

## A APPENDIX

### A.1 LLM USAGE

We used ChatGPT (GPT-5 Thinking) solely as a general-purpose writing assistant to refine prose after complete, author-written drafts were produced. Its role was limited to language editing—suggesting alternative phrasings, improving clarity and flow, and reducing redundancy—without introducing new citations or technical claims. The research idea, methodology, experiments, analyses, figures, and all substantive content were conceived and executed by the authors. LLMs were not used for ideation, data analysis, or result generation. All AI-assisted text was reviewed, verified, and, when necessary, rewritten by the authors, who take full responsibility for the manuscript’s accuracy and originality.

### A.2 EXTENDED RELATED WORK

**VLMs for RL.** Foundation models (Wiggins & Tejani, 2022) have proven broadly useful across downstream applications (Ramesh et al., 2022; Khandelwal et al., 2022; Chowdhury et al., 2025), motivating their incorporation into reinforcement learning pipelines. Early work showed that language models can act as reward generators in purely textual settings (Kwon et al., 2023), but extending this idea to visuomotor control is nontrivial because reward specification is often ambiguous or brittle. A natural remedy is to leverage visual reasoning to infer progress toward a goal directly from observations (Mahmoudieh et al., 2022; Rocamonde et al., 2023; Adeniji et al., 2023). One approach (Wang et al., 2024) queries a VLM to compare state images and judge improvement along a task trajectory; another aligns trajectory frames with language descriptions or demonstration captions and uses the resulting similarities as dense rewards (Fu et al., 2024; Rocamonde et al., 2023). However, empirical studies indicate that such contrastive alignment introduces noise, and its reliability depends strongly on how the task is specified in language (Sontakke et al., 2023; Nam et al., 2023).

**Natural Language in Embodied AI.** With VLM architectures pushing this multimodal interface forward (Liu et al., 2023; Karamcheti et al., 2024; Laurençon et al., 2024), a growing body of work integrates visual and linguistic inputs directly into large language models to drive embodied behavior, spanning navigation (Fried et al., 2018; Wang et al., 2019; Majumdar et al., 2020), manipulation (Lynch & Sermanet, 2020a;b), and mixed settings (Suglia et al., 2021; Fu et al., 2019; Hill et al., 2020). Beyond end-to-end conditioning, many systems focus on interpreting natural-language goals (Lynch & Sermanet, 2020b; Nair et al., 2022; Shridhar et al., 2022; Lynch et al., 2023) or on prompting strategies that extract executable guidance from an LLM—by matching generated text to admissible skills (Huang et al., 2022b), closing the loop with visual feedback (Huang et al., 2022c), planning over maps or graphs (Shah et al., 2023; Huang et al., 2022a), incorporating affordance priors (Ahn et al., 2022), explaining observations (Wang et al., 2023b), learning world models for prospective reasoning (Nottingham et al., 2023; Zellers et al., 2021), or emitting programs and structured action plans (Liang et al., 2022; Singh et al., 2022). Socratic Models (Zeng et al., 2022) exemplify this trend by coordinating multiple foundation models (e.g., GPT-3 (Brown et al., 2020) and ViLD (Gu et al., 2021)) under a language interface to manipulate objects in simulation. Conversely, our framework uses natural language not as a direct policy or planner, but as structured, episodic feedback that supports causal credit assignment in robotic manipulation.

**Failure Reasoning in Embodied AI.** Diagnosing and responding to failure has a long history in robotics (Ye et al., 2019; Khanna et al., 2023), yet many contemporary systems reduce the problem to success classification using off-the-shelf VLMs or LLMs (Ma et al., 2022; Ha et al., 2023; Wang et al., 2023a; Duan et al., 2024; Dai et al., 2025), with some works instruction-tuning the vision–language backbone to better flag errors (Du et al., 2023). Because large models can hallucinate or over-generalize, several studies probe or exploit model uncertainty to temper false positives (Zheng et al., 2024); nevertheless, the resulting detectors typically produce binary outcomes and provide little insight into *why* an execution failed. Iterative self-improvement pipelines offer textual critiques or intermediate feedback—via self-refinement (Madaan et al., 2023), learned critics that comment within a trajectory (Paul et al., 2023), or reflection over prior rollouts (Shinn et al., 2023)—but these methods are largely evaluated in text-world settings that mirror embodied environments such as ALFWorld (Shridhar et al., 2020), where perception and low-level control are abstracted away. In contrast, our approach targets visual robotic manipulation and treats language as struc-

tured, episodic *explanations* of failure that can be aligned with image embeddings and converted into temporally grounded reward shaping signals.

### A.3 EXPERIMENTAL SETUP

All experiments (including ablations) were run on a Linux workstation running Ubuntu 24.04.2 LTS (kernel 6.14.0-29-generic). The machine is equipped with an Intel Core Ultra 9 285K CPU, 96 GB of system RAM, and an NVIDIA GeForce RTX 4090 (AD102, 24 GB VRAM) serving as the primary accelerator; an integrated Arrow Lake-U graphics adapter is present but unused for training. Storage is provided by a 2 TB NVMe SSD (MSI M570 Pro). The NVIDIA proprietary driver was used for the RTX 4090, and all training/evaluation leveraged GPU acceleration; results reported in the paper were averaged over multiple random seeds with identical software and driver configurations on this host.

### A.4 META-WORLD MT10

We evaluated LAGEA on the MetaWorld (Yu et al., 2020) MT-10 benchmark. Meta-World MT10 is a widely used benchmark for multi-task robotic manipulation, comprising ten goal-conditioned environments drawn from the broader Meta-World suite (Yu et al., 2020). All tasks are executed with a Sawyer robotic arm under a unified control interface: a  $4D$  continuous action space (three Cartesian end-effector motions plus a gripper command) and a fixed  $39D$  observation vector that encodes the end-effector, object, and goal states. Episodes are capped at 500 steps and share a common reward protocol across tasks, enabling a single policy to be trained and evaluated in a consistent manner.

Figure 7 depicts the ten tasks, and Table 4 lists the corresponding natural-language instructions that ground each goal succinctly. The suite spans fine motor skills (e.g., button pressing, peg insertion) as well as larger object interactions (e.g., reaching, opening/closing articulated objects), making MT10 a demanding testbed for generalization and multi-task policy learning.

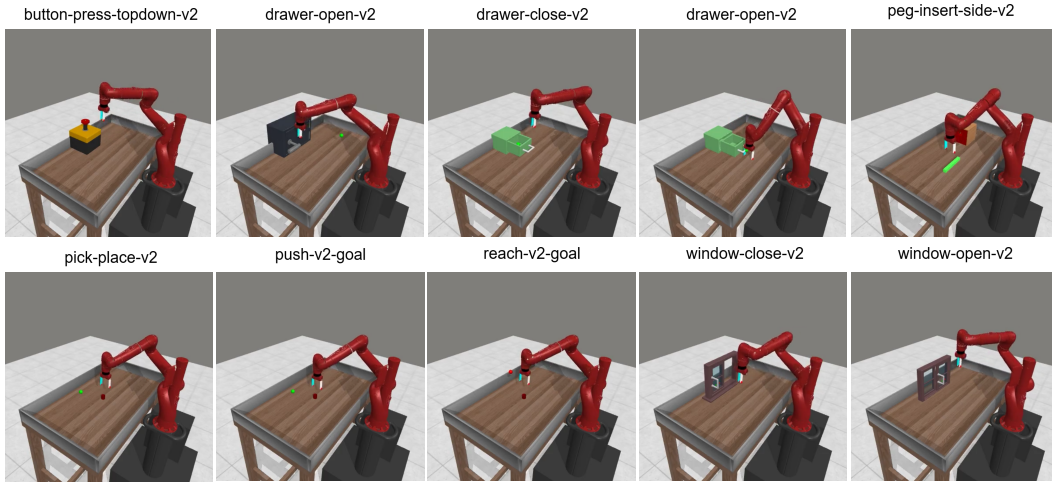


Figure 7: Meta-world MT10 benchmark tasks.

Table 4: Environments and their text instructions of Meta-world MT10 benchmark tasks.

Environment	Text instruction
button-press-topdown-v2	Press a button from the top.
door-open-v2	Open a door with a revolving joint.
drawer-close-v2	Push and close a drawer.
drawer-open-v2	Open a drawer.
peg-insert-side-v2	Insert the peg into the side hole.
pick-place-v2	Pick up the puck and place it at the target.
push-v2	Push the puck to the target position.
reach-v2	Reach a goal position.
window-close-v2	Push and close a window.
window-open-v2	Push and open a window.

#### A.5 IMPLEMENTATION DETAILS

In our experiments, we use the latest Meta-World M10 (Yu et al., 2020) environment. The main software versions are as follows:

- Python 3.11
- jax 0.4.16
- jaxlib 0.4.16+cuda12.cudnn89
- flax 0.7.4
- gymnasium 0.29.1
- gymnasium-robotics 1.2.4
- mujoco 2.3.7
- optax 0.2.1
- torch 2.2.1
- torchvision 0.17.1
- numpy 1.26.4
- imageio 2.34.0
- matplotlib 3.8.3

#### A.6 ALGORITHM

The pseudocode algorithm 1 formalizes the LAGEA training loop. Each episode, the policy collects a trajectory with RGB observations and a task instruction; we select a small set of key frames and query an instruction-tuned VLM (Qwen-2.5-VL-3B) (Bai et al., 2025) to produce a structured reflection (error code, key-frame indices, brief rationale). The instruction and reflection are encoded with a lightweight GPT-2 text encoder and paired with visual embeddings; a projection head is trained with a keyframe-gated alignment objective followed by a symmetric, weighted contrastive loss so that feedback becomes control-relevant. At training time we compute two potentials from these aligned embeddings: one that measures instruction–state goal agreement and one that measures transition consistency with the VLM diagnosis around the cited frames. We use only the change in these signals between successive states as a per-step shaping reward, add it to the environment reward with adaptive scaling and simple agreement gating (emphasizing failure episodes early and annealing over time), and update a standard SAC (Haarnoja et al., 2018) agent from a replay buffer with target networks.

**Algorithm 1:** LAGEA: Feedback–Grounded Reward Shaping (lean)

---

**Input** : Encoders  $\Phi_I, \Phi_T, \Phi_F$ ; VLM  $\mathcal{Q}$ ; goal image  $o_g$ ; instruction  $y$ ; replay buffer  $\mathcal{D}$ ; episodes  $N$

**Output**: trained policy  $\pi$

1 **Initialize**: projection heads  $E_i, E_t, E_f$ ; policy  $\pi$ ; SAC learner.

2  $z_g \leftarrow \text{norm}(E_i(\Phi_I(o_g)))$ ,  $z_y \leftarrow \text{norm}(E_t(\Phi_T(y)))$ .

3 **for**  $i = 1$  to  $N$  **do**

4   /\* Collect Trajectories Figure 1 \*/

5   Roll out  $\pi$  to obtain  $\{(o_t, r_t^{\text{task}})\}_{t=0}^{T-1}$ ; push to  $\mathcal{D}$ .

6   /\*Key frames & per-step weights Section 3.1.2\*/

7    $x_t \leftarrow \Phi_I(o_t)$ ;  $s_t \leftarrow \langle \text{norm}(E_i(x_t)), z_g \rangle$ ;

8    $\mathcal{K} \leftarrow \text{GETKEYFRAMES}(s_{0:T-1}, M)$ ;  $\hat{w} \leftarrow \text{TRIANGULARWEIGHTS}(\mathcal{K}, h)$  (unit mean).

9   /\*Structured episodic reflection Section 3.1.1\*/

10   Subsample  $N$  frames; query  $\mathcal{Q}$  with frames; encode feedback  $z_f \leftarrow \text{norm}(E_f(\Phi_F(f)))$

11   /\*Feedback alignment Section 3.1.3\*/

12    $\text{UPDATEFEEDBACKALIGNMENT}(E_i, E_f; \mathcal{D}, \hat{w})$ ;

13    $\text{UPDATEFEEDBACKCONTRASTIVEWEIGHTED}(E_i, E_f; \mathcal{D}, \hat{w})$ .

14   /\*Dense Reward shaping Section 3.2\*/

15   **for**  $t = 0$  to  $T - 2$  **do**

16      $z_t \leftarrow \text{norm}(E_i(x_t))$ ,  $z_{t+1} \leftarrow \text{norm}(E_i(x_{t+1}))$ ;

17     Calculate goal delta;  $r_t^{\text{goal}} \leftarrow \text{GOALDELTA}(z_t, z_{t+1}; z_y, z_g)$ ;

18     Calculate feedback delta;  $r_t^{\text{fb}} \leftarrow \text{FEEDBACKDELTA}(z_t, z_{t+1}; z_f)$ ;

19      $\alpha \leftarrow \text{CLIP}(\alpha_{\text{base}} \cdot \frac{1 + \langle z_y, z_f \rangle}{2}, [\alpha_{\text{min}}, \alpha_{\text{max}}])$ ;

20     Calculate fused dense reward;  $\tilde{r}_t \leftarrow (1 - \alpha) r_t^{\text{goal}} + \alpha \hat{w}_t r_t^{\text{fb}}$ .

21   /\*Adaptive reward shaping Section 3.2\*/

22    $\rho_t \leftarrow \text{ADAPTIVERHO}(\text{progress EMA / schedule})$ ;

23   Overall reward;  $r_t \leftarrow r_t^{\text{task}} + \rho_t \tilde{r}_t$ .

24   /\*Update SAC\*/

25    $\text{UPDATESAC}(\pi; \mathcal{D}, r_t)$ .

---

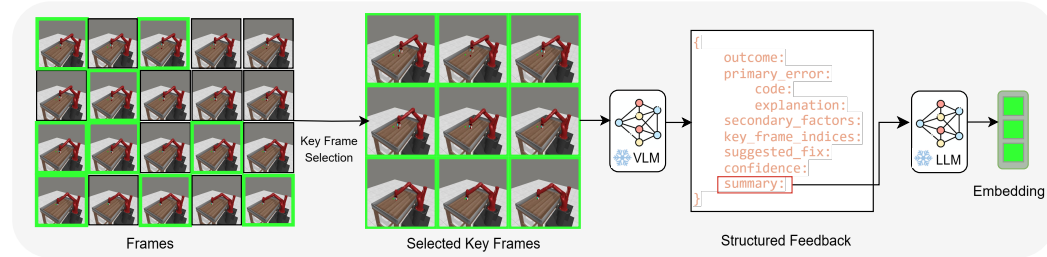


Figure 8: **Feedback Generation Pipeline:** Keyframes are selected from an episode, analyzed by a VLM to produce structured feedback text, which is then encoded into a final feedback embedding.

## A.7 FEEDBACK PIPELINE

At the end of each episode, we run a deterministic key-frame selector over the image sequence to extract a compact set of causal moments  $\mathcal{K}$ . We then assemble a prompt with the task instruction, a compact error taxonomy, few-shot exemplars, and the selected frames, and query a frozen VLM (Qwen-2.5-VL-3B). The model is required to return a schema-constrained JSON with fields outcome, primary\_error{code, explanation}, secondary\_factors, key\_frame\_indices, suggested\_fix, confidence, and summary. Responses are validated against the schema and retried on violations. Textual slots are normalized and embedded with a lightweight GPT-2 encoder to produce a feedback vector  $f$  that is time-anchored via  $\mathcal{K}$ .

This structured protocol reduces hallucination, yields feedback comparable across episodes and viewpoints, and makes the language signal embeddings directly consumable by the alignment and reward-shaping modules.

#### A.8 HYPERPARAMETERS

Hyperparameters for LAGEA are illustrated in Table 5. We followed the hyperparameter and relay steps parameters provided in (Fu et al., 2024). Many of these hyperparameters are not tuned to perfection; therefore, tuning them could achieve slightly better performance.

Table 5: Summarization of hyper-parameters.

Parameter	Value
Camera Id	2
Residual connection	False
Total environment steps	$1 \times 10^6$
Start timesteps	10,000
Decay timesteps	$7.5 \times 10^5$
Evaluation episodes	100
Evaluation frequency	1e6/100
Checkpoint frequency	1e6/10
Adam learning rate	$1 \times 10^{-4}$
Batch size	256
Discount factor $\gamma$	0.99
Target network $\tau$	0.01
Hidden dimensions	(256, 256)
Initializer	Orthogonal
Random seed	0
Relay threshold	2500
Exploration noise	0.2
VLM reward weight $\rho$	0.25
$\rho$ cap	1.0
Gap	10
Crop	False
L2 margin	0.25
Cosine margin	0.25
Embed buffer size	20,000
Feedback coefficient $\alpha_{\text{feedback}}$	0.5
Episode success threshold	100
Contrastive start step	25,000
Contrastive temperature $\tau$	0.07
Label smoothing	0.05
$\lambda_{\text{align}}$	0.02
$\lambda_{\text{uniform}}$	$1 \times 10^{-3}$
Shaping target ratio	0.20
Shaping warmup steps	20,000
Shaping anneal end step	600,000
Shaping scale cap	10.0
Shaping scale floor	0.1
Shaping clip per step	1.0
Shaping EMA $\beta$	0.1
Shape only on fail	True
Use goal delta	True
Success EMA $\beta$	0.01
Progress power	2.0
$\alpha_{\text{min}}$	0.20
$\alpha_{\text{max}}$	0.95

#### A.9 ERROR TAXONOMY

An error taxonomy is introduced to systematically characterize the types of failures observed in robot manipulation trajectories. This taxonomy provides discrete error codes that capture common failure modes in manipulation tasks, such as interacting with the wrong object, approaching from an incorrect direction, failing to establish a stable grasp, applying insufficient force, or drifting away from the intended goal. By mapping trajectories to these interpretable categories, we enable

Table 6: Error codes and their descriptions.

Error Code	Description
wrong_object	Interacted with the wrong object.
bad_approach_direction	Approached object from a wrong angle/direction.
failed_grasp	Contact without a stable grasp; slipped or never closed gripper appropriately.
insufficient_force	Touched correct object but did not exert proper motion/force.
drift_from_goal	Trajectories drifted away from the goal, no course correction.

```

{
  task: {string},
  outcome: {success | failure},
  primary_error: {
    code: {error_code or success_code},
    explanation: {one sentence explanation}
  },
  secondary_factors: [{error_code, ...}],
  key_frame_indices: [{int, int, int}],
  suggested_fix: {string or (n/a)},
  confidence: {float in [0,1]},
  summary: {one sentence summary}
}

```

Figure 9: Schema for structured feedback returned by the VLM

structured analysis of failure cases and facilitate targeted improvements in policy learning. Table 6 summarizes the error codes and their descriptions.

#### A.10 STRUCTURED FEEDBACK

Structured feedback mechanism constrains the VLM to produce precise, interpretable, and reproducible outputs. After each rollout, the model returns a JSON object that follows the schema shown in Figure 9, rather than free-form text. The schema records the task identifier, the binary outcome (**success** or **failure**), a single primary error code with a short explanation, optional secondary factors, key frames, a suggested fix, a confidence score, and a concise summary. This format anchors feedback to concrete evidence, keeps annotations consistent across episodes, and makes the signals directly usable for downstream analysis.

Example structured feedback is shown for two Meta-World tasks - button-press-topdown-v2 and door-open-v2 - with two success cases in Figures 10 and Figure 11 and two failure cases in Figures 12 and Figure 13.

For the success cases, the schema assigns `primary_error.code=good_grasp`, with empty `secondary_factors`, high confidence, and `suggested_fix=(n/a)`. In button-press-topdown-v2, success is attributed to a secure grasp followed by a vertical, normal-aligned press that achieves the goal. In door-open-v2, success is similarly tied to a stable grasp on the handle and the application of sufficient force to open the door.

In the failure counterparts, the same schema yields concise, actionable diagnoses. For button-press-topdown-v2, `primary_error.code=bad_approach_direction` reflects a lateral approach that causes sliding; the prescribed fix is a topdown, normalaligned press. For door-open-v2, `primary_error.code=failed_grasp` with `insufficient_force` as a secondary factor attributes failure to unstable closure and inadequate actuation; the recommended remedy is a tighter grasp and sufficient force. Across both tasks, explanations remain succinct and suggested fixes translate diagnosis into concrete adjustments, ensuring comparability and evidential grounding within the structured format.

```

{
  task: button-press-topdown-v2-goal-observable,
  outcome: success,
  primary_error: {
    code: good_grasp,
    explanation: The gripper successfully grasped the
button.
  },
  secondary_factors: [ ],
  key_frame_indices: [12, 18],
  suggested_fix: (n/a),
  confidence: 0.9,
  summary: The agent succeeded because it grasped the
button securely and pressed it straight down, achieving the
goal.
}

```

Figure 10: Success case with structured feedback for *button-press-topdown-v2-goal-observable* task.

```

{
  task: door-open-v2-goal-observable,
  outcome: success,
  primary_error: {
    code: good_grasp,
    explanation: The gripper successfully grasped the black
block and opened its door.
  },
  secondary_factors: [ ],
  key_frame_indices: [9, 18, 27],
  suggested_fix: (n/a),
  confidence: 0.9,
  summary: The robot successfully opened the door of the
black block by grasping it and applying the appropriate
force.
}

```

Figure 11: Success case with structured feedback for *door-open-v2-goal-observable* task.

#### A.11 ABLATION

To quantify the contribution of each component in LAGEA, we run controlled ablations with identical training settings, three random seeds per task, and we report mean (std.) success. All variants use the same encoders, SAC learner, and goal image; unless noted otherwise. The protocol followed for the ablation study is as follows:

**Feedback Alignment** Drop the multi-stage feedback—vision alignment and rely on frozen encoder similarities; tests whether learned alignment is required to obtain a control-relevant embedding geometry.

**Feedback Quality Ablation** Replace the schema-constrained (structured) feedback with unconstrained free-form VLM feedback text; measures the impact of feedback structure, reliability and hallucination on reward stability.

**Keep all, drop adaptive  $\rho$**  Use the full shaping signals but fix the mixing weight instead of scheduling it; probes the role of progress-aware scaling for stable learning.

```

{
  task: button-press-topdown-v2-goal-observable,
  outcome: failure,
  primary_error: {
    code: bad_approach_direction,
    explanation: The gripper came from the side, sliding
off the button instead of a vertical press.
  },
  secondary_factors: [ ],
  key_frame_indices: [18, 22],
  suggested_fix: Approach from directly above the button;
align gripper normal to the button surface, then press
straight down.,
  confidence: 0.85,
  summary: The robot failed to press the button correctly
because it approached from the side instead of a vertical
press. This resulted in the gripper sliding off the button.
}

```

Figure 12: Failure case with structured feedback for *button-press-topdown-v2-goal-observable* task.

```

{
  task: door-open-v2-goal-observable,
  outcome: failure,
  primary_error: {
    code: failed_grasp,
    explanation: The gripper did not close properly around
the door handle, leading to a failed attempt to open the
door.
  },
  secondary_factors: [insufficient_force],
  key_frame_indices: [16, 24],
  suggested_fix: Ensure the gripper closes tightly around
the door handle and applies sufficient force.,
  confidence: 0.9,
  summary: The agent failed to open the door as the gripper
did not close properly around the handle, indicating a failed
grasp.
}

```

Figure 13: Failure case with structured feedback for *door-open-v2-goal-observable* task.

**Drop all, keep adaptive  $\rho$**  Remove goal-/feedback-delta terms and keyframe gating while retaining the adaptive schedule (no auxiliary signal added); controls for the possibility that the schedule alone yields gains.

**Key frame ablation** Replace keyframe localization with uniform per-step weights; assesses the value of temporally focused credit assignment around causal moments.

**Delta reward ablation** Use absolute similarities instead of temporal deltas; tests whether potential-based differencing (which avoids static-state bias) is essential.

Table 7: Ablation results of LAGEA. Experiments were done using three different seeds. Results are averaged here.

Task	Feedback Alignment	Feedback Quality Ablation	Keep all, drop adaptive $\rho$	Drop all, keep adaptive $\rho$	Key frame ablation	Delta reward ablation
button-press-topdown-v2-observable	20 (34.64)	10 (10)	13.33(23.09)	33.33(57.74)	30 (51.96)	30 (51.96)
drawer-open-v2-observable	100 (0)	96.67(5.77)	100 (0)	0 (0)	76.67(40.41)	100 (0)
door-open-v2-observable	100 (0)	100 (0)	100 (0)	0 (0)	100 (0)	76.67(40.41)
push-v2-hidden	100 (0)	66.67(57.74)	66.67(57.74)	33.33(57.74)	100 (0)	100 (0)
drawer-open-v2-hidden	100 (0)	100 (0)	100 (0)	33.33(57.74)	100 (0)	66.67(57.74)
door-open-v2-hidden	100 (0)	100 (0)	100 (0)	33.33(57.74)	100 (0)	100 (0)

#### A.12 SUCCESSFUL TRAJECTORY VISUALIZATION

Figure 14 presents successful trajectory visualizations generated by LAGEA across nine environments from Meta-World MT10. Each trajectory illustrates how LAGEA effectively completes the corresponding manipulation task, highlighting its generalization ability across diverse settings. The only exception is `peg-insert-side-v2`, where LAGEA was unable to produce a successful episode; therefore, no trajectory is shown for this environment.

#### A.13 LIMITATIONS

LAGEA still inherits occasional hallucinations from the underlying VLM, which our structure and alignment mitigate but cannot eliminate. While the study spans diverse simulated tasks, real-robot generalization and long-horizon observability remain open challenges. A natural next step is to translate from simulation to real-robot deployment, closing the sim-to-real gap.

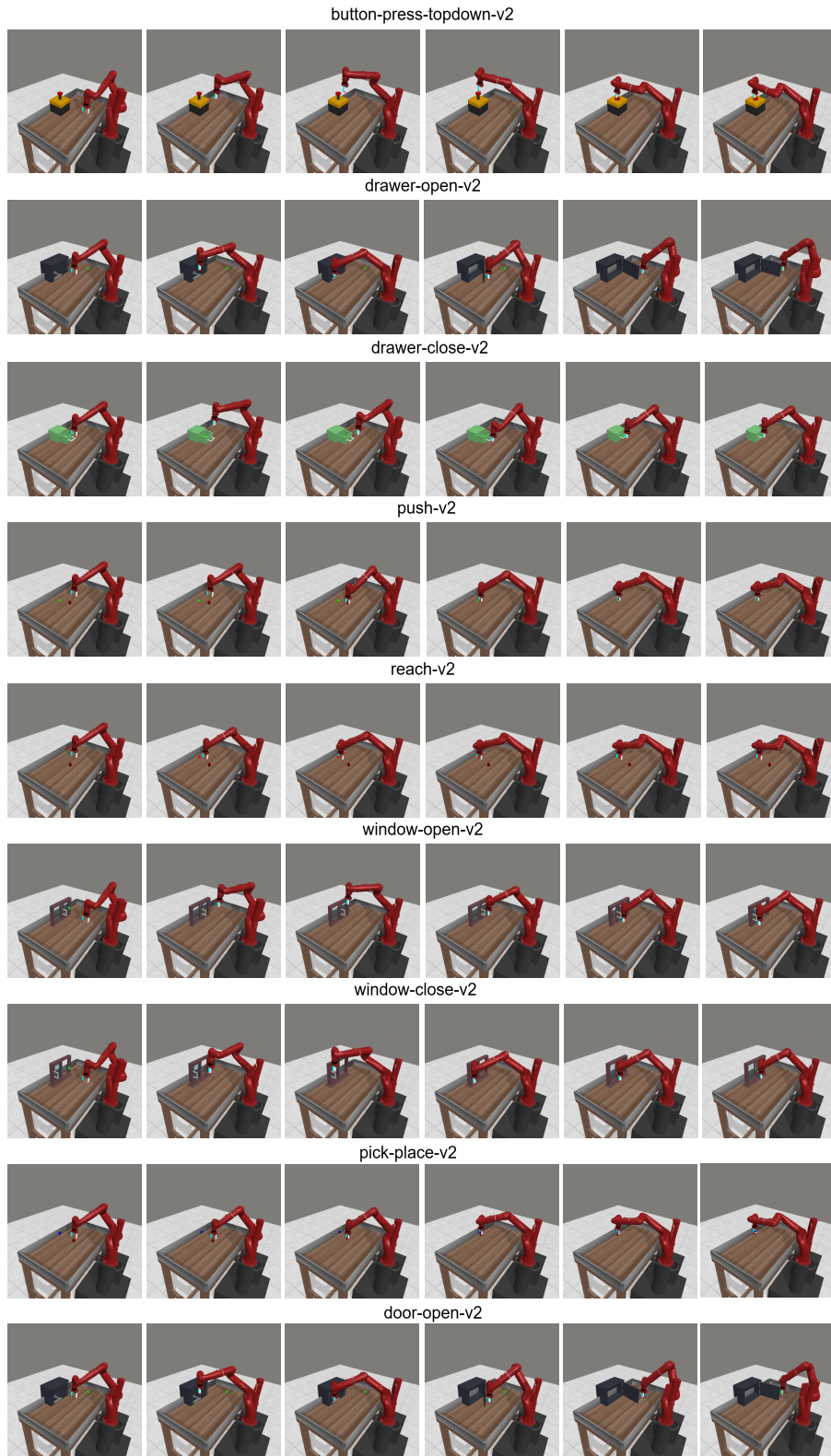


Figure 14: Visualization of successful trajectories using LAGEA on environments from Meta-World MT10 benchmark tasks.