, and most vision-language models read them as one undifferentiated stream of text: nothing inside the model separates the robot's intent from the person's, so the decision cannot fully weigh both intents in the current situation. Existing evaluations mostly measure how well a robot executes instructions; they cannot tell which intent a decision followed or whether the robot attended to the detail that determines the outcome.

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.

We present TWILL (TWo Intents in a Language-closed Loop), a proactive robotic agent that weighs its own proposal against a person's request, and formulate its task as dual-intent decision making. A correct decision requires more than understanding the current situation: the robot must judge whether to continue with its own proposal, follow the request instead, wait, or merge the two into a new plan.

TWILL has three components: a proposer, a decision module, and an executor. An image-only proposer forms a proposal without seeing any request. The decision module SHUTTLE shuttles the scene, the proposal, and the request into one natural-language decision. A frozen VLA executor carries out the decision, and the system runs in a closed loop, so the robot can revise its decision as soon as the scene or the request changes.

To keep the two intents distinguishable, SHUTTLE adds separate attention paths for the scene, the proposal, and the request at the positions where the decision is generated inside the pretrained model. Our main baseline, the Direct method, fine-tunes the same backbone on the same inputs with the same low-rank adaptation (LoRA), but without these paths.

For evaluation, we build TWINE (TWo-INtent Evaluation), a benchmark that evaluates the whole loop, from perception through decision to execution. For the proposer, it checks whether the robot proposes when the scene requires an action and holds when it does not. For the decision module, it uses controlled pairs: the two cases of a pair are identical except for one factor that determines the correct decision, such as whether a request's condition has been met, so a model that disregards this factor cannot decide both cases correctly; separate probes also assess the model's understanding of the scene, the proposal, and the request. For the full system, it checks whether executed decisions succeed in closed loop. On TWINE, the proposer achieves higher joint accuracy than its untuned backbone. SHUTTLE produces fully correct decisions more often than Direct, with the largest gains where the two intents must be weighed against each other, and identifies the source of an instruction more accurately. In closed loop, TWILL completes request-free episodes more often than an action policy trained without language and than TWILL's untuned backbone directing the same executor. When a person makes a request, TWILL with SHUTTLE handles a request that names an absent object more often than TWILL with Direct or the untuned backbone, and it stops on every cancellation.

Our main contributions are as follows:

1. **TWILL**, a proactive robotic agent that connects an image-only proposer, a decision module, and a frozen VLA executor in one closed loop. TWILL formulates proactive assistance as dual-intent decision making, in which the robot's own proposal and a person's request are two sources of one decision.
2. **SHUTTLE**, a decision module that adds separate attention paths for the proposal and the request to a pretrained vision-language model. The module sits at a single decoder layer, starts from the fine-tuned backbone through a zero-initialized residual, and is trained on complete natural-language decisions.
3. **TWINE**, a benchmark that evaluates the proposer, the decision module, and the full closed loop. Its decision tests use controlled pairs that check whether a decision follows the factor that should decide it, and separate probes measure the model's understanding of the scene, the proposal, and the request.

2 RELATED WORK

Instruction-driven policies and planners. Vision-language-action policies and language planners generalize across scenes, reason before acting, and use execution feedback (Physical Intelligence et al., 2025; Bu et al., 2025; Wu et al., 2026; Huang et al., 2025; Bai et al., 2026; Shi et al., 2025; Hsieh et al., 2025; Yao et al., 2023; Wang et al., 2024), and in all of them the goal arrives with the instruction. TWILL adds the robot's own reading of the scene as a second goal source.

Proactive assistance. Proactive agents anticipate needs from text, speech, video streams, or preferences (Fu et al., 2026; Yan et al., 2026; Zhang et al., 2026; Fang et al., 2026; Wang et al., 2026b), and equilibrium maintenance (Buyukgoz et al., 2022) acts on symbolic world states specified by

hand. TWILL finds problems in camera images on its own and weighs them against a later request in one explicit decision.

Reading multiple sources. Separate encoders for several sources (Libovický et al., 2018; Chen et al., 2020) and zero-initialized residual branches in pretrained networks (Bachlechner et al., 2021; Alayrac et al., 2022; Zhang et al., 2023) are established designs. SHUTTLE brings both into one layer of a pretrained vision-language model.

Evaluating multimodal decisions. Benchmarks probe recognition, constraint following, and physical reasoning (Yu et al., 2024; Wang et al., 2026a; Chow et al., 2025), and contrast sets test the minimal change that alters an answer (Gardner et al., 2020; Kaushik et al., 2020). TWINE applies controlled pairs to embodied dual-intent decisions and scores understanding, decisions, and closed-loop execution.

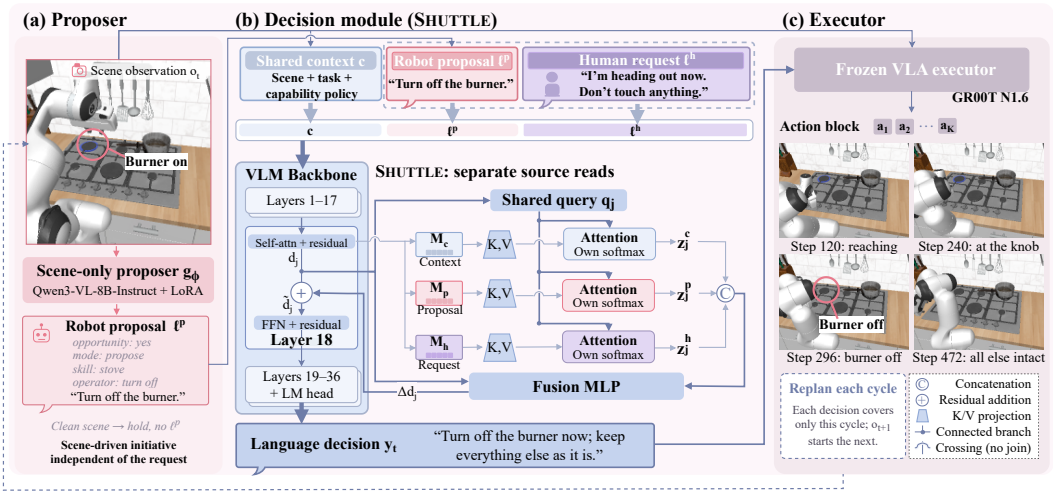

Figure 1: **TWILL.** (a) The proposer forms the robot's own proposal ℓ^{p} from the scene alone. (b) SHUTTLE reads the context c, the robot proposal ℓ^{p}, and the human request ℓ^{h} on separate paths at layer 18 of the vision-language backbone and generates one language decision y_t. (c) A frozen VLA executor carries it out, and the next observation starts a new cycle. The person is leaving and asks the robot not to touch anything; however, a burner left on with nobody home is unsafe, so the decision still turns it off.

3 TWILL: TWO INTENTS IN A LANGUAGE-CLOSED LOOP

TWILL runs in cycles (Figure 1). In each cycle, the proposer forms a proposal from the current scene, SHUTTLE weighs the proposal against a request when a person has made one, and the executor carries out the resulting decision.

Notation. At cycle t the robot observes RGB images o_t and proprioception ρ_t. The context c_t holds the images, a scene description, the current task, and the capability policy that bounds what the robot may do. Two natural-language intents act on this context: the proposal ℓ_t^{p}, which the robot forms from the scene, and the request ℓ_t^{h}, which the person may give. The decision y_t states in natural language what the robot does now. One cycle computes

$$\ell_t^{\mathrm{p}} = g_\phi(o_t), \qquad y_t \sim p_\theta(\cdot \mid c_t, \ell_t^{\mathrm{p}}, \ell_t^{\mathrm{h}}), \qquad a_{t,1:K_t} = e_\psi(y_t, o_t, \rho_t), \tag{1}$$

with the proposer g_ϕ, the decision module p_θ, and the executor e_ψ; $a_{t,1:K_t}$ are the K_t low-level control actions that the executor issues in the current cycle, forming one action block ($K_t = 8$ in our experiments).

Loop. An agent harness runs the components in a closed loop. Each cycle begins with a new observation, which the proposer processes first. Without a request, the harness passes the proposal directly to the executor, or ends the cycle if the proposer holds. The decision module is called only when a request is present, and it reads the proposal and the request together. The executor then carries out the decision (Equation 1). A request that arrives while the executor is acting is handled in the next cycle, and a cancellation stops the robot until the person gives a new goal.

Shared backbone and execution. The proposer and the decision module are two adapters on one Qwen3-VL-8B-Instruct backbone (Bai et al., 2025), so the loop keeps a single backbone in GPU memory and loads the decision adapter only when a request is present. The executor e_ψ is the frozen, off-the-shelf GR00T N1.6 policy (NVIDIA et al., 2025; NVIDIA, 2025), which turns each decision into low-level actions on the object and destination the decision names.

4 Proposing from the Scene and Deciding from Two Intents

4.1 The proposer

The proposer maps one RGB image from o_t to an answer with five fields: an opportunity flag, a propose-or-hold mode, a skill family (water, stove, door, or drawer), an operator (stop flow, turn off, or close), and a proposal sentence ℓ^{p}. It never sees the request, so the robot's judgment is formed independently of the person.

The main difficulty is restraint: an untuned vision-language model proposes on most scenes that need no action, and a running faucet often looks similar to a closed one. The proposer is Qwen3-VL-8B-Instruct with rank-16 LoRA on the query, key, value, and output projections of the language model's attention layers, and with a trainable vision encoder. The opportunity flag in its answer is a single yes or no token. For an image x, let $s_{\mathrm{yes}}(x)$ and $s_{\mathrm{no}}(x)$ be the logits of these two tokens at the flag position, so that $\zeta(x) = s_{\mathrm{yes}}(x) - s_{\mathrm{no}}(x)$ measures how strongly the proposer sees an opportunity. Training minimizes

$$\mathcal{L}_g = \sum_{x \in \{x_{\mathrm{on}}, x_{\mathrm{off}}\}} \big[\mathcal{L}_{\mathrm{ans}}(x) + \mathcal{L}_{\mathrm{opp}}(x)\big] + \lambda \,\mathrm{softplus}\big(\zeta(x_{\mathrm{off}}) - \zeta(x_{\mathrm{on}})\big), \tag{2}$$

where $\mathcal{L}_{\mathrm{ans}}$ is the cross-entropy over the whole answer and $\mathcal{L}_{\mathrm{opp}}$ is a two-way cross-entropy over the two flag logits. The last term, weighted by $\lambda = 0.1$, compares two images of the same scene, one with the problem present (x_{on}) and one with it resolved (x_{off}), and pushes the opportunity score of the first above that of the second. Because the two images share the layout, objects, and lighting, the proposer can reduce this term only by attending to the state that differs, such as whether water flows. Training details are in Appendix A.3.

4.2 Shuttle: arbitrating two intents

Dual-intent arbitration. Which intent takes precedence depends on the situation (Section 1; Figure 2). A valid request that changes the goal takes precedence over the proposal; a request that names a wrong object or asks for something unsafe leaves the proposal unchanged; a conditional request postpones the proposal until the condition holds; and a request that adds a goal or a prerequisite is combined with the proposal. The decision therefore keeps every goal that either source validly contributes. The output y is one natural-language decision that says what the robot does in this cycle, in what order, and after which prerequisites; when the robot should hold, wait, or ask for clarification, it states this together with the condition, and it also states any later steps that the inputs explicitly require. The decision is generated token by token,

$$p_\theta(y \mid c, \ell^{\mathrm{p}}, \ell^{\mathrm{h}}) = \prod_{j=1}^{|y|} p_\theta(y_j \mid c, \ell^{\mathrm{p}}, \ell^{\mathrm{h}}, y_{<j}), \tag{3}$$

where $y_{<j} = (y_1, \ldots, y_{j-1})$.

Source reads. Shuttle inserts a lightweight residual module at one decoder layer of the pretrained vision-language model, so that the decision can address each source separately (Figure 1b).

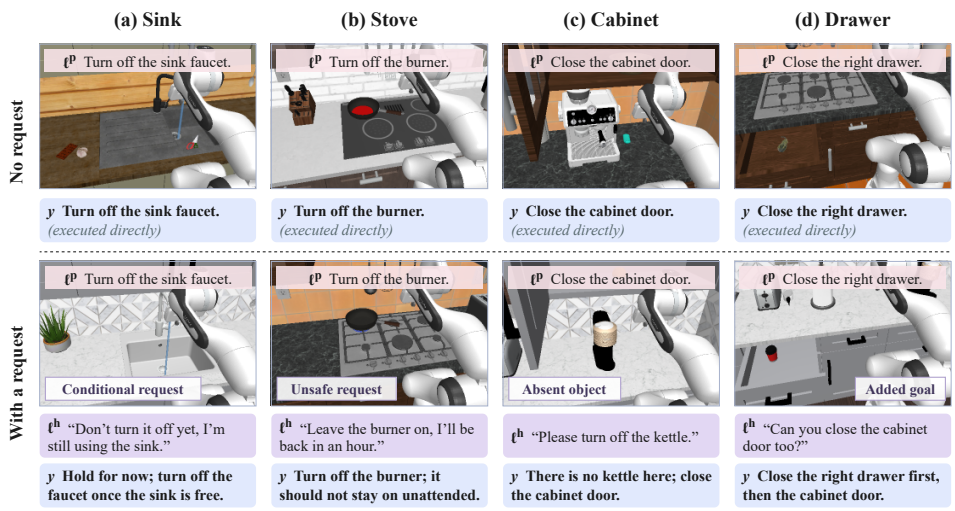

Figure 2: **Scenes and decisions for four kitchen fixtures**. Without a request (top), TWILL executes its proposal ℓ^{p}. With a request ℓ^{h} (bottom), the decision y defers the proposal on a condition, keeps it against an unsafe or absent-object request, and orders an added goal after it.

The chat template places c, ℓ^{p}, and ℓ^{h} one after another, followed by the decision; we record which tokens belong to each source, and every position keeps the model's ordinary causal attention. In this native computation, a decision token attends over the whole prefix through one softmax, in which the long scene context and the short proposal and request share the same attention mass. SHUTTLE gives each source its own softmax, so every decision position also receives a separate read of the scene, the proposal, and the request. We place the module at text layer 18 of the 36-layer decoder, after the self-attention residual and before the feed-forward block (Appendix A); among single insertions at layers 9, 18, and 27, this depth gives the highest complete-decision pass rate (Appendix B.12). Let $d_j \in \mathbb{R}^{4096}$ be the state at decision position j, and let M_{c}, M_{p}, and M_{h} collect the states at the source positions at the same point. A shared query and per-source key and value projections give

$$q_j = \mathrm{LN}(d_j)W_Q, \qquad K_R = \mathcal{N}_R(M_R)W_K^R, \qquad V_R = \mathcal{N}_R(M_R)W_V^R, \tag{4}$$

for $R \in \{\mathrm{c}, \mathrm{p}, \mathrm{h}\}$ and rank $k = 128$, where LN is an affine LayerNorm, $\mathcal{N}_{\mathrm{c}}$ is a parameter-free per-token RMS normalization, and $\mathcal{N}_{\mathrm{p}}$ and $\mathcal{N}_{\mathrm{h}}$ are identity maps that leave the proposal and request states unchanged. Each source is read with its own softmax, and an absent source contributes a zero read:

$$z_j^R = \begin{cases} \mathrm{softmax}\left(q_j K_R^\top / \sqrt{k}\right) V_R, & |M_R| > 0, \\ \mathbf{0} \in \mathbb{R}^k, & |M_R| = 0. \end{cases} \tag{5}$$

Joint update. A fusion MLP combines the native state with the three reads and writes a residual on the decision positions only:

$$u_j = \mathrm{GELU}\left([d_j; z_j^{\mathrm{c}}; z_j^{\mathrm{p}}; z_j^{\mathrm{h}}]W_{\mathrm{in}} + b_{\mathrm{in}}\right), \qquad u_j \in \mathbb{R}^{245}, \tag{6}$$

$$\widetilde{d}_j = d_j + \Delta d_j, \quad \Delta d_j = u_j W_{\mathrm{out}}, \tag{7}$$

$$h_j^{(18)} = \widetilde{d}_j + \mathrm{FFN}_{18}\left(\mathrm{RMSNorm}_{18}(\widetilde{d}_j)\right). \tag{8}$$

The states at the source positions are not modified, and the remaining layers are computed as in the original model. W_{out} is initialized to zero, so at the first training step the module has no effect and SHUTTLE starts from the same model as Direct, the fine-tuned backbone without the module.

Training. The native LoRA adapters (Hu et al., 2022) and SHUTTLE are trained on complete decisions, first the LoRA alone and then both jointly:

$$\mathcal{L}(\theta) = -\frac{1}{|y|}\sum_{j=1}^{|y|} \log p_\theta(y_j \mid c, \ell^{\mathrm{p}}, \ell^{\mathrm{h}}, y_{<j}). \tag{9}$$

Table 1: **Test decision set** (six cases per family).

Module	Capability	Factor changed within a pair (pairs)	Pairs
A	Goals and order	action order (30); keep or cancel the proposal (15); prerequisite (15)	60
B	Triggers	confirmation received or not (15); relevant or irrelevant signal (15)	30
C	Local revision	destination of an added step (15); one object or a set of two (15)	30
D	Mixed validity	permitted or forbidden extra action (15); current or withdrawn request (15)	30
E	Scene constraints	rule flips on one image (15); same rule, new scene state (15)	30
Total	60 scene families, 360 cases		180

The pretrained weights stay fixed, and SHUTTLE adds 5.78M parameters (Appendix A).

5 TWINE: EVALUATING THE PROACTIVE LOOP

TWINE evaluates each component of the proactive loop in native RoboCasa365 kitchens (Nasiriany et al., 2026) in three tiers: proposing, deciding, and executing.

5.1 PROPOSAL ACCURACY

The proposing tier asks whether the proposer proposes when a scene requires an action and holds when it does not. It uses 198 held-out observations of 256×256 RGB images, including pairs that show the same scene with the problem present and resolved. We report joint accuracy, which requires the opportunity flag, mode, skill family, and operator all to be correct; restraint, the accuracy on observations that need nothing; and paired on/off accuracy, which requires both images of a pair to be answered correctly (Appendix B.1).

5.2 CONTROLLED DECISION PAIRS

The deciding tier asks whether a decision module weighs the proposal and the request as each situation requires. A single case cannot show whether a decision depends on the right factor: a model that always follows the request, for example, is correct whenever the request is valid. TWINE therefore tests decisions in controlled pairs.

Controlled pairs. The two cases of a pair are identical except for one factor that changes the correct decision, such as whether the person has finished using the sink (Figure 3). A model that ignores this factor gives both cases the same answer and gets at least one of them wrong. The 360 test cases form 180 pairs in five modules (Table 1): each of 60 scene families contributes one module-A pair and two pairs from modules B–E. In scene-change pairs, the image changes while the proposal remains the one generated for the earlier scene, so the decision must revise that proposal for the current scene. A separate invariance set of 60 cases rewords a request or adds an irrelevant sentence without changing the correct decision (Figure 3; Appendix B.9).

Cases. Each case combines one image with the context, a proposal, and a request. A scene family is a layout–style combination; the 60 test families span 34 layouts, 35 styles, and 76 images, and none of them appears in training. Requests are generated from reviewed templates filled with objects and places from the scene, and every mentioned object is checked against the image before use. Each case carries a reference decision with two to four critical obligations, such as the current action, a goal to keep, an order, or a trigger, and nine understanding probes that ask about the image ($U1$), the meaning of each source ($U2$), and relations across sources ($U3$).

Scores. A decision passes only if four conditions hold: the output is complete (C), every critical obligation is met (V), all necessary content is present (N), and the answer is consistent as a whole (O),

$$\mathrm{CompletePass}(y) = \mathbf{1}[\, C(y) \wedge V(y) \wedge N(y) \wedge O(y) \,]\,. \tag{10}$$

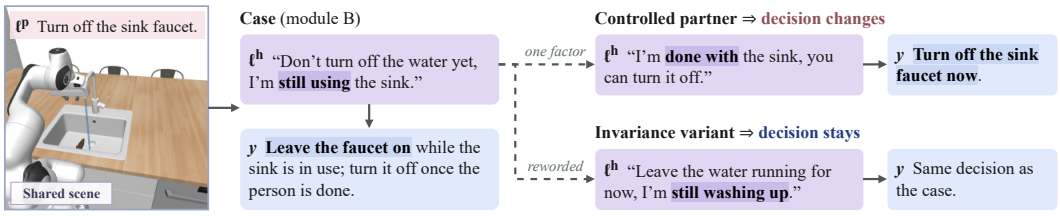

Figure 3: **A controlled pair and an invariance variant** (module B). The partner changes one factor, whether the person is done with the sink, so the correct decision changes, and the pair passes only when both members are right; rewording the request alone leaves the decision unchanged.

Rules check completeness, and Gemini-3.8-Flash (Google DeepMind, 2026) checks the other conditions against verified scene facts. It judges each answer twice, with the proposal and the request listed in either order, and a check passes only if it passes both times. The resulting complete-decision pass rate is the primary endpoint. The understanding score is the mean accuracy over $U1$–$U3$. On 128 decisions, it agrees with an author's blind judgments on 92.97% ($\kappa = 0.849$), comparable to the 90.63% agreement between two blind annotators ($\kappa = 0.745$; Appendix B.5). Methods are compared within case and seed, with 95% confidence intervals (CIs) from 10,000 scene-family bootstrap draws and Holm-adjusted p-values over the five primary contrasts (Appendix B).

5.3 Closed-loop execution

The executing tier runs the full TWILL loop in closed loop on three fixtures: a running faucet, an open cabinet door, and an open drawer. Seven conditions cover behavior without and with a request. Without a request, a problem is present from the start, appears during monitoring, or is absent and calls for restraint; with a request, the person gives a matched instruction, asks about an absent object, or cancels while the robot is acting or idle. An episode succeeds only when all physical obligations hold, including protected states, and success rates weight the three fixtures equally (Appendix B.14).

6 Experiments

We use TWINE to answer four questions about TWILL, in the order in which it runs. **Q1:** Does the proposer propose when the scene calls for help and hold otherwise? **Q2:** Does SHUTTLE decide better than same-backbone baselines? **Q3:** How does the complete loop behave with and without a request? **Q4:** Where does the decision gain come from? Decision tests use 60 held-out scene families; closed-loop tests use held-out kitchen styles.

6.1 Setup

Decision models. The decision models start from the same LoRA checkpoint on Qwen3-VL-8B-Instruct and train in two stages on 6,268 complete-decision demonstrations; for each method and seed, the checkpoint is chosen on 198 development cases from unused scenes (Table 5). Each chosen checkpoint is tested once on TWINE, and results average three seeds (17, 23, and 41).

Baselines. All baselines share the backbone, native LoRA, training data, and update plan. Direct is the same fine-tuned model without the SHUTTLE module (Section 4.2). A parameter-matched adapter without source-specific reads tests added capacity, and hierarchical attention (Libovický et al., 2018) and DMRM-style tri-attention (Chen et al., 2020), two established multi-source designs, test whether an existing scheme suffices. Merged-text reads keep the SHUTTLE module but read the proposal and the request as one text, testing whether separate reads preserve the source of each instruction (Q4).

Closed loop. On the executing tier of TWINE, we compare TWILL with SHUTTLE against TWILL with Direct, which changes only the decision module, and four external systems: a policy trained without language (VA), a mission-prompted VLA (the executor given one fixed kitchen mission), an instruction-driven VLA, and a vision-language model (VLM) commander (Appendix B.14).

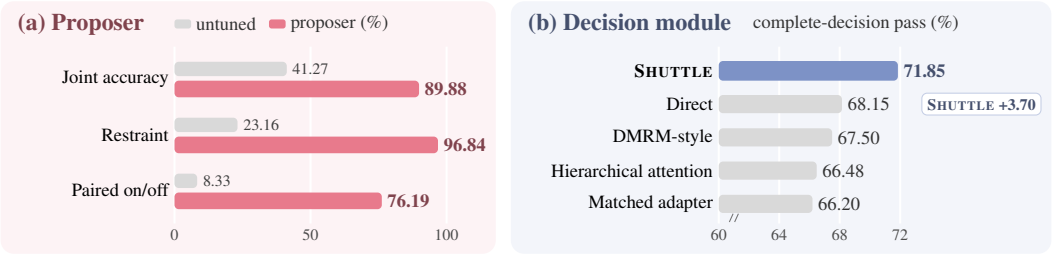

Figure 4: **Proposer and decision results.** (a) Proposer scores (Section 5.1). (b) Complete-decision pass rate; the axis starts at 60.

Table 2: **Closed-loop outcomes** (%, task-family-weighted, decision seeds 17, 23, and 41; 22 episodes per condition and seed; Cancel averages cancellation during action and while idle; see Tables 20 and 21).

	No request		With a request		
System	Success	Clean hold	Matched request	Absent object	Cancel
TWILL with SHUTTLE	81.48	93.33	88.15	42.22	100.00
TWILL with Direct	81.48	93.33	96.30	21.48	100.00
VA (no language)	6.67	20.00	–	–	–
Mission-prompted VLA	0.00	0.00	–	–	–
Instruction-driven VLA	33.33	100.00	88.89	5.56	100.00
VLM commander (Qwen3-VL-8B)	50.00	88.89	60.00	0.00	80.00

6.2 PROPOSING AND HOLDING (Q1)

On 198 observations of 256×256 RGB images (Figure 4a; Appendix B.1), the proposer's joint accuracy is 89.88%, against 41.27% for the untuned backbone with the same prompt. The gain comes from restraint: on scenes that need no action, the proposer holds in 96.84% of cases against 23.16%, false proposals fall from 88 to 3, and both images of an on/off pair are answered correctly in 64 of 84 pairs against 7.

6.3 DECISION QUALITY (Q2)

The decisions in TWINE are difficult: zero-shot Gemini-3.8-Flash passes 46.11% of the cases completely, and the untuned backbone 19.44% (Appendix B.6). SHUTTLE has the highest complete-decision pass rate (the share of fully correct decisions; Section 5.2) of the five methods in Figure 4b: 71.85%, against 68.15% for Direct, a paired gain of 3.70 points (95% CI [1.76, 5.65], Holm-adjusted $p = 0.0008$). The gain is largest where the two intents must be weighed against each other. On module A (goals and order), SHUTTLE passes 74.17% of cases against 68.06% for Direct, and across the full test set it more often retains every required goal and places actions in the required order (+3.14 and +3.81 points, Holm-adjusted $p = 0.010$ and 0.0052; Appendix B.8). SHUTTLE also answers both cases of a controlled pair correctly more often (53.33% of pairs against 47.96%). Its lead over the parameter-matched adapter (5.65 points, $p < 0.001$) shows that the gain comes from the source reads. The module decides as fast as Direct.

6.4 CLOSED-LOOP BEHAVIOR (Q3)

Without a request (Table 2), TWILL meets all physical obligations in 81.48% of episodes, against 50.00% for the VLM commander and 6.67% for VA; the decision module is not called in these episodes (Section 3), so both TWILL variants share this result. With a request (Figure 5 shows one episode), the decision module has the largest effect when the request conflicts with the scene: when the person asks about an absent object, TWILL with SHUTTLE handles it correctly in 42.22% of cases, about twice the 21.48% with Direct, against 5.56% for the instruction-driven VLA and 0.00% for the VLM commander. TWILL stops on every cancellation.

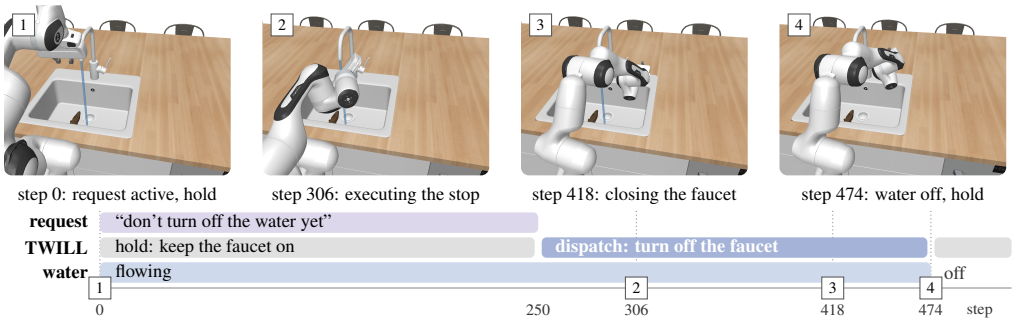

Figure 5: **A conditional-request episode.** TWILL holds its proposal while the request is active, dispatches the stop once it lifts, and then holds again.

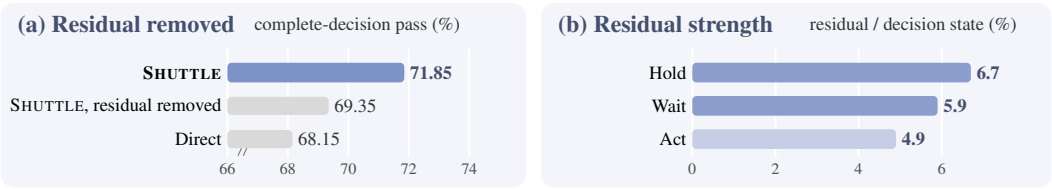

Figure 6: **Where the gain comes from.** (a) Complete-decision pass over three seeds with the residual zeroed at test time (axis starts at 66). (b) Residual size relative to the decision state.

6.5 WHERE THE GAIN COMES FROM (Q4)

The module's residual carries most of the gain. Setting the residual to zero at test time, with every other weight unchanged, lowers SHUTTLE from 71.85% to 69.35% complete-decision pass over the three seeds, which removes about two thirds of its 3.70-point gain over Direct (68.15%; Figure 6a; Appendix B.13). The residual is also largest where the request should change what the robot does: its size relative to the decision state is 6.7% on hold decisions and 5.9% on wait decisions, against 4.9% on act decisions (Figure 6b). On source-swapped pairs, SHUTTLE names the source of an instruction in 87.50% of answers, against 84.72% for merged-text reads ($p = 0.014$) and 84.03% for Direct; it answers all four source questions of a pair correctly in 58.89% of pairs, against 45.00% and 41.11% (Appendix B.10).

7 CONCLUSION

Proactive robots rarely form goals from their own view of the scene, and when they do, the robot's intent and the person's request usually reach the model as one undifferentiated input. TWILL treats proactive assistance as dual-intent decision making: the robot forms a proposal from the scene, weighs it against the person's request, and acts on the result. We show that keeping the two intents apart inside the vision-language model, as SHUTTLE does, yields better decisions than the same model without this separation, most of all where the two intents must be weighed against each other. In closed loop, the full system resolves problems it detects on its own and handles requests that conflict with the scene more reliably than an instruction-driven VLA or an untuned VLM directing the same executor.

Limitations and future work. We evaluate in simulated RoboCasa365 kitchens; the proposer covers six household problems that are resolved by closing or turning off a fixture, and the closed-loop tests cover three of these fixtures. Future work will extend proposals and closed-loop tests to open-ended household tasks, deploy the loop on physical robots, and model how precedence between the intents shifts with context.

AI USE STATEMENT

Generative AI tools were used to polish the writing, to assist with implementation and experiment management, and to generate synthetic text for the training and benchmark data. Synthetic benchmark content was reviewed before use (Section 5.2), and LLM-based scoring is part of the evaluation protocol, validated against blind human annotation (Section 5.2). The authors reviewed and tested the code behind all reported results and take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

Training data and optimization details are in Appendix A. The benchmark construction, annotation rules, aggregation, and uncertainty protocol are in Appendix B; the closed-loop protocol is in Appendix B.14. Code and result records will be released in an anonymized repository.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, et al. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2022.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 23716–23736, 2022.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian McAuley. ReZero is all you need: fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pp. 1352–1361. PMLR, 2021.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. doi: 10.48550/arXiv.2511.21631. URL https://arxiv.org/abs/2511.21631.

Shuanghao Bai, Jing Lyu, Wanqi Zhou, et al. Latent reasoning VLA: Latent thinking and prediction for vision-language-action models. In *Proceedings of the International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.01166.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, et al. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010.

Qingwen Bu, Yanting Yang, Jisong Cai, et al. UniVLA: Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025. URL https://arxiv.org/abs/2505.06111.

Sera Buyukgoz, Jasmin Grosinger, Mohamed Chetouani, and Alessandro Saffiotti. Two ways to make your robot proactive: Reasoning about human intentions or reasoning about possible futures. *Frontiers in Robotics and AI*, 9:929267, 2022. doi: 10.3389/frobt.2022.929267. URL https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2022.929267/full.

Feilong Chen, Fandong Meng, Jiaming Xu, Peng Li, Bo Xu, and Jie Zhou. DMRM: A dual-channel multi-hop reasoning model for visual dialog. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7504–7511, 2020. URL https://arxiv.org/abs/1912.08360.

Wei Chow, Jiageng Mao, Boyi Li, Daniel Seita, Vitor Guizilini, and Yue Wang. PhysBench: Benchmarking and enhancing vision-language models for physical world understanding. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2501.16411.

Kuofei Fang et al. RobotEQ: Towards social proactive intelligence in embodied agents. *arXiv preprint arXiv:2605.06234*, 2026. URL https://arxiv.org/abs/2605.06234v3.

Yuxuan Fu et al. PRISM: Festina lente proactivity—risk-sensitive, uncertainty-aware deliberation for proactive agents. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2602.01532.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323, 2020.

Google DeepMind. Gemini 3.8 Flash. Gemini API, model `gemini-3.8-flash`, 2026. Accessed September 2026.

Jennifer Grannen, Siddharth Karamcheti, Blake Wulfe, and Dorsa Sadigh. ProVox: Personalization and proactive planning for situated human-robot collaboration. *IEEE Robotics and Automation Letters*, 2025. URL https://arxiv.org/abs/2506.12248.

Wen-Han Hsieh, Elvis Hsieh, Dantong Niu, Trevor Darrell, Roei Herzig, and David M. Chan. Do what? Teaching vision-language-action models to reject the impossible. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025. doi: 10.48550/arXiv.2508.16292. URL https://aclanthology.org/2025.findings-emnlp.635/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. doi: 10.48550/arXiv.2106.09685. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Yu-Chiang Frank Wang, and Fu-En Yang. ThinkAct: Vision-language-action reasoning via reinforced visual latent planning. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2507.16815.

Divyansh Kaushik, Eduard Hovy, and Zachary C. Lipton. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*, 2020.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag 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. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, PMLR 270, pp. 2679–2713, 2024. doi: 10.48550/arXiv.2406.09246. URL https://arxiv.org/abs/2406.09246.

Lin Li et al. Causal world modeling for robot control. In *Proceedings of Robotics: Science and Systems (RSS)*, 2026. URL https://arxiv.org/abs/2601.21998.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

Jindřich Libovický, Jindřich Helcl, and David Mareček. Input combination strategies for multi-source transformer decoder. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 253–260, 2018. URL https://aclanthology.org/W18-6326/.

Yaxi Lu, Shenzhi Yang, Cheng Qian, et al. Proactive agent: Shifting LLM agents from reactive responses to active assistance. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.12361.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026. doi: 10.48550/arXiv.2603.04356. URL https://arxiv.org/abs/2603.04356.

NVIDIA. GR00T N1.6. https://huggingface.co/nvidia/GR00T-N1.6-3B, 2025.

NVIDIA, Johan Bjorck, Fernando Castañeda, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. doi: 10.48550/arXiv.2503.14734. URL https://arxiv.org/abs/2503.14734.

Physical Intelligence et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025. URL https://arxiv.org/abs/2504.16054.

Qwen Team. Qwen3.8-Flash. Alibaba Cloud DashScope API, model `qwen3.8-flash`, 2026. Accessed September 2026.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi Robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2502.19417.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Haoyu Wang, Sihang Jiang, Xiangru Zhu, Yuyan Chen, Xiaojun Meng, Jiansheng Wei, Yitong Wang, and Yanghua Xiao. MMIFEvol: Towards evolutionary multimodal instruction following. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 26206–26214, 2026a. doi: 10.1609/aaai.v40i31.39824. URL https://doi.org/10.1609/aaai.v40i31.39824.

Siyin Wang et al. RoboOmni: Proactive robot manipulation in omni-modal context. In *International Conference on Learning Representations (ICLR)*, 2026b. URL https://arxiv.org/abs/2510.23763.

Wei Wu, Fan Lu, Yunnan Wang, et al. A pragmatic VLA foundation model. *arXiv preprint arXiv:2601.18692*, 2026. URL https://arxiv.org/abs/2601.18692.

Weicai Yan, Yuhong Dai, Qi Ran, Haodong Li, et al. Proact-VL: A proactive VideoLLM for real-time AI companions. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2603.03447.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Seonghyeon Ye et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026. URL https://arxiv.org/abs/2602.15922.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. MM-Vet: Evaluating large multimodal models for integrated capabilities. In *Proceedings of the International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2308.02490.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023.

Zeyi Zhang et al. ProAct: Harnessing streaming motion generation and agentic reasoning for real-time embodied social interaction. *ACM Transactions on Graphics (SIGGRAPH Asia 2026)*, 2026. URL https://arxiv.org/abs/2602.14048.

A Implementation and Training Details

A.1 Input layout and generation

In Equation 3, c contains the scene images and description, the current task, and the capability policy; ℓ^{p} and ℓ^{h} contain the proposal and the optional human request. The token range of each source is taken directly from the model's chat template, including its field headings, and an absent source has an empty range. The system prompt, image tokens, and the fixed text that opens the model's reply stay as in the original model, and positions follow Qwen3-VL's own rotary encoding.

The source states that SHUTTLE reads (Equations 4–5) are taken at text layer 18 after self-attention and its residual. Because every source comes before the decision, these states are computed once when the prompt is encoded and reused at every generated token, so generating a decision needs no extra pass over the input.

A.2 Module dimensions, training data, and training stages

The scene read uses a parameter-free RMS normalization, computed in FP32:

$$\mathcal{N}_{\mathrm{c}}(x) = \frac{x}{\sqrt{d^{-1}\sum_{j=1}^{d} x_j^2 + 10^{-6}}}. \tag{11}$$

The decision query uses an affine LayerNorm. The query and the three key/value projections have no bias term; the first layer of the fusion MLP (W_{in} in Equation 6) has one, and the output projection W_{out} has none and is initialized to zero, so the module starts as the identity. For hidden width d, read rank k, and fusion width w, the module adds

$$P_{\textsc{Shuttle}} = 7dk + 2d + (2d + 3k + 1)w \tag{12}$$

parameters, which is 5,779,573 for $d = 4096$, $k = 128$, and $w = 245$ (Table 4).

Training data. All methods train on the same 6,268 complete-decision demonstrations from RoboCasa365 kitchen scenes (Nasiriany et al., 2026). No training example shares a scene family with TWINE, and no test request reaches a token-level Jaccard similarity of 0.6 with any training request (Figure 7). The collection has three parts (Table 3). A scenario block of 620 examples contains 420 authored kitchen scenarios, covering ambiguous references, fixed appliances, outdated or missing tasks, cabinet and stove operations, placing objects at occupied or free destinations, taking an item out before closing a door, prerequisites such as an unknown temperature, and prohibitions; its other 200 examples state a fact or its opposite in one source over 60 checked images while the reference decision stays the same. An expansion of 4,052 examples, kept after visual inspection and blind comparison, fills each template with several request variants on a pair of images of the same scene; the request text is identical for both images, and in most pairs the reference decision differs. A composition block of 1,596 examples, built from training images only, adds a step before the proposal, withdraws a request, gives two destinations, or limits a request to one object or a set. Stage 1 runs 10,964 updates, presenting each scenario example six times, each composition example twice, and each expansion example once.

Training stages. Stage 1 trains the LoRA alone over the full plan. Stage 2 starts every method from the stage-1 checkpoint of the same seed and runs the first 6,000 updates of the same plan, training the added module at a learning rate of 10^{-4} and the LoRA at 10^{-5}; the complete-answer loss (Equation 9) updates both. For the four methods with separate source reads (SHUTTLE, merged-text reads, hierarchical attention, and DMRM-style fusion), each update sets the proposal read to zero with probability 0.15, and otherwise the request read with probability 0.15, without changing the input text. Checkpoints are saved every 2,000 updates; Direct, which is stage 2 without a module, also keeps its stage-1 checkpoint as a candidate. For each method and seed, the candidate with the highest complete-decision pass rate on 198 development cases (99 pairs from 12 scenes whose families and images appear in neither training nor test; ties go to the earlier checkpoint) is evaluated once on TWINE. Figure 8 shows every candidate.

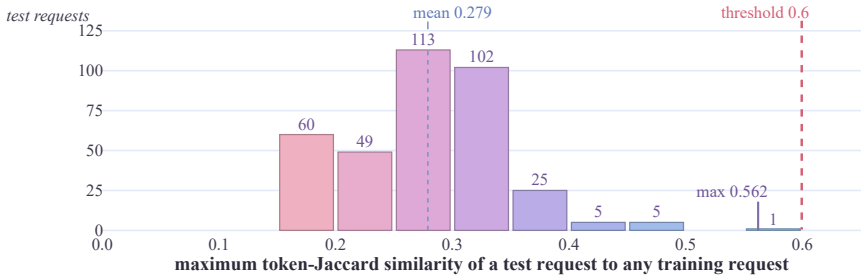

Figure 7: **Train/test request similarity.** For each of the 360 test requests, the highest token-level Jaccard similarity to any of the 6,268 training requests. No test request reaches 0.6; the maximum is 0.562 and the mean 0.279.

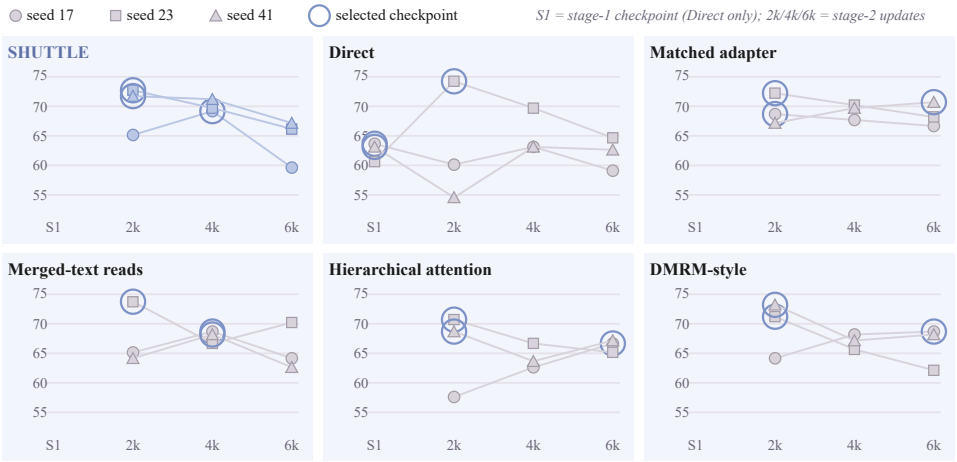

Figure 8: **Checkpoint selection on the development set.** Complete-decision pass on the 198 development cases for every candidate (Gemini-3.8-Flash scorer). S1 is the stage-1 checkpoint, a candidate for Direct only; 2k–6k are stage-2 updates. Circles mark the selected checkpoint.

Table 3: **Training-collection composition.** Expansion rows come from image pairs of the same scene; within a pair the request text is identical while the reference decision can differ.

Part	Examples
Scenario block: authored kitchen scenarios	420
Scenario block: a fact or its opposite (decision fixed; 60 images)	200
Expansion: visually grounded rows	4,052
Composition block: rows built from training images	1,596
Total	6,268

The matched adapter is $\mathrm{Linear}(705, 4096) \circ \mathrm{GELU} \circ \mathrm{Linear}(4096, 705) \circ \mathrm{RMSNorm}$, with a bias-free output projection. Merged-text reads keep a separate key/value projection for each text source, but each head attends over the proposal and the request together, so the two sources share one softmax; a head whose own source is absent returns zero. When only one text source is present, merged and separate reads compute the same result.

Table 4: **Matched learned controls.** Every added module has a zero-initialized output projection.

Method	Added computation at the decision positions	Parameters
Direct	LoRA fine-tuning only; no added module.	0
Matched adapter	RMSNorm, width-705 bottleneck, GELU, output projection.	5,780,161
Merged-text reads	Two distinct text heads over combined $[\ell^{\mathrm{p}}; \ell^{\mathrm{h}}]$; unchanged scene read and fusion.	5,779,573
SHUTTLE	Separately addressable $c/\ell^{\mathrm{p}}/\ell^{\mathrm{h}}$ reads and joint residual fusion.	5,779,573
Hierarchical attention	Per-source attention followed by attention over the source contexts.	5,775,360
DMRM-style	Tri-attention reading with source interaction.	5,783,975

Table 5: **Shared decision-module settings.** Training and generation settings for the decision comparisons.

Setting	Shared decision-component recipe
Backbone	Qwen3-VL-8B-Instruct; 36 text layers; hidden width 4,096.
Native adaptation	LoRA rank 16, $\alpha = 32$, dropout 0.05 on query/key/value/output projections.
Module placement	Text layer 18, after the attention residual and before the native FFN.
SHUTTLE dimensions	Read rank 128; shared query; three key/value pairs; fusion width 245.
Precision	Fixed base and visual weights in BF16; trainable parameters and AdamW moments in FP32; BF16 autocast.
Optimization	AdamW; learning rate 10^{-4} (stage 2: 10^{-4} for the added module, 10^{-5} for the LoRA); weight decay 0.01; gradient-norm clipping 1.0.
Batch and schedule	One example per update without gradient accumulation; constant learning rate without warmup.
Starting point	Stage 1: the shared warm-started LoRA checkpoint; stage 2: the same-seed stage-1 checkpoint; added-module initialization seed 170128.
Data and updates	6,268 examples; stage 1 runs all 10,964 updates, stage 2 the first 6,000 of them.
Token limits	Prefix 4,096; full target including EOS 256; decision 512; understanding 768 (1,536 when an answer is regenerated after a format error).
Reported decision runs	Three paired training seeds (17, 23, and 41); per method and seed, the stage-2 checkpoint (every 2,000 updates; for Direct also the stage-1 checkpoint) with the best development score, tested once.
Decoding	Greedy generation with native caching and EOS stopping.

A.3 PROPOSER TRAINING

Task. The proposer takes one 256×256 RGB image as input and decides whether the scene requires an action. It covers six problems: water flowing from a faucet, a burner left on, and an open cabinet, dishwasher, refrigerator, or drawer. These map to four skill families (water, stove, door, and drawer) and three operators (stop flow, turn off, and close). The proposer answers with an opportunity flag, a propose-or-hold mode, a skill family, an operator, and a short proposal sentence, which becomes ℓ^{p}; the other fields are used for scoring and routing. The image is its only input. The closed-loop experiments use the same checkpoint.

Training. The proposer is trained on 1,416 left-camera images grouped into pairs that show the same scene with the problem present and resolved; for doors, the resolved image shows the door fully closed. Training data are weighted so that the six problems, the scenes, and the two states count equally, and duplicate images share one weight. Both images of a pair pass through the model separately with the same prompt, and the loss is Equation 2 with $\lambda = 0.1$. The pairwise term ranks the two states, while the answer and flag losses still require the proposer to propose on the first image and hold on the second. At deployment the proposer reads one image. Training starts from the untuned Qwen3-VL-8B-Instruct with a fresh rank-16 LoRA and a trainable vision encoder, with learning rates of 2×10^{-5} for the language model and 2×10^{-6} for the vision encoder; each update

accumulates gradients over 16 examples that together cover all six problems. The final proposer averages the weights of the checkpoints after 89 and 178 updates.

B Evaluation Details

B.1 Proposer evaluation

Evaluation. The evaluation set has 198 held-out observations in 84 problem–scene groups, each with images of the same scene with the problem present and resolved. Joint accuracy requires the opportunity flag, mode, skill family, and operator all to be correct; we average within each group and then weight groups equally. Recall and restraint use the same weights: recall is the weighted share of positive observations on which the proposer reports an opportunity, restraint is the weighted share of negative observations on which it holds, and paired accuracy requires both answers of an on/off pair to be correct. FN and FP in Table 6 are unweighted counts over the 83 positive and 115 negative observations. For reference, a proposer that always holds reaches 56.55% joint accuracy. The zero-shot baseline is the untuned Qwen3-VL-8B-Instruct with the same prompt, chat template, greedy decoding, and a 96-token output limit. All 198 raw answers and their scores will be released.

Table 6: **Proposer accuracy.** Joint, recall, and restraint are percentages; pairs count on/off pairs with both answers correct. FN and FP count missed opportunities and false proposals. Both models use the same backbone, single RGB input, prompt, answer format, and greedy decoding.

Model	Joint	Recall	Restraint	Pairs	FN	FP
Qwen3-VL-8B-Instruct zero-shot	41.27	86.76	23.16	7/84	12	88
Ours	**89.88**	83.56	96.84	64/84	16	3

Results. Fine-tuning raises joint accuracy from 41.27% to 89.88% and restraint from 23.16% to 96.84%. False proposals fall from 88 to 3, and correctly answered on/off pairs rise from 7 to 64 of 84. The improvement therefore comes from restraint on scenes that require no action.

B.2 Construction and source information

The test set has 60 layout–style families, 76 native 256×256 images, and 180 pairs of cases, with six cases per family. No family used for training or checkpoint selection appears in the test set. Each case records its input, the sources the answer depends on, the permitted actions, the required obligations, and the equivalent decisions that also count as correct. In scene-change pairs, both cases share the proposal made on the earlier image, and the new image may require the decision to revise it.

Module A contains 30 pairs that exchange action order, 15 that distinguish retaining from canceling a goal, and 15 that change only a prerequisite. Module B contains 15 pairs that change whether the required confirmation has arrived and 15 that distinguish a relevant from an irrelevant signal. Module C contains 15 pairs that change the destination of an added step and 15 that change the scope between one object and a set of two. Module D contains 15 pairs with a permitted or forbidden extra action and 15 with a current or withdrawn request. Module E contains 15 same-image rule changes and 15 same-rule scene changes. These allocations yield the counts in Table 1.

B.3 Understanding and decision scores

Each understanding item has a dimension (Table 7), a question, its acceptable answers, and the sources it depends on. For each dimension, we pool the applicable items over cases and seeds, and the understanding score averages the three accuracies. Accuracy counts the items a method answered in a parseable form; items marked as not applicable to a case are left out. $U3$ items can relate the scene to the proposal, the scene to the request, or the two intents to each other.

Table 7: **Five diagnostic dimensions.** The understanding probes are generated independently of the decision. $D2$ is an expression diagnostic.

Dimension	Definition
$U1$	Task-relevant visual objects, attributes, relations, and states
$U2$	Goals, negation, quantifiers, and conditions in either intent source
$U3$	Reference binding, local modification, and cross-source relations
$D1$	Proportion of required correctness checks, including necessary content, satisfied within the case
$D2$	Clarity, coherence, and absence of obstructive redundancy; $0/0.5/1$ anchors

For decisions, $D1$ is computed within each case and averaged over cases and seeds. A decision passes completely only if, under both judging orders, it meets every critical obligation, covers the necessary content, is consistent as a whole, and is complete; failed or unfinished outputs count as failures. Any equivalent valid plan counts as correct, and a decision need not name its sources or repeat connectives.

Table 10 adds finer diagnostics. Control accuracy checks whether the decision correctly acts, waits for an explicit trigger, or holds without one. Authority accuracy checks whether it relies on the sources that actually determine the answer, and direction accuracy requires both. Authority and direction are scored on the 255 cases per seed whose deciding source is clear. Content accuracy averages $D1$ over each method's direction-correct cases, and CDPass counts decisions that pass completely and also have the correct direction. $D2$ is reported for each judging order.

B.4 Semantic review

Gemini-3.8-Flash (Google DeepMind, 2026), run at temperature 0 with a medium thinking level, scores anonymized answers against checked scene facts and the obligations of each case. Each answer is judged in two orders: order A lists the proposal before the request and keeps the rubric checks in their original order, and order B lists the request first and reverses the checks. A check passes only if it passes in both orders. We calibrated the scorer on 80 answers spanning all five modules, 40 correct paraphrases of reference decisions and 40 answers that each contain one critical error: in both orders it accepted all 40 correct answers, rejected all 40 wrong ones, and left none undecided.

B.5 Human evaluation

Two annotators independently judged complete decisions on 64 test cases from 30 scene families: one SHUTTLE and one Direct response per case, both from training seed 17, for 128 responses in total. Method identity was hidden. Before adjudication, the annotators agree on 116 of 128 responses (90.63%, Cohen's $\kappa = 0.745$; family-bootstrap 95% intervals [81.25, 99.11] and [0.516, 0.970]). The authors adjudicated the 12 disagreements, six per method. Against the adjudicated labels, the Qwen3.8-Flash scorer (Qwen Team, 2026), with the same rubric, agrees on 118 of 128 responses (92.19%, $\kappa = 0.800$). Gemini-3.8-Flash, which scores the test results, agrees with an author's separate blind judgment of the same 128 responses on 119 (92.97%, $\kappa = 0.849$). This check is scored separately from the three-seed results in Table 8. On the test decisions, the Qwen3.8-Flash scorer and Gemini-3.8-Flash agree on 92.70% of 6,480 complete-decision judgments (Cohen's $\kappa = 0.824$). The invariance, source-swap, and proposal-removal analyses below are also scored by Gemini-3.8-Flash, except that source attribution is checked by exact matching.

B.6 Matched baselines

The main comparison has one SHUTTLE configuration, four baselines, and the merged-text ablation, each trained with seeds 17, 23, and 41. Direct adds no module, the adapter matches the size of SHUTTLE, and merged-text reads pool the two intents. Hierarchical attention attends within each source and then across sources, and the DMRM-style model reproduces the reading and interaction structure of DMRM on the same backbone (Libovický et al., 2018; Chen et al., 2020).

Untuned and frontier references. To gauge how much of the task the decision module has to learn, we run two references on the 360 test cases with the same system prompt and input layout as the trained methods, deterministic decoding, and the same Gemini-3.8-Flash scorer. The untuned Qwen3-VL-8B-Instruct backbone without any adapter passes 19.44% of the cases completely. Gemini-3.8-Flash answering zero-shot passes 46.11%, with its answers scored by the same model. Both fall below the three-seed mean of SHUTTLE (71.85%, Table 8).

B.7 AGGREGATION, PAIRED COMPARISONS, AND COST

Decision scores weight cases and the three seeds equally (Table 12), and a controlled pair counts as correct only when both of its cases pass (Table 9). For confidence intervals we resample scene families 10,000 times, keeping all cases, pairs, seeds, and methods of a sampled family together, and take percentile 95% intervals. The five complete-pass contrasts in Table 11 form one Holm family and the two diagnostic contrasts in Table 13 a second; other diagnostics report effects and intervals only. The p-values are two-sided bootstrap p-values, Holm-adjusted within each family. Figure 9 shows each seed, Figure 10 the paired differences with their intervals, and Figure 11 the pass rate of each module.

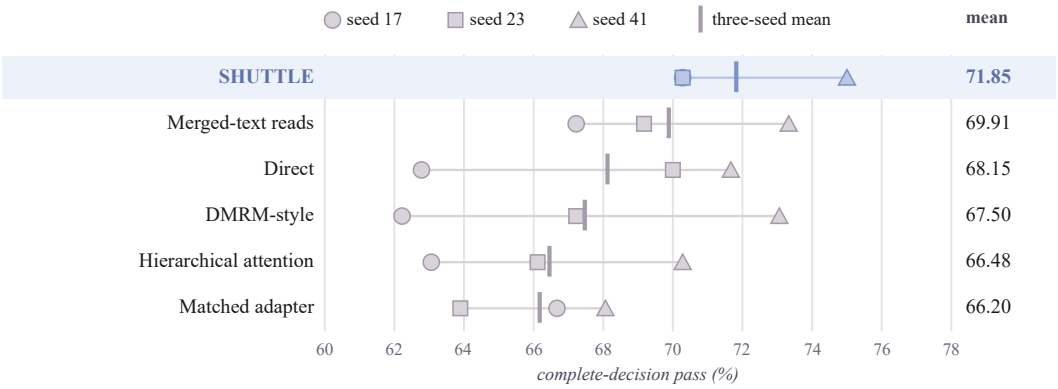

Figure 9: **Complete-decision pass by seed.** Each marker is one seed (360 test cases, Gemini-3.8-Flash scorer); the bar marks the three-seed mean (Table 8). SHUTTLE is highest on every seed.

Table 8: **Understanding and complete decisions on TWINE.** Percentages over 360 cases × 3 seeds. Understanding averages the $U1$–$U3$ accuracies over answered items (Table 9); Complete is the complete-decision pass rate of Equation 10; s/case is the mean response time in seconds, measured for each checkpoint in its own process on one idle NVIDIA RTX PRO 6000 Blackwell GPU (60 test cases, 3 warm-up cases excluded). $D2$ A/B gives $D2$ under the two judging orders.

Method	Understanding	$U3$	$D1$	$D2$ A/B	Complete	s/case
Direct	75.04	64.94	75.20	94.31/93.66	68.15	0.70
Matched adapter	75.50	65.16	72.46	93.67/92.24	66.20	0.70
Merged-text (ablation)	75.84	66.16	76.04	94.78/93.38	69.91	0.76
Hierarchical attention	75.07	64.85	73.21	93.04/91.67	66.48	0.77
DMRM-style	74.08	63.33	73.11	93.13/92.02	67.50	0.73
SHUTTLE	75.74	66.41	78.23	96.11/95.49	**71.85**	0.71

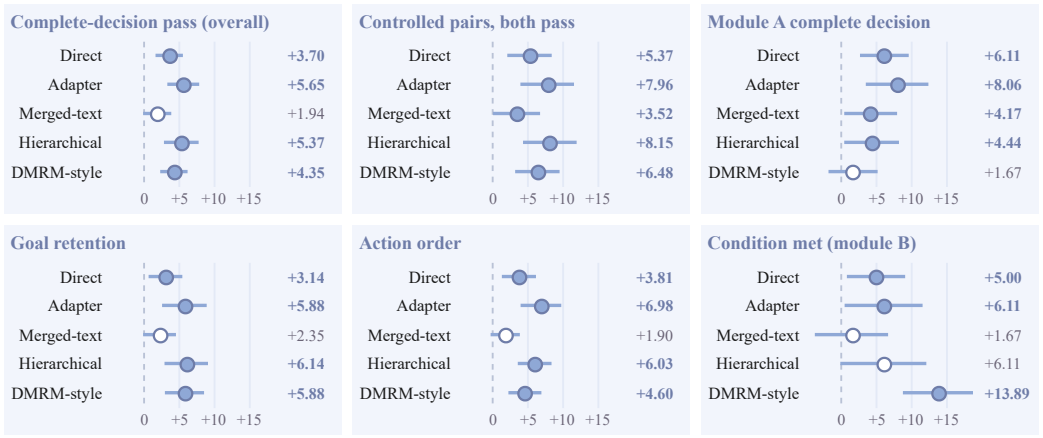

Figure 10: **Paired differences, SHUTTLE minus each method** (pp). Complete-decision pass (Table 11) and the diagnostic measures: controlled pairs (180 per seed), module A (120 cases), goal retention (255 cases per seed), action order (210 cases per seed), and the module-B condition check (60). Lines are unadjusted 95% scene-family bootstrap intervals (10,000 draws); filled dots mark intervals that exclude zero.

	Module A Goals and order 120 cases/seed	**Module B** Triggers 60 cases/seed	**Module C** Local revision 60 cases/seed	**Module D** Mixed validity 60 cases/seed	**Module E** Scene constraints 60 cases/seed	**All** modules 360 cases/seed
SHUTTLE	**74.17**	**88.89**	**95.00**	52.78	**46.11**	**71.85**
Merged-text reads	70.00	87.78	**95.00**	51.11	45.56	69.91
Direct	68.06	83.89	94.44	48.33	**46.11**	68.15
DMRM-style	72.50	74.44	89.44	**53.89**	42.22	67.50
Hierarchical attention	69.72	77.22	88.89	52.78	40.56	66.48
Matched adapter	66.11	82.78	90.00	48.89	43.33	66.20

complete-decision pass (%), mean of seeds 17, 23, 41; bold = best in column

Figure 11: **Complete-decision pass by module** (%, mean of three seeds, Gemini-3.8-Flash scorer), computed from the same per-case verdicts as Table 8. Bold marks the best method in each column.

Quality scores in the tables are percentages ($D1$ and $D2$ are multiplied by 100), and differences are in percentage points (pp). Table 8 reports decision time per case in seconds; SHUTTLE peaks at 16.6 GiB of memory during decision generation, which fits on a single 24 GB GPU.

Table 9: **Understanding and paired behavior.** Quality scores in %. $U1$–$U3$ are accuracies over the understanding items each method answered in a parseable form (870 to 1,004 of 1,080 cases across methods, three seeds); Understanding is their mean. $D2$ A/B gives $D2$ under the two judging orders. Complete pairs require both answers to pass, over 180 pairs per seed.

Method	Understanding	$U1$	$U2$	$U3$	$D2$ A/B	Complete pairs
Direct	75.04	76.79	83.40	64.94	94.31/93.66	47.96
Matched adapter	75.50	77.14	84.21	65.16	93.67/92.24	45.37
Merged-text (ablation)	75.84	76.80	84.54	66.16	94.78/93.38	49.81
Hierarchical attention	75.07	77.68	82.66	64.85	93.04/91.67	45.19
DMRM-style	74.08	76.03	82.89	63.33	93.13/92.02	46.85
SHUTTLE	75.74	76.80	84.02	66.41	96.11/95.49	53.33

Table 10: **Direction and conditional content diagnostics** (%). Control: the decision correctly acts, waits for a trigger, or holds. Authority: it relies on the sources that determine the answer. Direction: both are correct. Content: $D1$ on direction-correct cases. CDPass: the decision passes completely and its direction is correct. Control, Authority, Direction, and CDPass use the source-identifiable subset (255 cases per seed, 765 over three seeds). Content uses each method's direction-correct subset. Gemini-3.8-Flash scorer.

Method	Control	Authority	Direction	Content	CDPass
Direct	98.30	91.11	90.59	88.02	74.25
Matched adapter	98.43	93.20	93.20	82.55	71.37
Merged-text (ablation)	98.69	93.73	93.46	87.29	76.21
Hierarchical attention	98.82	92.68	92.68	84.37	72.29
DMRM-style	97.65	91.90	91.50	84.64	73.07
SHUTTLE	98.69	93.86	93.59	89.96	78.43

Table 11: **Paired comparisons of SHUTTLE with each method.** Same case and seed, 60 scene-family clusters; Holm-adjusted over the five contrasts in this table.

SHUTTLE minus	CompletePass Δ (pp)	95% CI (pp)	Holm p
Direct	+3.70	[+1.76, +5.65]	0.0008
Matched adapter	+5.65	[+3.43, +7.96]	$<$0.001
Merged-text (ablation)	+1.94	[+0.00, +3.98]	0.0604
Hierarchical attention	+5.37	[+2.96, +7.87]	$<$0.001
DMRM-style	+4.35	[+2.41, +6.30]	$<$0.001

Table 12: **Per-seed complete-decision pass rates** (%, 360 cases per seed).

Method	Seed 17	Seed 23	Seed 41
Direct	62.78	70.00	71.67
Matched adapter	66.67	63.89	68.06
Merged-text (ablation)	67.22	69.17	73.33
Hierarchical attention	63.06	66.11	70.28
DMRM-style	62.22	67.22	73.06
SHUTTLE	70.28	70.28	75.00

B.8 GOAL RETENTION AND ACTION ORDER

Table 13 gives two views of goal retention and action order. Module A (120 cases, 60 controlled pairs, 60 scene families) is built around these two abilities, and its complete-decision pass rate uses all 120 cases. Across the full test set, 255 cases per seed carry an obligation to keep a goal and 210 an obligation on action order, and we score these obligations as diagnostics. The prompt asks for the current action and every later step the inputs explicitly require, and a case counts as correct for goal retention or order only when every obligation of that type passes under both judging orders. The module-B check in Figure 10 scores the trigger obligation of the 60 module-B cases per seed: it passes when the decision waits until the request's condition has been met.

Table 13: **Module A and full-set temporal diagnostics.** Rates in %, three-seed means; cases per seed are the applicable counts. The two diagnostic contrasts have Holm-adjusted p-values 0.0100 and 0.0052; family-clustered intervals are [0.78, 5.56] and [1.44, 6.27] percentage points.

Endpoint	Cases/seed	SHUTTLE	Direct	Δ (pp)
Module A complete decision	120	74.17	68.06	+6.11
Goal retention (full test set)	255	83.14	80.00	+3.14
Action order (full test set)	210	82.86	79.05	+3.81

B.9 INVARIANCE TO MEANING-PRESERVING CHANGES

The invariance set (Table 14) tests whether a decision stays correct when the input changes without changing its meaning. One test case per scene family, 12 per module, receives such a change: an equivalent paraphrase in 30 cases and an added irrelevant sentence in the other 30, six of each per module. Each changed case keeps the obligations of its source case and forms an invariance pair with it. Source answers come from the main runs, and all six models answer the 60 changed cases at each of the three seeds. A pair counts as both-correct when the source and the changed case both pass complete-decision scoring. Within each seed, the both-correct and correct-to-error rates are computed over the 60 pairs, and retention divides the both-correct count by the number of correctly answered source cases; each metric is then averaged over the three seeds. All six models keep most of their correct answers under these changes, with retention between 87.27% and 93.86%.

Table 14: **Invariance pairs** (%, equal-seed means). Retention divides both-correct pairs by correctly answered source cases; the other two rates divide by all 60 pairs per seed. Gemini-3.8-Flash scorer.

Method	Retention	Both correct	Correct→error
Direct	93.56	65.00	4.44
Matched adapter	93.86	60.00	3.89
Merged-text (ablation)	92.74	63.89	5.00
Hierarchical attention	87.27	53.89	7.78
DMRM-style	89.98	60.00	6.67
SHUTTLE	91.35	64.44	6.11

B.10 SOURCE IDENTIFICATION ON SOURCE-SWAPPED PAIRS

This set tests whether a decision module identifies the source of each instruction. Each of 60 scene families contributes one source-swapped pair. In the first case, the proposal and the request carry two conflicting instructions, for example the same two actions in opposite orders; the swapped case exchanges the two texts, so the correct decision changes with the swap. For every case, two separate questions ask which source contains a given complete instruction, with the answers ROBOT, HUMAN, BOTH, and NEITHER; the questions never enter the decision context. Answers are parsed by exact matching without a model judge, and a missing or unparsed answer counts as wrong. SHUTTLE, merged-text reads, and Direct answer all 120 cases at seeds 17, 23, and 41.

SHUTTLE names the source of an instruction correctly in 87.50% of 720 answers, against 84.72% for merged-text reads (+2.78 points, 95% CI [0.56, 5.00], $p = 0.014$; Table 15) and 84.03% for Direct (+3.47 points, 95% CI [1.67, 5.28], $p < 0.001$). It also answers all four source questions of a pair correctly most often, in 58.89% of pairs against 45.00% and 41.11%, and attributes an instruction to both sources least often, in 72 of 720 answers against 87 for each alternative. On the same pairs, SHUTTLE also passes more complete decisions than merged-text reads (83.61% against 77.78%).

Table 15: **Source-swapped pairs** (%, 60 pairs per seed, three seeds). Pair both requires both decisions of a pair to pass (Gemini-3.8-Flash scorer); accuracy is over 720 source answers per method; pair all four requires all four source answers of a pair to be correct.

	Decision		Source identification	
Method	Complete	Pair both	Accuracy	Pair all four
Merged-text (ablation)	77.78	65.00	84.72	45.00
SHUTTLE	83.61	70.56	87.50	58.89

B.11 REMOVING THE PROPOSAL FROM THE INPUT

To measure how much the decision relies on the robot's own proposal, we delete the entire proposal field from the inputs of the trained SHUTTLE and Direct checkpoints, without retraining, and keep the request text unchanged, including any reference to the earlier suggestion. All 360 test cases contain a request. Without the proposal, complete-decision pass falls from 71.85% to 2.69% for SHUTTLE and from 68.15% to 1.85% for Direct (Table 16). The decision thus depends on the robot's own proposal in nearly every case, and the request alone cannot replace it.

Table 16: **Complete-decision pass with and without the proposal** (%, 360 cases, three seeds, Gemini-3.8-Flash scorer). Intervals from the scene-family bootstrap.

Method	Full input	Proposal removed	Change (pp) [95% CI]
SHUTTLE	71.85	2.69	−69.17 [−73.15, −65.19]
Direct	68.15	1.85	−66.30 [−70.19, −62.22]

B.12 INSERTION DEPTH

SHUTTLE has a single insertion point in the 36-layer decoder. To test this choice, the same rank-128 module is placed at layer 9, 18, or 27 (shallow, middle, deep) with everything else fixed: the same two-stage recipe on the 6,268 training examples, three seeds (17, 23, and 41), checkpoint selection on the 198 development cases, 360 test cases, and the same number of modules and parameters. Layer 18 reuses the checkpoints of the main run, and layers 9 and 27 are trained anew for each seed. All three placements are scored with the same rubric and Gemini-3.8-Flash scorer as the main tables. Layer 18 gives the highest complete-decision pass rate (Table 17), and layers 9 and 27 stay within 0.74 points of it; direction accuracy lies between 91.90% and 94.38% at all three depths. The module therefore works at shallow, middle, and deep placements, and layer 18 is kept as the default.

Table 17: **Insertion depth of SHUTTLE** (%, three seeds, 360 cases per seed, Gemini-3.8-Flash scorer; the layer-18 row is the main run). Direction accuracy requires both the control and the authority judgment to be correct and uses the source-identifiable subset of Table 10. Intervals from the paired scene-family bootstrap.

Insertion layer	Complete	Direction acc.	Δ Complete vs. 18 [95% CI]
9 (shallow)	71.57	94.38	−0.28 [−2.41, +1.76]
18 (middle, deployed)	**71.85**	93.59	–
27 (deep)	71.11	91.90	−0.74 [−2.50, +1.02]

B.13 MECHANISM PROBE

To locate where the gain of SHUTTLE enters the decision, we rerun the selected checkpoints of seeds 17, 23, and 41 on the 360 test cases and change the computation only at test time, leaving weights, inputs, and decoding as they are. Zeroing the module's residual (Equation 7) lowers complete-decision

pass from 70.28% to 66.94% at seed 17, from 70.28% to 68.89% at seed 23, and from 75.00% to 72.22% at seed 41, against 62.78%, 70.00%, and 71.67% for Direct. The residual strength is the norm of the module's residual divided by the norm of the decision state, averaged over generated tokens; over the three seeds it is 6.7% on hold decisions, 5.9% on wait decisions, and 4.9% on act decisions (Figure 6b). On the request path, the most-attended request token at a generated position is most often an object attribute, such as a color or a side, or a temporal word such as *yet*, *finished*, or *current*.

B.14 Closed-loop evaluation: autonomous behavior and human intervention

B.14.1 Systems and shared environment

The evaluation uses RoboCasa365 kitchens with the PandaOmron robot (Nasiriany et al., 2026). TWILL uses the seed-17 proposer on a 256×256 left-camera image, a decision module, and the official frozen GR00T N1.6 executor (NVIDIA et al., 2025; NVIDIA, 2025). TWILL with Direct uses decision seeds 17, 23, and 41, and TWILL with SHUTTLE uses the SHUTTLE checkpoints of the same seeds on the same scene resets. All systems (Table 18) start from the same initial states, see the same events and observations, act through the same interface, and are scored by the same rules.

VA starts from the same GR00T initialization and is trained and evaluated on RGB and proprioception without language. Its training trajectories include acting, holding in clean scenes, stopping after completion, and problems that appear mid-episode, and it chooses between acting and stopping with a learned head whose threshold is set on development scenes. The matched VLA is trained on the same trajectories with the same number of updates but also receives language. VA and the matched VLA output low-level actions directly, while TWILL, the mission-prompted VLA, the instruction-driven VLA, and the oracle all use the frozen GR00T N1.6 executor. The instruction-driven VLA is this executor given the person's request as its instruction, and the mission-prompted VLA receives the same mission at every step: "Monitor the kitchen and fix any problem you see: turn off a running faucet or a lit stove, and close any open cabinet door or drawer. Do nothing if everything is already fine."

The VLM commander replaces TWILL's proposer and decision module with a single vision-language model that drives the same frozen executor. In every cycle the model reads the current camera images, the standing task, the list of instructions the executor can carry out, and the person's current message if there is one, and returns one listed instruction or HOLD, in JSON with a short note on the visible evidence. It is the untuned Qwen3-VL-8B-Instruct, the backbone of TWILL, run once with greedy decoding.

Table 18: **Closed-loop systems.**

System	Policy input and experimental role
TWILL with SHUTTLE	RGB proposal, scene, and received human input; complete system
TWILL with Direct	The same chain with the native decision policy; decision-stage control
VA	RGB and proprioception, no language; autonomous-behavior control
Mission-prompted VLA	RGB, proprioception, and one fixed kitchen mission; the executor of TWILL without proposals
Instruction-driven VLA	RGB, proprioception, and the human instruction actually received
Matched VLA	Language-conditioned counterpart to VA with the same action training
VLM commander (Qwen3-VL-8B)	RGB, standing task, and received human input; the backbone of TWILL without our training, driving the executor
Oracle → executor	Correct target and goal after event onset; execution reference

B.14.2 Physical tasks and episode conditions

Tasks cover turning off a faucet, closing a cabinet door, and closing a drawer (conditions in Table 19), with the same checks for acting on the right object, visibility, and physical execution. Each scene fixes the task, layout, style, reset seed, target fixture, initial state, target state, the states that must stay untouched, and the success conditions for each object. A new RGB observation is taken after

every change to the initial state and after every event. An oracle run confirms that each target can be reached through the shared interface. A clean scene contains none of the problems defined for it.

Table 19: **Autonomous and human-intervention conditions.** Each count is the number of evaluated episodes per method and applicable seed: 11 scene resets, each run twice with two executor random seeds (17 and 23).

Condition	Required behavior	Episodes
Initial problem	Resolve the visible problem and protect other states	22
Problem during monitoring	Hold before onset; detect and resolve the new problem	22
Clean scene	Maintain the correct state without unnecessary action	22
Matched request	Execute the specified valid goal and do not repeat it	22
Absent object	Reject the nonexistent target while respecting other valid goals	22
Cancellation during action	Finish the current indivisible block, then stop dispatching	22
Cancellation while idle	Hold until new authorization	22

A request about an absent object forbids acting on that object but does not cancel a valid goal the robot found on its own. For problems that appear during monitoring, we record separately when the event is applied and when it first becomes visible. An episode lasts 480 control steps, or 544 control steps in the condition where the problem appears during monitoring, and every system acts in blocks of 8 steps.

B.14.3 Cancellation and event timing

All systems that take language share the same cancellation channel. In cancellation-during-action episodes, the cancellation is sent when the system starts acting. A cancellation takes priority over ordinary requests, clears the current request and any pending task, and stops new actions once the current block ends. Only a resume command or a newly authorized task lets the robot act again; clearing the request alone does not, and the cancellation stays in effect without further messages. We record when each human and environment event was planned, when it arrived, and when it took effect.

B.14.4 Physical obligations and scoring

Each episode has a set of physical obligations with time windows, such as holding before an event and resolving the problem after it. An episode succeeds only when every target condition holds and no protected state is disturbed; a correct final state does not undo an earlier violation. The request-free success rate covers all valid episodes of the three request-free conditions, including clean scenes, and problem resolution and clean holding are also reported separately (Table 20). Outcomes after a human request are reported per condition (Table 21). The scorer uses the event timeline, the dispatched decisions, and the simulator state at every step, and distinguishes success, valid non-completion, invalid scenes or infrastructure failures, and timeouts. Unparsable decisions and unfinished tasks count against the policy, and timeouts stay in the denominator as failures. Each reported rate comes with its number of applicable episodes.

B.14.5 Pairing, seeds, and appearance transfer

Results average decision seeds 17, 23, and 41 with equal weight. Within a seed, each rate weights the three task families (faucet, cabinet door, and drawer, with 3, 3, and 5 scene resets) equally, and weights the resets within a family and the two executor seeds equally. Resets are paired across methods. A baseline that does not depend on the decision seed runs each scene once, and that run serves all three seeds. The evaluation covers 8 layout–style combinations and 11 scene resets per method, condition, and seed, each run twice with two executor random seeds (17 and 23), and paired 95% intervals come from 10,000 bootstrap resamples over layout–style combinations.

Table 20: **Autonomous behavior without a human instruction.** Rates in %; episode count is per applicable seed. Success averages the three no-request conditions; resolve covers the initial-problem and monitoring conditions, and clean hold the clean scenes.

System	Episodes	Success	Resolve	Clean hold
TWILL with SHUTTLE	66	81.48	75.56	93.33
TWILL with Direct	66	81.48	75.56	93.33
VA	66	6.67	0.00	20.00
Mission-prompted VLA	66	0.00	0.00	0.00
Matched VLA	66	33.33	0.00	100.00
Instruction-driven VLA	66	33.33	0.00	100.00
VLM commander (Qwen3-VL-8B)	66	50.00	30.56	88.89
Oracle $\rightarrow$ executor	44	–	88.89	–

Table 21: **Behavior after human intervention** (%). 22 episodes per condition and seed. Absent-object success also requires the other valid goals to be met. All systems use the same cancellation channel, and cancellation during action is sent when a system first starts acting. VA and the mission-prompted VLA take no human requests and are not listed.

System	Matched request	Absent object	Cancel active	Cancel idle
TWILL with SHUTTLE	88.15	42.22	100.00	100.00
TWILL with Direct	96.30	21.48	100.00	100.00
Instruction-driven VLA	88.89	5.56	100.00	100.00
Matched VLA	11.11	0.00	22.22	100.00
VLM commander (Qwen3-VL-8B)	60.00	0.00	60.00	100.00