ROVER: Recursive Reasoning Over Videos with Vision-Language Models for Embodied Tasks

Philip Schroeder[1] Ondrej Biza[2] Thomas Weng[2] Hongyin Luo[1] James Glass[1]

[1]MIT CSAIL [2]Robotics & AI Institute

pschro@mit.edu

Abstract

Vision-language models (VLMs) have exhibited impressive capabilities across diverse image understanding tasks, but still struggle in settings that require reasoning over extended sequences of camera frames from a video. This limits their utility in embodied settings, which require reasoning over long frame sequences from a continuous stream of visual input at each moment of a task attempt. To address this limitation, we propose ROVER (Reasoning Over VidEo Recursively), a framework that enables the model to recursively decompose long-horizon video trajectories into segments corresponding to shorter subtasks within the trajectory. In doing so, ROVER facilitates more focused and accurate reasoning over temporally localized frame sequences without losing global context. We evaluate ROVER, implemented using an in-context learning approach, on diverse OpenX Embodiment videos and on a new dataset derived from RoboCasa that consists of 543 videos showing both expert and perturbed non-expert trajectories across 27 robotic manipulation tasks. ROVER outperforms strong baselines across three video reasoning tasks: task progress estimation, frame-level natural language reasoning, and video question answering. We observe that, by reducing the number of frames the model reasons over at each timestep, ROVER mitigates hallucinations, especially during unexpected or non-optimal moments of a trajectory. In addition, by enabling the implementation of a subtask-specific sliding context window, ROVER's time complexity scales linearly with video length, an asymptotic improvement over baselines.
Demos, code, and data available at: https://rover-vlm.github.io/anon/

1 Introduction

Vision-language models (VLMs) have demonstrated remarkable generalization capabilities across a wide range of tasks, including image captioning, visual question answering, and grounding language in complex visual scenes [3, 22, 52]. These models excel at extracting high-level semantic understanding from visual inputs and can generate fluent natural language that reflects this understanding [18, 56]. As a result, VLMs have become a key component in efforts to build general-purpose AI systems.

Given their success, VLMs are increasingly viewed as promising foundations for embodied intelligence: systems that perceive, reason about, and act within the physical world [38, 9, 19, 15, 31, 57, 25, 54, 63]. Embodied tasks require VLMs to recognize and reason about what is happening at each moment in time (given a continuous stream of visual input from camera video) when attempting to complete a task. Existing frameworks for VLM reasoning in this setting either perform highly localized reasoning over a small set of frames [57, 25, 54, 63] or attempt to reason over the entire sequence by concatenating all video frames into a single context [31]. Both strategies have limitations: the former sacrifices global context, while the latter is computationally expensive and overwhelms the model with irrelevant or redundant visual input. In this work, we seek to advance VLM capabilities in embodied settings by enabling high-precision frame-by-frame reasoning, without compromising

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

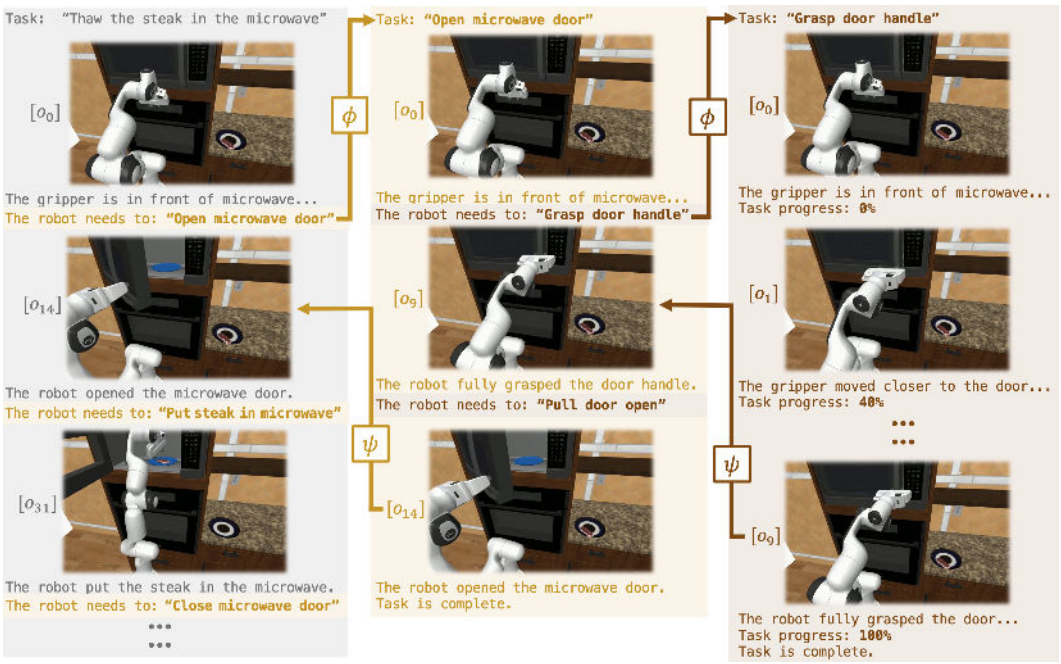

Figure 1: ROVER is a recursive framework for reasoning over camera video that decomposes a task into subtasks to maintain a compact temporal context, improving reasoning accuracy and efficiency.

efficiency or losing global context for the full task trajectory. With this motivation, we propose **R**easoning **O**ver **V**id**E**o **R**ecursively (ROVER).

ROVER is a framework that enables the model to recursively decompose the task corresponding to the given video input, reducing the number of camera frames the VLM must reason over at any given moment during the trajectory (Figure 1). Instead of generating a long, single line of reasoning spanning all timesteps of the video input for a task attempt (e.g., opening a door), ROVER decomposes the task and generates a separate line of reasoning for each subtask (e.g., grasping the door handle). When a subtask is complete, the corresponding line of reasoning terminates and a new one is created for the video input for the next subtask (e.g., pulling the door open). We show that the decomposition not only improves accuracy by focusing the reasoning on relevant temporal segments, but also enables the implementation of a subtask-specific sliding context window, which further reduces the number of frames the model must reason over at each moment of a trajectory.

We evaluate ROVER, implemented using an in-context learning approach, in the setting of robotic manipulation tasks using a large-scale dataset of videos collected from robot-mounted and third-person camera viewpoints during both successful and unsuccessful task attempts. We create this dataset by automatically perturbing expert demonstrations collected in RoboCasa [39] to produce diverse trajectories, ranging from near-optimal to fully random action sequences. In addition, we compute ground-truth task progress estimates at each timestep based on geometric distance to goal states. The generated dataset includes 543 videos across 27 tasks, each collected in a random kitchen scene. We leverage this dataset to evaluate ROVER across three benchmarks for embodied reasoning over camera video: 1) frame-level task progress estimation, 2) frame-level natural language reasoning, and 3) video question answering (QA). We also test ROVER on previous benchmarks based on diverse real-world OpenX Embodiment videos [41].

ROVER outperforms baselines across all settings: it achieves a stronger correlation with ground-truth task progress, lower frame-level reasoning error rates, and higher video QA accuracy. Specifically, we observe that ROVER reduces model hallucinations, especially during moments of a trajectory that deviate from expected or optimal behavior. We show these types of hallucinations during non-optimal states are most likely to occur when the model is reasoning over long sequences of camera frames. ROVER mitigates these hallucinations by reducing the number of frames the model must reason over at each timestep. In addition, by leveraging a sliding window within each subtask, ROVER's time complexity scales linearly with video length, an asymptotic improvement over baselines.

Overall, the contributions of this work including the following:

- We propose ROVER, a recursive framework that improves video reasoning accuracy and efficiency in embodied settings.
- We introduce a protocol to generate diverse non-expert trajectories with ground-truth progress labels from expert demonstrations in simulation.
- We release a large-scale video dataset featuring both expert and non-expert robotic task executions, annotated with natural language descriptions and fine-grained task progress signals.
- We establish a video reasoning benchmark that evaluates models on frame-level progress estimation, frame-level natural language reasoning, and video QA across diverse task trajectories.

2 Problem setup

We model tasks as goal-conditioned finite-horizon partially observed Markov decision processes:

$$\mathcal{M} := \langle S, A, P, R, \Omega, O, T, G \rangle. \tag{1}$$

The tuple $\langle S, A, P, R, G \rangle$ forms a Markov Decision Process, where S is the (unobserved) state space, A is the action space, $P : S \times A \to \Delta(S)$ is the state-transition function and $R : S \times G \to \mathbb{R}$ is the reward function. An agent observes the world through $O : S \to \Delta(\Omega)$, the observation function, where Ω is the set of observations. Finally, T is the time horizon and G specifies the task semantically. Conditioned on a history $h_t = (a_{<t}, o_{\leq t}) \in H$ and a goal $g \in G$, we aim to find a policy $\pi : H \times G \to A$ that maximizes the γ-discounted return $J_\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t, g) \right]$. The goal-conditioned value function V^π is defined as the expected cumulative reward when starting from an initial state s_0 and following policy π:

$$V^\pi(s_0; g) = \mathbb{E}_{\pi, P} \left[\sum_{t=0}^{T-1} \gamma^t R(s_t, a_t, g) \,\middle|\, s_0 \right].$$

Given a task description $g \in G$ and sequence of observations $\{o_t\}_{t=1}^T$ from video input, our goal is to generate text, $\{m_t\}_{t=1}^T$, following each observation that reasons in natural language about the task trajectory relative to the goal g, including estimating task progress, at every timestep. Section 3 outlines the recursive framework we propose for this reasoning task. To evaluate reasoning accuracy, we generate a diverse dataset of videos, $\mathcal{D}$, exhibiting task attempts with varying levels of expertise by inserting random perturbations within a given dataset, $\mathcal{D}_{src}$, of expert trajectories in simulation (described in section 4.1). In addition, we compute ground-truth task progress estimates at each timestep of the generated videos based on geometric distance to goal states (described in section 4.2).

3 ROVER framework for reasoning over video trajectories

Given a task description g and visual input Ω from camera video, ROVER recursively generates sequences of reasoning at each timestep of the task attempt. The sequences of reasoning are comprised of camera frames for each timestep, interwoven with natural language text (Figure 1). ROVER can be implemented using the recursive function shown in Algorithm 1. The function takes two inputs: the context c for the given task, and the sequence of tokens, Y, that have been provided or generated so far for this task. Y includes text tokens generated by the VLM and image tokens representing the frames from the camera input. We treat a token sequence (e.g., c and Y) as a list of tokens.

ROVER begins with $Y = [\,]$ and c includes text tokens of the initial task description (e.g., "Thaw the steak in the microwave") and the first observation (image tokens of the camera frame). The model generates, $\Theta(c + Y)$, one token at a time conditioning on the given context, c, appended with the growing token sequence, Y.

If Y includes the tokens for retrieving the next observation (details explained below), then the image tokens for the camera frame at the next

Algorithm 1 ROVER

```
function ROVER(c, Y)
    while True do
        Y = Y + Θ(c + Y)
        if Y progresses to next frame then
            Y = Y + o_next
        else if Y specifies a new task then
            Y = Y + ψ( ROVER(φ(Y), [] ) )
        else if Y indicates task complete then
            return Y
```

timestep, o_{next}, are appended to Y. If Y specifies a new subtask, then a new line of reasoning is created with ROVER($\phi(Y), [\,]$) where Y for the child process is initialized as an empty list and c is based on the token sequence of the parent, $\phi(Y)$. When the child process ends, its output tokens are appended to the parent's token sequence and the parent continues generating. If Y indicates the task is complete, or the end of the video is reached, then the process returns Y (its full token sequence).

Implementation of ϕ and ψ functions. The ϕ function defines the context for a child process based on the full token sequence of the parent at the time the child is spawned, including tokens directly generated by the parent or returned as output from previous child processes of that parent. The ψ function defines the output tokens of a child based on its full token sequence at the time it terminates. In our implementation of ROVER (as depicted in Figure 1), ϕ returns the new task description provided by the parent along with the last frame of the parent line of reasoning. We implement ψ as a function that returns the final frame, and corresponding final text description, of the child process.

Special text tokens for retrieving next frame and specifying new subtasks. When the VLM is done generating the natural language text tokens for reasoning about the current camera frame, it either moves on to the frame for the next timestep or specifies a new subtask. The next frame of the video can be retrieved by generating the text tokens "[next-frame]". A new subtask can be specified by generating the text tokens "The robot needs to: {new_subtask}", as depicted in Figure 1.

Sliding context window for video. In our implementation of ROVER, we find that using a sliding window at each timestep not only improves inference efficiency, but also mitigates observed problems with hallucination. When appending a new frame, $Y = \mathrm{Window}(Y + o_{next})$, the window extracts three frames: the first frame of the current line of reasoning and the most recent previous frame (each paired with their previously generated text descriptions), along with the newly added next frame. We observe a complementary effect between the sliding window and the recursive decomposition. The window forces the model to focus on the most relevant frames for the current subtask and the current timestep, while the recursive specification of new subtasks ensures the current subtask description and first frame of the subtask-specific window are continually updated as task milestones are achieved during the video. Further, by capping the maximum number of frames included in the context to three, the inference time complexity of ROVER scales linearly with the total video frame count. This provides an asymptotic improvement over baselines, which scale quadratically with total frame count.

4 Generating diverse video trajectories and ground-truth value estimates

To test vision-language reasoning across diverse video trajectories, we seek to use a source dataset of expert demonstrations, $\mathcal{D}_{src}$, for a task in simulation to build a dataset of trajectories, $\mathcal{D}$, exhibiting a wide spectrum of expertise with that task, along with value estimates for all states. We focus on robot manipulation tasks and, following prior work [36, 11], we assume tasks consist of a known sequence of object-centric subtasks. If we let $E = \{e_1, e_2, ...\}$ be the set of entities (objects) in a task $\mathcal{M}$, we assume that tasks consist of a sequence of object-centric subtask trajectories $\tau = \{\tau_i(e_{\tau_i})\}_{i=1}^{M}$, where the manipulation in each subtask trajectory $\tau_i(e_{\tau_i})$ is relative to a single object's coordinate frame ($e_{\tau_i} \in E$). We include details regarding how this sequence is specified for each task in Appendix M.

4.1 Non-expert trajectory generation

In this section, we outline an approach which, given a set of expert demonstrations, $\mathcal{D}_{src}$, generates a dataset of trajectories, $\mathcal{D}$, exhibiting varying levels of expertise, ranging from nearly expert to fully random action sequences. Each expert demonstration, $\tau^E \in D_{src}$, is decomposed into object-centric segments $\tau^E = \{\tau_i^E(e_{\tau_i})\}_{i=1}^{M}$ (see Appendix M for full details). Following MimicGen [36], each subtask trajectory $\tau_i^E = \{(s_t^E, a_t^E)\}_{t=1}^{T_i^E}$ corresponds to a manipulation relative to a single object e_{τ_i}.

Non-expert trajectories are derived by injecting random deviations into these expert demonstrations (Figure 2). For a given subtask trajectory τ_i^E, we construct a corresponding non-expert version $\tau_i^N = \{(s_t^N, a_t^N)\}_{t=1}^{T_i^N}$ by inserting random actions at selected moments $q \in \mathcal{Q}_i \subseteq \{1, \ldots, T_i^E\}$. Each time in $\mathcal{Q}_i$ marks a branching point, at which τ_i^N diverges from τ_i^E for some number of steps, w. The index of each step in the deviation is specified based on the index of the expert state, q, at which the deviation began, along with the number of steps, j (ranging from 0 to $w-1$), it has deviated from that expert state. For each step during the deviation $\{q, j\}_{j=0}^{w-1}$, a randomly sampled

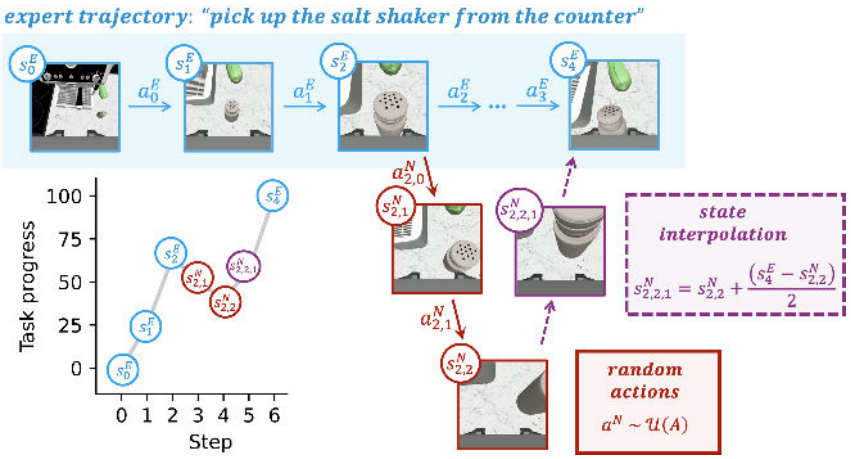

Figure 2: Generating diverse video trajectories by inserting random deviations during expert path.

action $a^N_{q,j} \sim \mathcal{U}(A)$ is executed, where $\mathcal{U}(A)$ denotes the uniform distribution over the action space. The deviation leads to a sequence of states and actions $\{s_{q,j}, a_{q,j}\}_{j=0}^{w-1}$ that diverges from the expert trajectory, where $s_{q,0}$ is the expert state from which the deviation began. Following deviation, the trajectory can either terminate or start to recover to the expert trajectory. Recovery is implemented by interpolating between the deviated state and a downstream expert state (Figure 2). Specifically, given the final deviated state $s^N_{q,w}$ and a selected future expert state s^E_{q+h} (for some $h > 0$), we generate a sequence of interpolated states:

$$s^N_{q,w,z} = (1 - \alpha_z) \cdot s^N_{q,w} + \alpha_z \cdot s^E_{q+h}, \quad \alpha_z = \frac{z}{n_{\text{interp}}}, \quad z = 1, \ldots, n_{\text{interp}},$$

to gradually transition back to the expert trajectory. The index of each step during recovery is specified based on the index of the expert state, q, at which the original deviation began, the deviation length, w, and the number of interpolation steps, z (from 1 to n_{interp}) that have occurred. n_{interp} determines the smoothness of the recovery. This procedure results in trajectories that deviate to varying extents from the expert trajectory. We ensure diversity across trajectories by varying deviation timesteps, number and magnitude of deviations (including nested deviations), and recovery points.

4.2 Value functions

For each task, we define a ground-truth value function to evaluate the accuracy of VLM reasoning throughout the video trajectories. In robotics, value functions can be difficult to define due to task heterogeneity. We follow prior work [14, 29, 31, 45, 46, 53] in modeling value using task progress, which provides a normalized temporal measure of how far an agent has advanced toward completing a task. Formally, we define a progress-based value function, $V : \mathcal{S} \times G \to [0, 1]$, mapping an observation and goal specification to a scalar value between 0 and 1.

Goal-focused distance measures. To quantify progress, we define subtask-specific metrics based on spatial relationships in the observable state. For each subtask, we identify key geometric distances that reflect meaningful aspects of completion and are continuous and monotonic with respect to progress. These include the sum of distances between the contact points of the robot end effector, $\{r_1, r_2, \cdots, r_C\}$, and the corresponding contact points of the target object, $\{l_1, l_2, \cdots, l_C\}$, in the scene, denoted $y^{r,e}$, along with the distance between the current position of the target object and its respective goal position, denoted $y^{e,f}_t$,

$$y^{r,e}_t = \sum_{j=1}^{C} \left\| p^{r_j}_t - p^{l_j}_t \right\|_2, \quad y^{e,f}_t = \left\| p^{e_{\tau_i}}_t - p^{e_{\tau_i}}_f \right\|_2$$

where e_{τ_i} is the subtask-specific target object discussed above, $p^{r_j}_t$ and $p^{l_j}_t$ are positions of the j^{th} robot and object contact point, respectively, at time t, $p^{e_{\tau_i}}_t$ is the position of e_{τ_i} at time t, and $p^{e_{\tau_i}}_f$ is the target position of e_{τ_i}. All distances are computed using the L2 norm (Euclidean distance). The environment includes B entities, each of which is represented by a position in three dimensional

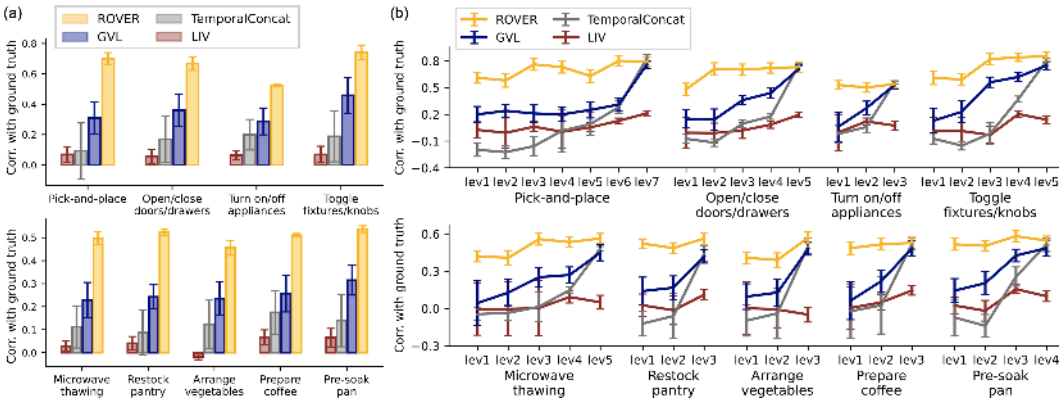

Figure 3: Mean and standard error of correlation between ground-truth values and progress values predicted for all videos (a) and stratified across trajectory level (b). Highest-level videos for each task show near-expert completion. Amount of non-expert behavior increases as level decreases.

Cartesian space within a global coordinate frame. The total goal-focused distance, $y = \{y_t\}_{t=1}^{T_i}$, is the sum of these metrics, $y_t = (1-\beta)\, y_t^{r,e} + \beta\, y_t^{e,f}$, where β is a scalar with range $[0, 1]$ that is manually tuned for each object-centric subtask to ensure the degree to which $y^{r,e}$ and $y^{e,f}$ contribute to y is proportional to how meaningful each component is to the overall subtask completion.

Trajectory-focused distance measures. The goal-focused metrics are necessary to track what we value in the task (e.g., moving the gripper closer to the target object, the target object moving closer to the goal position). However, they may fail to capture important nuances in real-world trajectories. For instance, the shortest Euclidean path between the robot gripper and target objects may intersect with obstacles that must be avoided. To address this, we refine the value function by leveraging known expert states within each trajectory. Within the expert demonstrations, we identify a sparse set of keypoints, $\kappa = \{s_{k_i}^E\}_{i=1}^{n}$, that form the local maxima in the goal-focused distance, $y = \{y_t\}_{t=1}^{T}$, for a given object-centric subtask. Since these keypoints derive from the expert demonstrations, we know that, despite forming local maxima in y, they represent intermediate states that lie along high-value trajectories. For each timestep in a non-expert trajectory, we identify the nearest downstream expert keypoint and use this as an auxiliary signal to adjust the local value estimate.

$$u_t = \|s_{t,j,z}^N - s_{k_{\text{next}}}^E\|_2, \quad k_{\text{next}} = \min\{k_i \in \kappa \mid k_i > t\}, \quad \text{where } j = z = 0 \text{ when } t \notin \mathcal{Q}_i$$

where $s_{t,j,z}^N$ is the environment state at the current step of the non-expert trajectory and $s_{k_{\text{next}}}^E \in \mathbb{R}^{3\times B}$ is the full environment state at time k_{next} during the expert trajectory.

Final value estimates. We transform the raw distance measures for the full subtask trajectory into normalized progress values, $v = \{v_t\}_{t=1}^{T_i}$, between 0 and 1 by inverting and rescaling them:

$$v_t = \frac{-d_t + \max(d)}{-\min(d) + \max(d)}, \quad d_t = y_t + u_t, \quad d = \{d_t\}_{t=1}^{T_i}$$

To form the value estimates for the full trajectory, $\tau = \{\tau_i(e_{\tau_i})\}_{i=1}^{M}$, we sequence together the value estimates corresponding to each subtask, adding the value estimates of each subtask to the final value of the previous subtask and, when complete, rescaling the list of values to the range $[0, 1]$.

5 Experiments

We conduct large-scale experiments to evaluate ROVER across diverse tasks for embodied reasoning over camera video. The baseline methods include the state-of-the-art value model LIV [32], a multi-modal contrastive model [43] fine-tuned with a value learning objective. We also include the best existing in-context learning approach, Generative Value Learning (GVL) [31], and an ablated version of GVL, called TemporalConcat, that processes frames in temporal order. Similar to GVL, we implement ROVER using an in-context learning approach, with the same prompting used for all backbone VLMs (details in Appendix D). We test ROVER using various backbone models. The main results are based on Gemini-1.5-Pro. Appendix J includes results for Gemini-2.5-Pro-Preview,

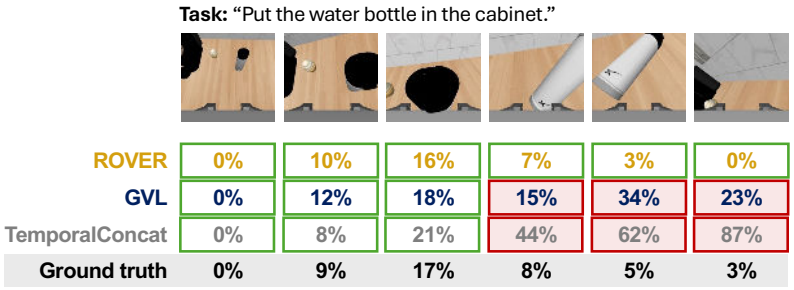

Figure 4: ROVER exhibits more accurate reasoning and progress prediction during non-expert states.

GPT-4o, and Qwen-2.5-VL-32B-Instruct. Examples showcasing ROVER video reasoning, along with anonymous links to all code and data, are provided at: `https://rover-vlm.github.io/anon/`

We use RoboCasa [39] as the simulation environment and leverage the demonstrations they provide (collected via human teleoperation) as the source dataset of expert trajectories (details in Appendix E). Our generated evaluation dataset, comprising trajectories that exhibit a wide range of task expertise, includes 543 videos across 27 tasks (Appendix B), each collected in a random kitchen scene. The videos are separated into levels based on the amount of the task completed during the video (Appendix B). The highest-level videos in each task group show full task completion with near-expert behavior.

5.1 Vision-language reasoning tasks with generated video dataset

Our simulation-generated video trajectories, each annotated with dense value estimates, enable the evaluation of multiple vision-language reasoning tasks that test fine-grained temporal understanding and visual grounding. We separate the evaluation into the following three tasks: 1) frame-level task progress estimation, 2) frame-level natural language reasoning, and 3) video question answering (QA). Further details regarding the task design and performance metrics are provided in Appendix C.

5.2 Task 1: Frame-level task progress estimation

This benchmark involves predicting task progress at each timestep of a video (see Appendix C.1 for details). Following prior work [31], we implement this by having the model predict integer values representing the percentage progress achieved relative to the first frame of the video, which is always assigned a progress value of zero. We implement ROVER such that the model predicts task progress values up to 100 for each subtask, as depicted in Figure 1. To form progress values for the full task horizon, we divide the values within each subtask by the total number of subtasks and add them to the value for the last timestep of the previous subtask (see Appendix D for details).

ROVER shows higher correlation with ground-truth task progress estimates. For videos that exhibit task completion with near-expert behavior (i.e., the highest level within each task group), ROVER, GVL, and TemporalConcat achieve a Pearson correlation near or above 0.5 (Figure 3) and an L2 distance below 200 (Appendix A.1), relative to the ground-truth value estimates, across most task groups. However, for videos with incomplete task execution, GVL and TemporalConcat deviate significantly from the ground truth (as depicted in Figure 4). These deviations become more extreme as the trajectory level decreases (i.e., as the number of non-expert states in the video increases).

ROVER achieves higher correlation with frame number on real-world expert demonstrations. In Appendix K, we include the same value estimation analysis conducted in Ma et al. [31] using 1,000 videos from 50 real-world datasets from OXE [41]. We see that ROVER achieves higher correlation with ground-truth value estimates (which are based on frame number) across the OXE datasets, most of which contain human-collected expert demonstrations.

5.3 Task 2: Frame-level natural language reasoning

At each frame of the video, ROVER, along with baselines, generates a description of the frame before predicting task progress for that timestep. We cross-reference these descriptions with information about the ground-truth simulator state at each timestep to calculate the reasoning error rate and success rate (described in Appendix C.2). We treat this evaluation as a language model task, where we prompt the model with the generated description paired with the ground-truth information and ask if the description includes any statements that are verifiably true or false relative to the ground truth.

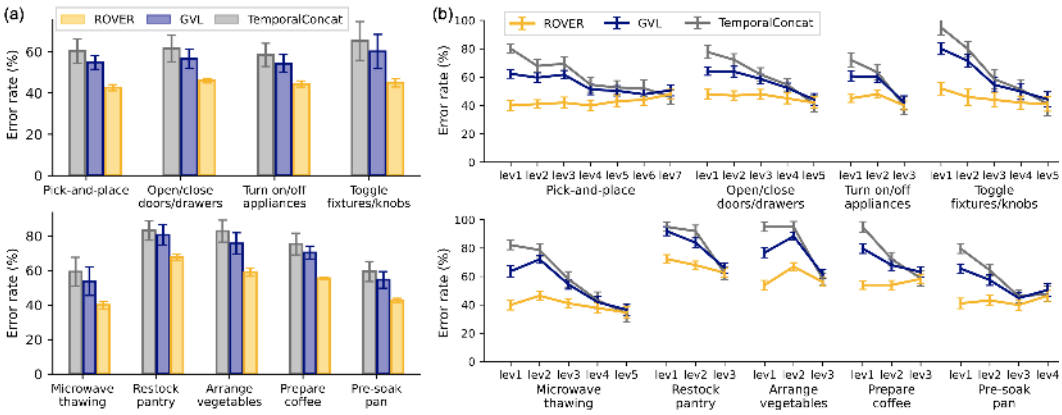

Figure 5: Mean and standard error of reasoning error rate (percentage of frames model states something that is verifiably wrong) for all videos (a) and stratified across trajectory level (b).

ROVER achieves lower error rate in frame-level reasoning during video. The reasoning error rate is similar for all methods for videos containing near-expert task completion (Figure 5). However, as videos deviates from expert behavior, the error rate increases significantly for GVL and TemporalConcat. We observe a similar pattern for reasoning success rate (Appendix A.2). These trends mirror the results of the progress prediction task, suggesting that errors in progress prediction result from errors in the natural language descriptions of the frame that precedes the progress prediction at each timestep. We further evaluate the nature of these errors in the video QA analysis below.

ROVER reduces errors by focusing context. When stratifying results across state type (expert, non-expert, and interpolating states) and context length in Appendix F.2, we see that GVL and TemporalConcat are most likely to error when encountering non-expert states and when the number of frames included in the context extends beyond 10. We observe a similar pattern when stratifying the progress prediction results across state type (Appendix F.1). ROVER reduces these errors by using recursive decomposition, along with a sliding window, to narrow the context at each timestep.

5.4 Task 3: Video QA

The video QA task includes questions about whether specific events occur (and the time they occur) during the video (details in Appendix C.3). We generate answers to each question for this task using the frame-level descriptions produced by ROVER, GVL, and TemporalConcat. For each question, we use a language model to generate an answer given all frame descriptions concatenated together. In doing so, the QA performance provides a secondary measure of the quality of the frame-level natural language reasoning of each method.

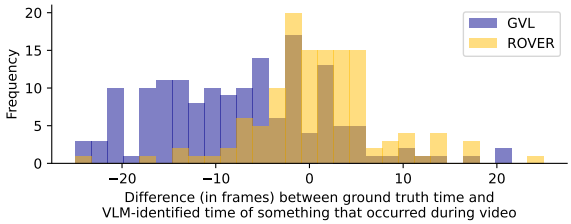

Figure 6: Difference in time (in frames) between actual time and VLM-identified time of events that occurred in video. One second in real time corresponds to ~3 frames. Negative frame difference indicates VLM states something occurs before it actually occurs during video.

ROVER reduces hallucinations during videos exhibiting non-expert behavior. ROVER shows significantly higher accuracy than GVL and TemporalConcat on the video QA benchmark across all task groups (Appendix A.3). The video QA results reveal a hallucination problem with GVL and TemporalConcat, where the VLM is likely to state that an event occurred during a video, regardless of whether it actually occurs. This is illustrated by the low precision (20 to 50%) and high recall values (near 100%) across all task groups for GVL and TemporalConcat (Appendix A.3). The hallucination problem is also highlighted in the analysis of the distance between the time when something occurs in a video and the time when the VLM states that it occurred (Figure 6). We see that, even when GVL correctly states something occurs during a video, it is much more likely than ROVER to state this prematurely (as shown by the negative frame difference in Figure 6).

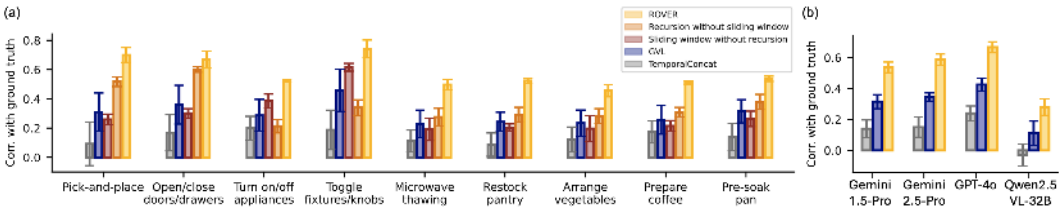

Figure 7: Mean and standard error of correlation between ground-truth value estimates and progress values predicted across all videos for (a) design ablations and (b) changing backbone VLM.

5.5 Ablations

We ablate ROVER to demonstrate how different algorithmic design decisions impact the performance gains we observe relative to baselines. Further, we show that ROVER is robust to video length and frame rate (Appendix H), camera view (Appendix I), and backbone VLM (Figure 7b, Appendix J).

We test two ablations of ROVER (details in Appendix L). One version ablates only the sliding window while the other ablates only the recursive reasoning. Consistent with prior work [31], we observe that baseline methods such as TemporalConcat often hallucinate steady, monotonically increasing task progress values, even when the video includes significant stalls or regressions. GVL mitigates this problem via random frame shuffling, which disrupts the strong temporal priors and forces the model to assess each frame more carefully [31]. We observe that the sliding window (even without recursion) similarly reduces hallucination by retaining temporal context while avoiding overly rigid sequential processing. Our ablations show that the window-only variant of ROVER significantly outperforms TemporalConcat and matches or exceeds GVL's performance on short-horizon tasks, such as turning on/off appliances (Figure 7a). We see that adding the recursive decomposition further improves performance for tasks consisting of numerous subtasks, such as pick-and-place tasks.

6 Related work

There is significant prior work leveraging VLMs in embodied reasoning settings [38, 9, 19, 15, 32]. Methods such as [31, 57, 25, 54, 63] have applied VLMs for reasoning over video sequences, either by attending to a small number of keyframes or processing the entire trajectory. However, these approaches struggle to balance fine-grained reasoning with the preservation of global task context. Separately, decomposition-based reasoning has been explored in both language and vision-language domains [24, 55, 16, 28, 37, 40, 64]. Recent work has also explored structured prompting and recursive task breaking in VLMs [21, 61, 42, 44], though primarily in static or text-centric settings. ROVER builds on these ideas by introducing a recursive, subtask-oriented video-reasoning framework to address challenges of continuous visual input in embodied tasks, allowing for more precise and scalable temporal reasoning without sacrificing global context. **More discussion in Appendix O.**

7 Limitations

ROVER improves reasoning over videos by recursively decomposing the given task into subtasks and, in doing so, maintaining a compact temporal context that mitigates hallucination. However, when decomposition fails (e.g., by introducing unnecessary or invalid subtasks), the resulting reasoning may become fragmented or misaligned with the actual task progression. In addition, we implement ROVER with an in-context learning approach. We leave to future work the evaluation of fine-tuning methods that can improve ROVER's performance with frame-level reasoning and value estimation.

8 Broader impact

In this work, we show how ROVER improves VLM reasoning in embodied settings. VLMs are being increasingly deployed to autonomously interact with external environments. By improving this capacity, our work has the potential for amplifying risk associated with automated decision-making. Addressing these risks requires careful consideration of ethical guidelines and robustness checks.

9 Conclusion

We present ROVER, a recursive framework that improves VLM reasoning over video sequences in embodied tasks. By decomposing complex tasks into subtasks and enabling localized, temporally bounded reasoning within each, ROVER effectively reduces cognitive load on the model and mitigates common failure modes such as hallucination during off-nominal behavior. Our experiments across three benchmarks (progress estimation, frame-level reasoning, and video QA), along with previous benchmarks based on diverse real-world OXE videos, demonstrate consistent improvements over strong baselines, particularly in handling diverse, suboptimal trajectories. In addition, by enabling the use of a subtask-specific sliding window, ROVER's time complexity scales linearly with video length, an asymptotic improvement over baselines. By combining architectural insights with a new large-scale dataset and evaluation protocol, our work lays a foundation for more robust and scalable embodied reasoning systems grounded in real-world video input.

References

[1] Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Fu, C., Gopalakrishnan, K., Hausman, K., et al. (2022). Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

[2] Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48.

[3] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al. (2023). Qwen technical report. *arXiv preprint arXiv:2309.16609*.

[4] Baumli, K., Baveja, S., Behbahani, F., Chan, H., Comanici, G., Flennerhag, S., Gazeau, M., Holsheimer, K., Horgan, D., Laskin, M., et al. (2023). Vision-language models as a source of rewards. *arXiv preprint arXiv:2312.09187*.

[5] Bhateja, C., Guo, D., Ghosh, D., Singh, A., Tomar, M., Vuong, Q., Chebotar, Y., Levine, S., and Kumar, A. (2023). Robotic offline rl from internet videos via value-function pre-training. *arXiv preprint arXiv:2309.13041*.

[6] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. (2023). Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

[7] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al. (2022). Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*.

[8] Chen, A. S., Nair, S., and Finn, C. (2021). Learning generalizable robotic reward functions from "in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*.

[9] Chen, B., Xu, Z., Kirmani, S., Ichter, B., Sadigh, D., Guibas, L., and Xia, F. (2024). Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465.

[10] Dasari, S., Ebert, F., Tian, S., Nair, S., Bucher, B., Schmeckpeper, K., Singh, S., Levine, S., and Finn, C. (2019). Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*.

[11] Di Palo, N. and Johns, E. (2022). Learning multi-stage tasks with one demonstration via self-replay. In *Conference on Robot Learning*, pages 1180–1189. PMLR.

[12] Ding, Y., Zhang, X., Paxton, C., and Zhang, S. (2023). Task and motion planning with large language models for object rearrangement. *arXiv preprint arXiv:2303.06247*.

[13] Ebert, F., Yang, Y., Schmeckpeper, K., Bucher, B., Georgakis, G., Daniilidis, K., Finn, C., and Levine, S. (2021). Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*.

[14] Eysenbach, B., Salakhutdinov, R., and Levine, S. (2020). C-learning: Learning to achieve goals via recursive classification. *arXiv preprint arXiv:2011.08909*.

[15] Gao, J., Sarkar, B., Xia, F., Xiao, T., Wu, J., Ichter, B., Majumdar, A., and Sadigh, D. (2024). Physically grounded vision-language models for robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12462–12469. IEEE.

[16] Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. (2023). Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

[17] Gupta, N., Lin, K., Roth, D., Singh, S., and Gardner, M. (2019). Neural module networks for reasoning over text. In *International Conference on Learning Representations*.

[18] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

[19] Hong, Y., Zhen, H., Chen, P., Zheng, S., Du, Y., Chen, Z., and Gan, C. (2023). 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494.

[20] Huang, S., Jiang, Z., Dong, H., Qiao, Y., Gao, P., and Li, H. (2023a). Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*.

[21] Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. (2023b). Inner monologue: Embodied reasoning through planning with language models. In *Conference on Robot Learning*, pages 1769–1782. PMLR.

[22] Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. (2024). Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

[23] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., et al. (2018). Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673. PMLR.

[24] Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., and Sabharwal, A. (2023). Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

[25] Kim, C., Seo, S.-W., and Kim, S.-W. (2025). Lamour: Leveraging language models for out-of-distribution recovery in reinforcement learning. *arXiv preprint arXiv:2503.17125*.

[26] Klissarov, M., D'Oro, P., Sodhani, S., Raileanu, R., Bacon, P.-L., Vincent, P., Zhang, A., and Henaff, M. (2023). Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*.

[27] Kwon, T., Di Palo, N., and Johns, E. (2023). Language models as zero-shot trajectory generators. *arXiv preprint arXiv:2310.11604*.

[28] Lee, S. and Kim, G. (2023). Recursion of thought: A divide-and-conquer approach to multi-context reasoning with language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 623–658.

[29] Lee, Y., Szot, A., Sun, S.-H., and Lim, J. J. (2021). Generalizable imitation learning from observation via inferring goal proximity. *Advances in Neural Information Processing Systems*, 34:16118–16130.

[30] Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. (2023). Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE.

[31] Ma, Y. J., Hejna, J., Fu, C., Shah, D., Liang, J., Xu, Z., Kirmani, S., Xu, P., Driess, D., Xiao, T., et al. (2024). Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations*.

[32] Ma, Y. J., Kumar, V., Zhang, A., Bastani, O., and Jayaraman, D. (2023a). Liv: Language-image representations and rewards for robotic control. In *International Conference on Machine Learning*, pages 23301–23320. PMLR.

[33] Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. (2023b). Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*.

[34] Ma, Y. J., Sodhani, S., Jayaraman, D., Bastani, O., Kumar, V., and Zhang, A. (2022). Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*.

[35] Mahmoudieh, P., Pathak, D., and Darrell, T. (2022). Zero-shot reward specification via grounded natural language. In *International Conference on Machine Learning*, pages 14743–14752. PMLR.

[36] Mandlekar, A., Nasiriany, S., Wen, B., Akinola, I., Narang, Y., Fan, L., Zhu, Y., and Fox, D. (2023). Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*.

[37] Murali, V., Qi, L., Chaudhuri, S., and Jermaine, C. (2018). Neural sketch learning for conditional program generation. In *International Conference on Learning Representations*.

[38] Nag, S., Zhu, X., Song, Y.-Z., and Xiang, T. (2022). Zero-shot temporal action detection via vision-language prompting. In *European Conference on Computer Vision*, pages 681–697. Springer.

[39] Nasiriany, S., Maddukuri, A., Zhang, L., Parikh, A., Lo, A., Joshi, A., Mandlekar, A., and Zhu, Y. (2024). Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*.

[40] Nye, M., Hewitt, L., Tenenbaum, J., and Solar-Lezama, A. (2019). Learning to infer program sketches. In *International Conference on Machine Learning*, pages 4861–4870. PMLR.

[41] O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., et al. (2024). Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE.

[42] Prasad, A., Koller, A., Hartmann, M., Clark, P., Sabharwal, A., Bansal, M., and Khot, T. (2023). Adapt: As-needed decomposition and planning with language models. *arXiv preprint arXiv:2311.05772*.

[43] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

[44] Schroeder, P., Morgan, N., Luo, H., and Glass, J. (2024). Thread: Thinking deeper with recursive spawning. *arXiv preprint arXiv:2405.17402*.

[45] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. (2018). Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE.

[46] Sermanet, P., Xu, K., and Levine, S. (2016). Unsupervised perceptual rewards for imitation learning. *arXiv:1612.06699*.

[47] Shafiullah, N. M., Rai, A., Etukuru, H., Liu, Y., Misra, I., Chintala, S., and Pinto, L. (2023). On bringing robots home. *arXiv preprint arXiv:2311.16098*.

[48] Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. (2023). Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE.

[49] Szot, A., Schwarzer, M., Agrawal, H., Mazoure, B., Talbott, W., Metcalf, K., Mackraz, N., Hjelm, D., and Toshev, A. (2023). Large language models as generalizable policies for embodied tasks. *arXiv preprint arXiv:2310.17722*.

[50] Talmor, A. and Berant, J. (2018). The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651.

[51] Tang, Y., Yu, W., Tan, J., Zen, H., Faust, A., and Harada, T. (2023). Saytap: Language to quadrupedal locomotion. *arXiv preprint arXiv:2306.07580*.

[52] Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., et al. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

[53] Tian, S., Nair, S., Ebert, F., Dasari, S., Eysenbach, B., Finn, C., and Levine, S. (2020). Model-based visual planning with self-supervised functional distances. *arXiv preprint arXiv:2012.15373*.

[54] Venkataraman, S., Wang, Y., Wang, Z., Erickson, Z., and Held, D. (2024). Real-world offline reinforcement learning from vision language model feedback. *arXiv preprint arXiv:2411.05273*.

[55] Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. (2023). Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634.

[56] Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. (2024a). Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[57] Wang, Y., Sun, Z., Zhang, J., Xian, Z., Biyik, E., Held, D., and Erickson, Z. (2024b). Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*.

[58] Wang, Y., Sun, Z., Zhang, J., Xian, Z., Biyik, E., Held, D., and Erickson, Z. (2024c). Rl-vlm-f: Reinforcement learning from vision language foundation model feedback. *arXiv preprint arXiv:2402.03681*.

[59] Wang, Y., Wu, Z., Yao, J., and Su, J. (2024d). Tdag: A multi-agent framework based on dynamic task decomposition and agent generation. *arXiv preprint arXiv:2402.10178*.

[60] Xie, T., Zhao, S., Wu, C. H., Liu, Y., Luo, Q., Zhong, V., Yang, Y., and Yu, T. (2023). Text2reward: Automated dense reward function generation for reinforcement learning. *arXiv preprint arXiv:2309.11489*.

[61] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

[62] Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Gonzalez Arenas, M., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., et al. (2023). Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*.

[63] Zeng, Y., Mu, Y., and Shao, L. (2024). Learning reward for robot skills using large language models via self-alignment. *arXiv preprint arXiv:2405.07162*.

[64] Zheng, W., Sharan, S., Jaiswal, A. K., Wang, K., Xi, Y., Xu, D., and Wang, Z. (2023). Outline, then details: Syntactically guided coarse-to-fine code generation. In *International Conference on Machine Learning*, pages 42403–42419. PMLR.

[65] Zhu, Y., Wang, Z., Merel, J., Rusu, A., Erez, T., Cabi, S., Tunyasuvunakool, S., Kramár, J., Hadsell, R., de Freitas, N., et al. (2018). Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*.

[66] Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Nasiriany, S., and Zhu, Y. (2020). robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*.

Appendix Contents

- A Additional results for main experiments
  - A.1 Task 1: Frame-level task progress estimation
  - A.2 Task 2: Frame-level natural language reasoning
  - A.3 Task 3: Video QA
- B Details on generated evaluation dataset
  - Table 1. Tasks within each task group.
  - Table 2. Description of levels for non-expert trajectories and video dataset counts.
- C Details on evaluation tasks
  - C.1 Task 1: Frame-level task progress estimation
  - C.2 Task 2: Frame-level natural language reasoning
  - C.3 Task 3: Video QA
- D Prompting and implementation details
  - D.1 GVL
  - D.2 ROVER
- E Simulation environment and dataset of expert demonstrations
- Results for additional experiments
  - F Stratifying results by state type and context length
  - H Robustness to video length and frame rate
  - I Robustness to camera view
  - J Testing across different backbone models
  - K Task progress prediction with real-world OpenX Embodiment datasets
  - L Ablations
- M Additional details on object-centric subtasks
- O Full related work

A Additional results for main experiments

A.1 Task 1: Frame-level task progress estimation

ROVER becomes less correlated with *frame number* as the number of non-expert states during video increases. To further explore how reasoning quality changes as videos deviate from the expert behavior, we evaluate how the correlation between the video frame number and the predicted task progress changes as the number of non-expert states in the video increases. This provides a measure of reasoning quality that is independent of the ground-truth value estimates. The progress values generated by GVL and TemporalConcat show similar correlation with frame number regardless of the number of non-expert states in the video (Figure 8). In contrast, ROVER shows a significant drop in its correlation with frame number as the number of non-expert states in the video increases, as would be expected with accurate reasoning (and as is observed with the ground-truth value estimates).

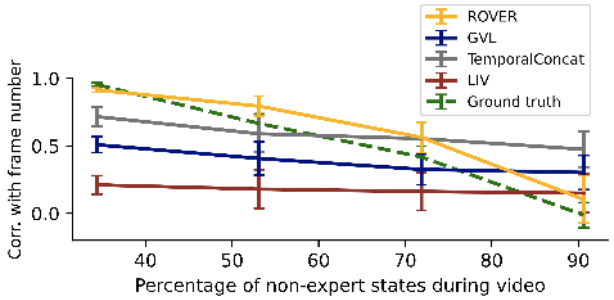

Figure 8: Mean and standard error of correlation between video frame number and the frame-level progress values predicted by each method stratified across proportion of non-expert states in video.

ROVER achieves smaller distance between predicted and ground-truth progress values with non-expert trajectories. Figure 9 is based on the same set of results presented in Figure 3, but uses L2 distance, instead of correlation, between model-predicted and ground-truth progress estimates as the performance metric. Similar to what we observe with the correlation results in Figure 3, we see in Figure 9 that ROVER significantly improves performance (i.e., better agreement with ground-truth value estimates reflected by smaller L2 distance) for videos exhibiting non-expert behavior. The performance improvement of ROVER over baselines becomes more significant as the trajectory level decreases (i.e., as the number of non-expert states in the video increases).

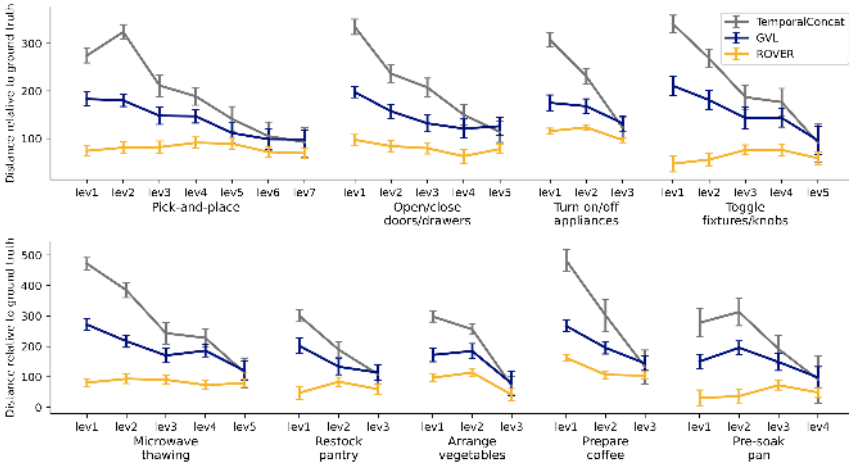

Figure 9: Mean and standard error of distance between model-predicted and ground-truth task progress estimates stratified across trajectory level.

A.2 Task 2: Frame-level natural language reasoning

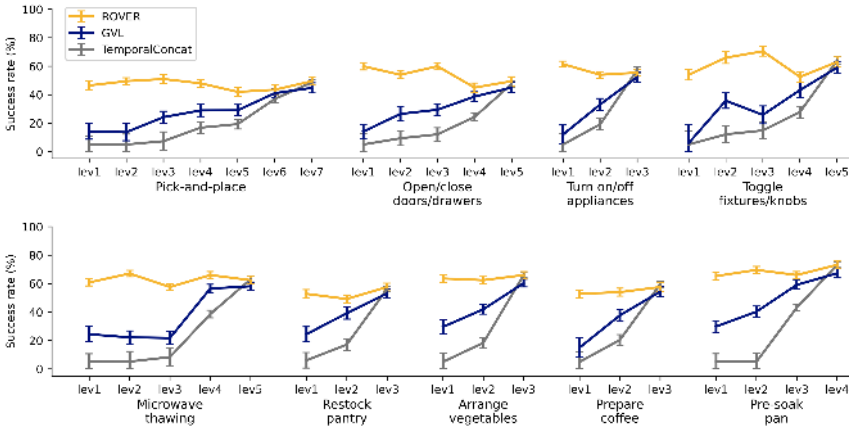

Figure 10: Mean and standard error of frame-level reasoning success rate (percentage of frames model states something that is verifiably correct relative to ground-truth information about simulator state) for videos stratified across trajectory level.

A.3 Task 3: Video QA

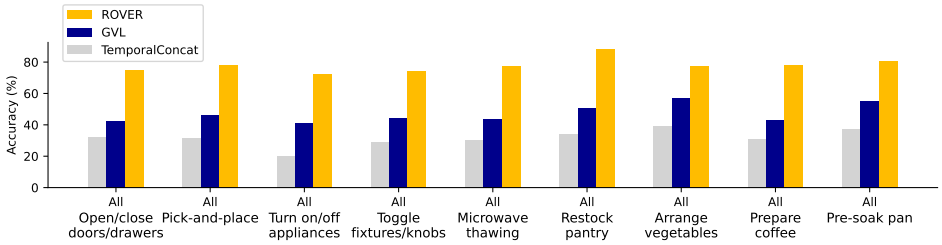

Figure 11: Video QA accuracy (the percentage of questions for which the model correctly identifies whether the event occurred) across all task groups.

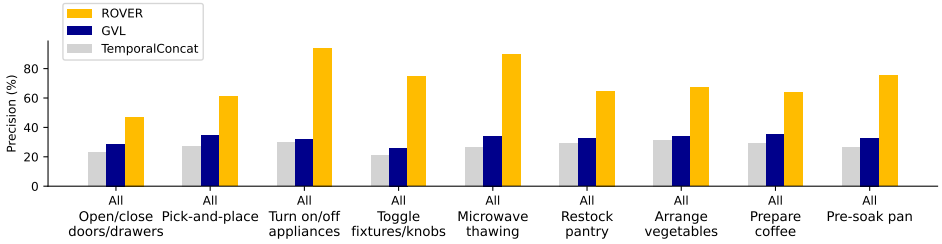

Figure 12: Video QA precision (the percentage of predicted occurrences that actually occurred in the video) across all task groups.

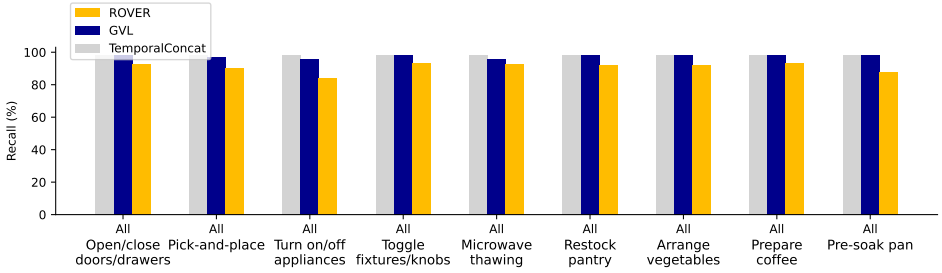

Figure 13: Video QA recall (the percentage of actual occurrences correctly identified by the model) across all task groups.

B Details on generated evaluation dataset

Our generated evaluation dataset comprises videos that exhibit a wide range of task expertise. The dataset includes 543 videos across 27 tasks, with about 20 videos per task (Tables 1 and 2). Each trajectory is collected in a random kitchen scene (random floor plan, random kitchen style, and random textures) in RoboCasa [39]. The videos are separated into levels based on the amount of the task completed during the video (Table 2). The highest-level videos in each task group show full task completion with near-expert behavior. The dataset contains both atomic and composite tasks from RoboCasa (descriptions of each task are provided in the appendix of Nasiriany et al. [39]). The composite tasks involve two to four times as many actions as the atomic tasks. For evaluation, we downsample videos to 30 frames for the atomic tasks (following Ma et al. [31]) and donwsample videos to 60 frames for the longer composite tasks.

Table 1: Tasks within each task group.

Task group	**Task**	**Task type**
Pick-and-place	PickPlaceCabToCounter PickPlaceCounterToCab PickPlaceCounterToMicrowave PickPlaceCounterToSink PickPlaceCounterToStove PickPlaceMicrowaveToCounter PickPlaceSinkToCounter PickPlaceStoveToCounter CoffeeSetupMug CoffeeServeMug	Atomic
Open/close doors/drawers	OpenSingleDoor CloseSingleDoor OpenDrawer CloseDrawer	Atomic
Turn on/off appliances	TurnOnMicrowave TurnOffMicrowave CoffeePressButton	Atomic
Toggle fixtures/knobs	TurnSinkSpout TurnOnSinkFaucet TurnOffSinkFaucet TurnOnStove TurnOffStove	Atomic
Microwave thawing	MicrowaveThawing	Composite
Restock pantry	RestockPantry	Composite
Arrange vegetables	ArrangeVegetables	Composite
Prepare coffee	PrepareCoffee	Composite
Pre-soak pan	PreSoakPan	Composite

Table 2: Levels for non-expert trajectories and video dataset counts. For each level, we assume that all task components corresponding to the preceding levels have already been successfully completed.

Task group	**Non-expert trajectory levels**	Dataset count
Pick-and-place	1: Fail to approach {obj}	3
	2: Approach {obj}	3
	3: Contact {obj}	3
	4: Pick up {obj}	3
	5: Keep grasp of {obj}	3
	6: Approach placing {obj} in {target_location}	3
	7: Place {obj} in {target_location}	3
Open/close doors/drawers	1: Fail to approach the door/drawer	4
	2: Approach door/drawer	4
	3: Contact door/drawer	4
	4: Start opening/closing door/drawer	4
	5: Finish opening/closing door/drawer	4
Turn on/off appliances	1: Fail to approach the on/off button/lever	6
	2: Approach the on/off button/lever	6
	3: Successfully adjust the on/off button/lever	6
Toggle fixtures/knobs	1: Fail to approach the lever/knob	4
	2: Approach lever/knob	4
	3: Contact lever/knob	4
	4: Start turning/twisting lever/knob	4
	5: Finish turning/twisting lever/knob	4
Microwave thawing	1: Fail to open microwave door	4
	2: Open microwave door	4
	3: Pick up the food item and place it in the microwave	4
	4: Close the microwave door	4
	5: Press the microwave start button	4
Restock pantry	1: Fail to pick up the first item	7
	2: Pick up the first item and place it in the pantry	7
	3: Pick up the second item and place it in the pantry	7
Arrange vegetables	1: Fail to pick up the first vegetable	7
	2: Pick up the first vegetable and place it on the cutting board	7
	3: Pick up the second vegetable and place it on the cutting board	7
Prepare coffee	1: Fail to pick up the mug	7
	2: Pick up the mug and place it in the coffee machine	7
	3: Press the start button on the coffee machine	7
Pre-soak pan	1: Fail to pick up the pan	5
	2: Pick up the pan and place it in the sink	5
	2: Pick up the sponge and place it in the pan	5
	4: Turn the sink handle to turn on the sink	5

C Details on evaluation tasks

C.1 Frame-level task progress estimation

This task evaluates the model's ability to estimate the scalar progress of a robot toward task completion at every frame of a video, conditioned on a natural language task description. Ground-truth progress is calculated based on the simulator state at each moment of the video (Section 4.2). Given only task description and the video frames, the model must provide a numerical value reflecting task progress at each timestep. The evaluation metrics for this task include the L2 distance and the Pearson correlation coefficient between the predicted and ground-truth progress values across all frames. In addition, as a measure of reasoning quality that is independent of the ground-truth value calculations, we evaluate the correlation between the video frame count and the VLM-predicted task progress, which should decrease as videos deviate further from the known expert trajectory.

C.2 Frame-level natural language reasoning

For this task, we evaluate the model's natural language text output as it reasons about what is happening at each frame of the video, such as "The robot gripper is moving closer to the mug" or "The robot is holding the carrot." The evaluation metrics include the frame-level reasoning error rate and success rate. The error rate is the percentage of frames where the generated description includes a statement that is verifiably false (i.e., directly contradicts information from the ground-truth simulator state at that moment). The success rate reflects the percentage of frames where the model states something that is verifiably true (i.e., is consistent with information from the ground-truth simulator state at that moment) and does not state something verifiably false. This task measures the model's capacity to precisely localize events in time, avoid hallucinations, and generate semantically faithful descriptions at a fine granularity.

To implement this for the results shown in Section 5, we treat the evaluation as a language model task, where we prompt the model with the frame description (generated by ROVER, GVL, or TemporalConcat) paired with the ground-truth information and ask if the description includes any statements that are verifiably true or false relative to the ground truth. We use `gpt-4.5-2025-02-27` as the language model for this evaluation.

C.3 Video question answering

The Video QA task tests a model's ability to reason over entire trajectories and answer questions about actions that may or may not have occurred. Each question is posed in the format: "Did the robot action? If so, when?" where action includes task-relevant operations like "contact the mug", "drop the potato", or "place the bowl in the microwave". Questions are automatically generated from a templated action grammar, grounded in the specific object and scene configuration of each video. The 27 robot manipulation tasks are grouped into 9 categories (Table 1). We ask similar questions within each task group based on the shared action types (Table 3). The evaluation metrics include:

- Accuracy: the percentage of questions for which the model correctly identifies whether the event occurred.
- Recall: the percentage of actual occurrences correctly identified by the model.
- Precision: the percentage of predicted occurrences that actually occurred in the video.
- Temporal distance: the difference in time between when something occurs during a video and when the VLM states that it occurred.

For the results shown in Section 5, we generate answers to each question for this task using the frame-level descriptions produced by ROVER, GVL, and TemporalConcat. For each question, we use a language model to generate an answer given all frame descriptions concatenated together. We again use `gpt-4.5-2025-02-27` as the language model for this evaluation.

Table 3: Questions for video QA task.

Task group	**Questions**
Pick-and-place	Did the robot contact the {obj}? If so, when? Did the robot pick up the {obj}? If so, when? Did the robot drop the {obj}? If so, when? Did the robot place the {obj} in {target_location}? If so, when?
Open/close doors/drawers	Did the robot contact the door/drawer handle? If so, when? Did the robot grasp the door/drawer handle? If so, when? Did the robot start opening/closing the door/drawer? If so, when? Did the robot completely open/close the door/drawer? If so, when?
Turn on/off appliances	Did the robot press the start/stop button of the microwave/coffee machine? If so, when?
Toggle fixtures/knobs	Did the robot contact the sink spout/stove? If so, when? Did the robot start turning the sink spout/sink handle/stove knob? If so, when? Did the robot completely turn on/off the sink/stove? If so, when?
Microwave thawing	Did the robot open the microwave? If so, when? Did the robot pick up the {obj}? If so, when? Did the robot drop the {obj}? If so, when? Did the robot place the {obj} in the microwave? If so, when?
Restock pantry	Did the robot pick up the {obj1}? If so, when? Did the robot drop the {obj1}? If so, when? Did the robot place the {obj1} in the cabinet? If so, when? Did the robot pick up the {obj2}? If so, when? Did the robot drop the {obj2}? If so, when? Did the robot place the {obj2} in the cabinet? If so, when?
Arrange vegetables	Did the robot pick up the {vegetable1}? If so, when? Did the robot drop the {vegetable1}? If so, when? Did the robot place the {vegetable1} in the cabinet? If so, when? Did the robot pick up the {vegetable2}? If so, when? Did the robot drop the {vegetable2}? If so, when? Did the robot place the {vegetable2} in the cabinet? If so, when?
Prepare coffee	Did the robot pick up the mug? If so, when? Did the robot drop the mug? If so, when? Did the robot place the mug in the coffee machine? If so, when? Did the robot press the coffee machine start button? If so, when?
Pre-soak pan	Did the robot pick up the pan? If so, when? Did the robot drop the pan? If so, when? Did the robot place the pan in the sink? If so, when? Did the robot pick up the sponge? If so, when? Did the robot drop the sponge? If so, when? Did the robot place the sponge in the pan? If so, when? Did the robot contact the sink handle? If so, when? Did the robot turn on the sink? If so, when?

D Prompting and implementation details

D.1 GVL

Generative Value Learning (GVL) [31] is the existing state-of-the-art in-context learning approach for VLM reasoning over video input for embodied tasks. The method is motivated by the observation in Ma et al. [31] that, when presented a choronological sequence of video frames, VLMs often ignore the trajectory quality and instead hallucinate monotonically increasing task progress. To disrupt this temporal bias, GVL randomly shuffles the input frames. In doing so, GVL forces the model to pay attention to what is happening to each frame, improving frame-level task progress prediction. We implement GVL using the following prompt from Ma et al. [31].

```
********************** SYSTEM PROMPT
You are an expert roboticist tasked to predict task completion
percentages for frames of a robot for the task of {task_description}.
Note that the robot has an unknown level of expertise with the task
and may perform actions that lead it signicantly further from
accomplishing the task. Note that these frames are in random order,
so please pay attention to the individual frames when reasoning
about task completion percentage. If the progress at the current
frame is less than the progress of the initial robot scene, then the
task completion percentages should be negative.

For the task of {task_description}, output the task completion
percentage for the following frames that are presented in random
order. For each frame, format your response as follows:
Frame description: {}
Task completion percentage: {}%

********************** TASK PROMPT
Initial robot scene: [IMG]
Frame description: {first_frame_description}
Task completion percentage: 0%

Frame 1: [IMG]
...
...
...
Frame n: [IMG]
```

D.2 ROVER

Similar to GVL, we implement ROVER using an in-context learning approach, with the same prompting, shown below, used for all backbone VLMs. As discussed in Section 3, for each frame, the VLM describes the frame and either specifies a new subtask that needs to be completed or predicts progress for the existing subtask. If a new subtask is specified, then a new line of reasoning is created (using the same prompting shown below) with the updated `{task_description}` for the new subtask. To specify a new subtask, the VLM generates the text tokens "The robot needs to: {new_subtask}" as depicted in Figure 1. If a new subtask is not specified, the VLM proceeds with the frame-by-frame progress prediction for the existing subtask. To move to the next frame, the VLM generates the text tokens "[next-frame]" as shown in the prompting below. As described in Section 3, we implement a sliding window when appending the next frame, o_{next}, to the existing context, Y. When appending a new frame, $Y = \text{Window}(Y + o_{next})$, the window extracts three frames: the first frame of the current line of reasoning and the most recent previous frame (each followed by their previously generated natural language descriptions), along with the newly added next frame. The Window function formats these three frames as shown in the prompt below.

```
************************ SYSTEM PROMPT
You are an expert roboticist tasked to predict subtask completion
percentages for frames of a robot for the subtask of {task_description}.

The frames are shown in temporal order. Frame 0 represents the beginning
of the subtask. Note that the robot has an unknown level of expertise with
the subtask and may perform actions that lead it signicantly further from
accomplishing the subtask. Therefore, please pay attention to the
individual frames when reasoning about subtask completion percentage. If
the progress at the current frame is less than the progress of the initial
robot scene, then the completion percentages should be negative.

If the given subtask can be decomposed into multiple subtasks, please
specify a new subtask. Some examples of decomposition are provided below.
Subtask: 'pick the cheese from the counter and place it in the cabinet'
New Subtasks: ['grasp the cheese', 'place the cheese in the cabinet']
...
...

If you decompose the subtask further or progress to the next subtask,
format your response as follows:
Frame description: {}
The robot needs to: {}

If you do not decompose the subtask further or progress to the next
subtask, format your response as follows:
Frame description: {}
Subtask completion percentage: {}%
[next-frame]

************************ TASK PROMPT
Initial robot scene: [IMG]
Frame description: {first_frame_description}
Subtask completion percentage: 0%

###### If VLM has progressed beyond second frame of existing subtask
Most recent previous frame: [IMG]
Frame description: {prev_frame_description}
Subtask completion percentage: {prev_subtask_progress}%
######

Current frame: [IMG]
```

E Simulation environment and dataset of expert demonstrations

We leverage RoboCasa [39] as the simulation environment, which includes a large dataset of expert demonstrations, collected via human teleoperation, for household tasks in its initial release. RoboCasa is a large-scale simulation framework centered around home environments for training generalist robots. RoboCasa builds upon RoboSuite [66], a framework based in MuJoCo with a focus on physical realism, high speed, and modular design. RoboCasa contains 120 kitchen scenes and over 2,500 3D assets spanning 153 unique object categories, created with the aid of generative AI tools [39] (depicted in Figures 14 and 15 from Nasiriany et al. [39]). RoboCasa includes a dataset of demonstrations collected through human teleoperation. A team of four human operators collected 50 high-quality demonstrations for each atomic task, along with 5 composite tasks, using a 3D SpaceMouse [65, 66].

Figure 14: Figure from Nasiriany et al. [39] showing RoboCasa kitchen scenes.

Figure 15: Figure from Nasiriany et al. [39] showing RoboCasa interactable appliances.

F Stratifying results by state type and context length

GVL and TemporalConcat attempt to reason over the full video by concatenating all frames together. Figure 16 shows the correlation GVL and TemporalConcat achieve with ground-truth value estimates when reasoning over 0-10 frames, 10-20 frames, and 20-30 frames. In addition, the figure further stratifies results based on the state type at the current timestep (expert, non-expert, or interpolating between expert and non-expert) of the trajectory. These are the same results presented in Section 5.2, but now stratified across these two dimensions (instead of stratifying across trajectory level, as done in Figure 3). We see that the performance of GVL and TemporalConcat degrades when encountering non-expert moments of the trajectory, especially when the number of frames included in the context extends beyond 10. We observe a similar trend below when stratifying the error rates with the frame-level natural language reasoning across context length and state type (Figure 17). ROVER limits the maximum number of frames the model must reason over to three frames using the sliding context window (described in Section 3). In doing so, ROVER avoids these errors observed in long-context reasoning.

F.1 Frame-level task progress estimation

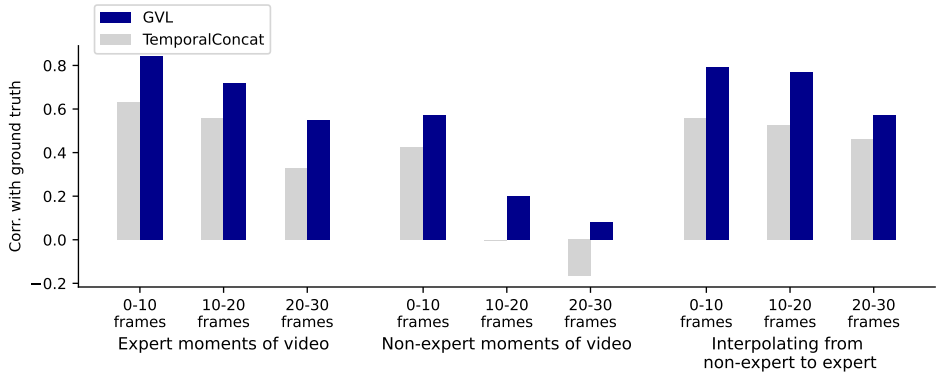

Figure 16: Average correlation GVL and TemporalConcat achieve with ground-truth value estimates in the frame-level progress prediction task when reasoning over frame sequences of different lengths.
.

F.2 Frame-level natural language reasoning

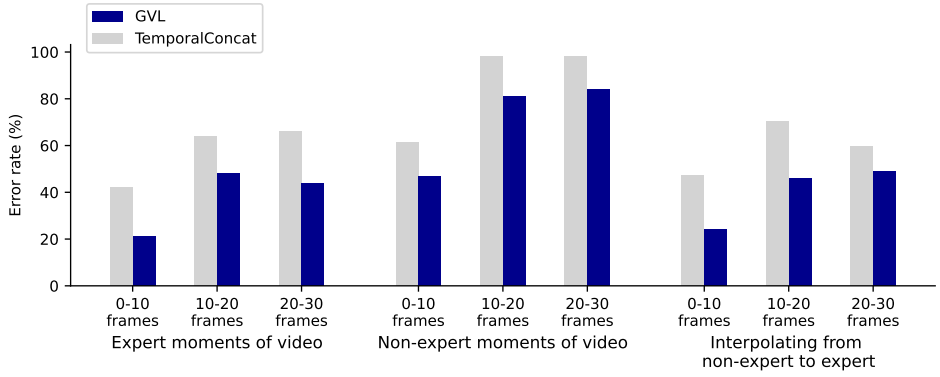

Figure 17: Average error rate of GVL and TemporalConcat with the frame-level natural language reasoning task when reasoning over frame sequences of different lengths.

H Robustness to video length and frame rate

The total frame count of a video is determined by two factors: 1) the video length: the total amount of time that the video represents and 2) the frame rate: the number of image frames used to represent each unit of time. These two factors can vary widely for a given video at test time. Therefore, for a method to achieve reliable performance in the wild, it should be robust to both factors.

ROVER scales more readily to longer videos and higher frame rates. We see that ROVER, by leveraging recursive reasoning and using a maximum context of three frames, is not only more efficient, but is also less sensitive to video length and frame rate of the given video relative to GVL and TemporalConcat. We first observe this in the results stratified across task group (Figures 3 and 5 from the main results in Section 5), where we see the improved performance of ROVER for the composite tasks (bottom row of each figure), which include much longer videos than the atomic tasks (top row of each figure). We also observe ROVER's robustness to frame rate when varying the total number of frames used to represent each video between 30 and 240 (i.e., increasing the effective frame rate by a factor of 8) for the 10 pick-and-place tasks for the progress prediction task (Figure 18) and frame-level natural language reasoning (Figure 19). These figures show that the performance of GVL and TemporalConcat significantly degrades when given the same pick-and-place videos, but with effective frame rate increased.

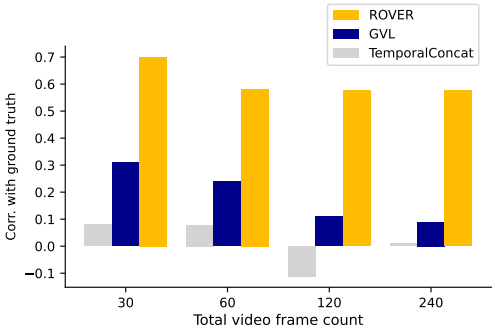

Figure 18: Average correlation each method achieves with ground-truth value estimates in the frame-level progress prediction task when increasing the total frame count from 30 to 240 (by increasing the frame rate) for the 10 pick-and-place tasks.

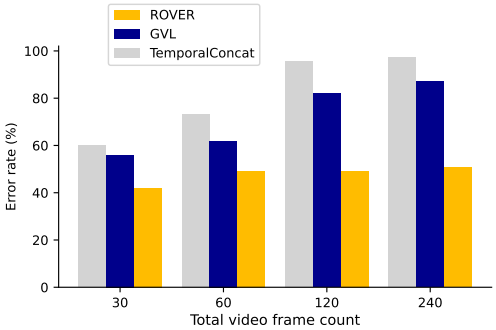

Figure 19: Average error rate of each method with the frame-level natural language reasoning task when increasing the total frame count from 30 to 240 (by increasing the frame rate) for the 10 pick-and-place tasks.

I Robustness to camera view

The main results presented in Section 5 use the wrist view (eye in hand view), as all methods were observed to perform best on the manipulation tasks using this view. The figures below show the results of the progress prediction task (Figure 20) and frame-level natural language reasoning task (Figure 21) when using different camera views. The left view and right view are fixed third-person views with the camera positioned to the left and right, respectively, of the robot at a distance sufficent to show the full robot arm within the camera frame. Although all methods decline in performance with these fixed external views (likely due to occlusions of the object or gripper at different moments during the trajectory), ROVER continues to outperform baselines across all camera views.

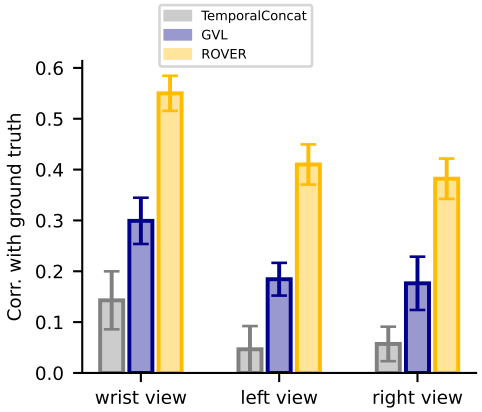

Figure 20: Mean and standard error of correlation between ground-truth value estimates and progress values predicted by each method stratified across different camera views.

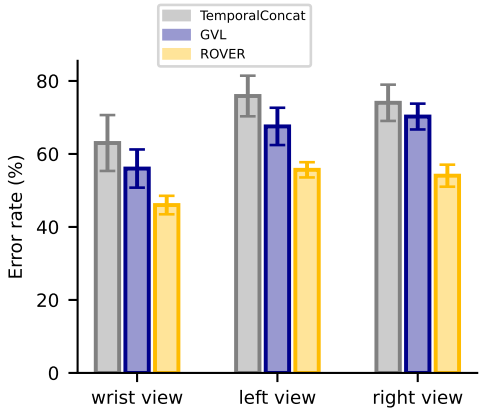

Figure 21: Mean and standard error of error rate in frame-level natural language reasoning of each method stratified across different camera views.

J Testing across different backbone models

To ensure consistency across benchmarks and with prior work, We use `gemini-1.5-pro` for Gemini-1.5-Pro and `gemini-2.5-pro-preview` for Gemini-2.5-Pro-Preview from Google's Gemini API. We use `gpt-4o` from the OpenAI API for GPT-4o. We use `Qwen2.5-VL-32B-Instruct` for Qwen2.5-VL-32B available on Huggingface. For the open source model, all experiments were performed on A6000 GPUs. Overall, we see that ROVER consistently outperforms baselines across all backbone VLMs.

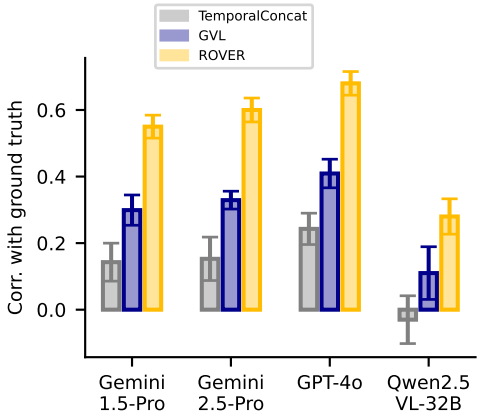

Figure 22: Mean and standard error of correlation between ground-truth value estimates and progress values predicted by each method stratified across different backbone VLMs.

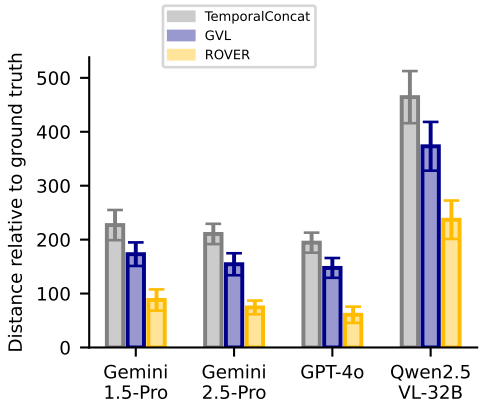

Figure 23: Mean and standard error of distance between ground-truth value estimates and progress values predicted by each method stratified across different backbone VLMs.

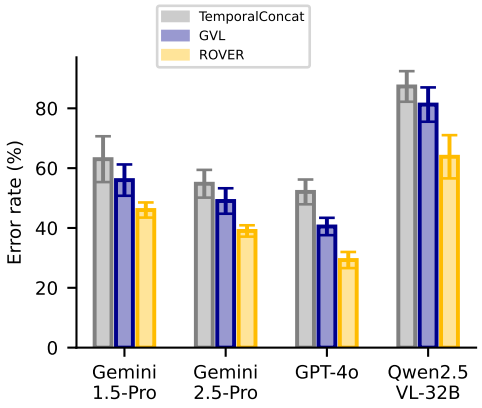

Figure 24: Mean and standard error of error rate in frame-level natural language reasoning of each method stratified across different backbone VLMs.

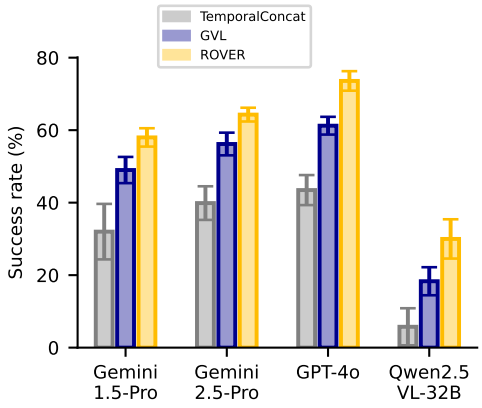

Figure 25: Mean and standard error of success rate in frame-level natural language reasoning of each method stratified across different backbone VLMs.

K Task progress prediction with real-world OpenX Embodiment datasets

We perform the same task progress prediction analysis conducted in Ma et al. [31] using 1,000 videos from 50 real-world OpenX Embodiment datasets (20 videos randomly selected from each dataset) [41]. The tables below show the average correlation each method achieves with the ground-truth value estimates across the OXE datasets when using Gemini-1.5-Pro (Table 4) and GPT-4o (Table 5) as the backbone VLM. Since there are no true value estimates for these real-world videos, we follow Ma et al. [31] and use frame number as the ground-truth value estimate. For known high-quality datasets collected from human teleoperators with fixed camera placements, such as Bridge [13], RT-1 [7], and Dobb-E [47], ROVER achieves a significantly higher correlation with frame number than GVL (highlighted in yellow in the tables below). Both ROVER and GVL show low correlation with frame number for datasets with videos that show lower quality trajectories produced by autonomous data collection via scripted motions or motor babbling, such as QT-OPT [23] and RoboNet [10] (highlighted in green in the tables below).

Table 4: Correlation of each method with ground-truth progress estimates (based on frame number) on OpenX Embodiment datasets with Gemini-1.5-Pro as the backbone VLM.

Dataset	GVL	ROVER
utokyo_xarm_pick_and_place_converted_externally_to_rlds	0.753	0.898
utokyo_xarm_bimanual_converted_externally_to_rlds	0.822	0.890
nyu_door_opening_surprising_effectiveness	0.668	0.879
berkeley_autolab_ur5	0.761	0.874
utokyo_pr2_tabletop_manipulation_converted_externally_to_rlds	0.619	0.870
maniskill_dataset_converted_externally_to_rlds	0.740	0.866
utokyo_pr2_opening_fridge_converted_externally_to_rlds	0.824	0.856
fractal20220817_data	0.762	0.817
iamlab_cmu_pickup_insert_converted_externally_to_rlds	0.513	0.805
toto	0.564	0.780
ucsd_kitchen_dataset_converted_externally_to_rlds	0.529	0.768
utaustin_mutex	0.576	0.767
asu_table_top_converted_externally_to_rlds	0.477	0.762
austin_sirius_dataset_converted_externally_to_rlds	0.539	0.755
dobbe	0.487	0.754
berkeley_cable_routing	0.488	0.752
berkeley_rpt_converted_externally_to_rlds	0.495	0.748
viola	0.415	0.747
fmb	0.555	0.739
austin_buds_dataset_converted_externally_to_rlds	0.350	0.727
usc_cloth_sim_converted_externally_to_rlds	0.457	0.722
bridge	0.457	0.717
jaco_play	0.414	0.710
stanford_hydra_dataset_converted_externally_to_rlds	0.351	0.701
bc_z	0.397	0.696
berkeley_mvp_converted_externally_to_rlds	0.360	0.680
cmu_stretch	0.276	0.677
tokyo_u_lsmo_converted_externally_to_rlds	0.362	0.639
berkeley_fanuc_manipulation	0.252	0.632
roboturk	0.314	0.631
ucsd_pick_and_place_dataset_converted_externally_to_rlds	0.281	0.615
dlr_edan_shared_control_converted_externally_to_rlds	0.119	0.598
dlr_sara_pour_converted_externally_to_rlds	0.115	0.564
droid	-0.002	0.546
taco_play	0.083	0.539
stanford_robocook_converted_externally_to_rlds	-0.010	0.513
imperialcollege_sawyer_wrist_cam	-0.038	0.432
kaist_nonprehensile_converted_externally_to_rlds	-0.021	0.425
austin_sailor_dataset_converted_externally_to_rlds	-0.072	0.371
kuka	0.304	0.316
cmu_play_fusion	-0.096	0.304
stanford_kuka_multimodal_dataset_converted_externally_to_rlds	-0.110	0.295
nyu_franka_play_dataset_converted_externally_to_rlds	-0.139	0.287
stanford_mask_vit_converted_externally_to_rlds	-0.131	0.270
uiuc_d3field	-0.212	0.169
robo_net	-0.240	0.096
columbia_cairlab_pusht_real	-0.229	0.095
dlr_sara_grid_clamp_converted_externally_to_rlds	-0.301	-0.024
cmu_franka_exploration_dataset_converted_externally_to_rlds	-0.230	-0.078

Table 5: Correlation of each method with ground-truth progress estimates (based on frame number) on OpenX Embodiment datasets with GPT-4o as the backbone VLM.

Dataset	GVL	ROVER
utokyo_pr2_opening_fridge_converted_externally_to_rlds	0.907	0.969
nyu_door_opening_surprising_effectiveness	0.891	0.928
berkeley_autolab_ur5	0.749	0.894
utaustin_mutex	0.837	0.893
berkeley_mvp_converted_externally_to_rlds	0.835	0.889
fractal20220817_data	0.788	0.879
utokyo_xarm_bimanual_converted_externally_to_rlds	0.714	0.874
utokyo_pr2_tabletop_manipulation_converted_externally_to_rlds	0.722	0.867
austin_sirius_dataset_converted_externally_to_rlds	0.717	0.867
toto	0.717	0.851
dlr_edan_shared_control_converted_externally_to_rlds	0.696	0.840
dobbe	0.568	0.829
iamlab_cmu_pickup_insert_converted_externally_to_rlds	0.554	0.824
utokyo_xarm_pick_and_place_converted_externally_to_rlds	0.726	0.822
ucsd_kitchen_dataset_converted_externally_to_rlds	0.629	0.814
berkeley_fanuc_manipulation	0.603	0.807
viola	0.503	0.804
asu_table_top_converted_externally_to_rlds	0.505	0.803
bridge	0.661	0.799
maniskill_dataset_converted_externally_to_rlds	0.527	0.787
jaco_play	0.570	0.786
berkeley_rpt_converted_externally_to_rlds	0.616	0.781
roboturk	0.586	0.769
usc_cloth_sim_converted_externally_to_rlds	0.466	0.765
uiuc_d3field	0.524	0.730
austin_buds_dataset_converted_externally_to_rlds	0.458	0.728
kaist_nonprehensile_converted_externally_to_rlds	0.483	0.704
austin_sailor_dataset_converted_externally_to_rlds	0.353	0.681
berkeley_cable_routing	0.278	0.676
tokyo_u_lsmo_converted_externally_to_rlds	0.394	0.674
ucsd_pick_and_place_dataset_converted_externally_to_rlds	0.245	0.663
cmu_play_fusion	0.374	0.661
dlr_sara_pour_converted_externally_to_rlds	0.244	0.615
imperialcollege_sawyer_wrist_cam	0.219	0.602
stanford_robocook_converted_externally_to_rlds	0.206	0.598
stanford_hydra_dataset_converted_externally_to_rlds	0.178	0.598
cmu_stretch	0.140	0.598
bc_z	0.158	0.591
nyu_franka_play_dataset_converted_externally_to_rlds	0.182	0.567
robo_net	0.367	0.438
stanford_kuka_multimodal_dataset_converted_externally_to_rlds	-0.042	0.434
stanford_mask_vit_converted_externally_to_rlds	-0.042	0.433
columbia_cairlab_pusht_real	-0.048	0.416
eth_agent_affordances	-0.088	0.416
cmu_franka_exploration_dataset_converted_externally_to_rlds	-0.064	0.356
taco_play	-0.094	0.299
kuka	0.150	0.170
dlr_sara_grid_clamp_converted_externally_to_rlds	-0.301	-0.047

L Ablations

As described in Section 5.5, we test two ablations of ROVER: one variant ablates only the sliding window while the other ablates only the recursive reasoning. We observe that baseline methods such as TemporalConcat often hallucinate steady, monotonically increasing task progress values, even when the video includes significant stalls or regressions. GVL addresses this problem via random frame shuffling, which disrupts the strong temporal priors and forces the model to assess each frame more carefully [31]. We observe that the sliding window, even when ablating the recursive decomposition, similarly reduces hallucination by retaining temporal context while avoiding long-context sequential processing. Our ablations show that the window-only variant of ROVER significantly outperforms TemporalConcat and matches or exceeds GVL's performance on short-horizon tasks, such as turning on/off appliances or toggling fixtures/knobs. We see that adding the recursive decomposition further improves performance for more complex tasks consisting of numerous subtasks.

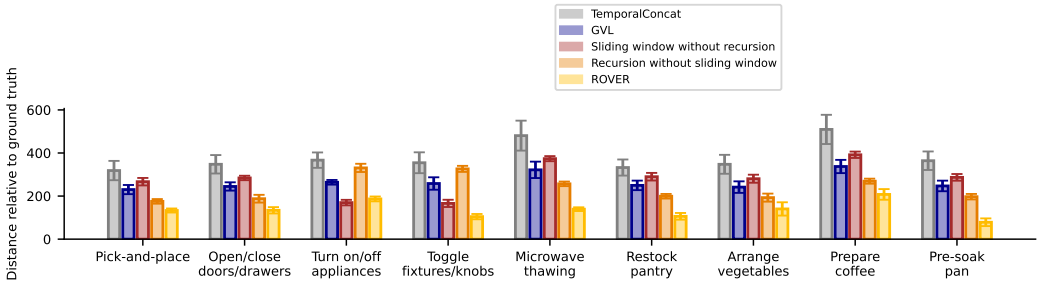

Figure 26: Mean and standard error of distance between ground-truth value estimates and progress values predicted by baselines and ablated versions of ROVER stratified by task group.

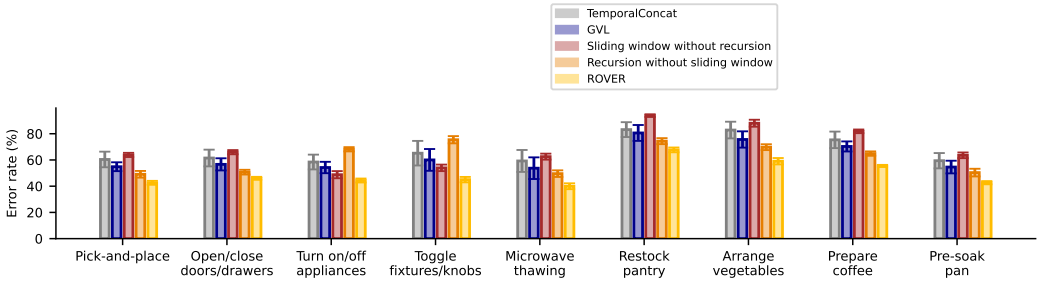

Figure 27: Mean and standard error of error rate in frame-level natural language reasoning of baselines and ablated versions of ROVER stratified by task group.

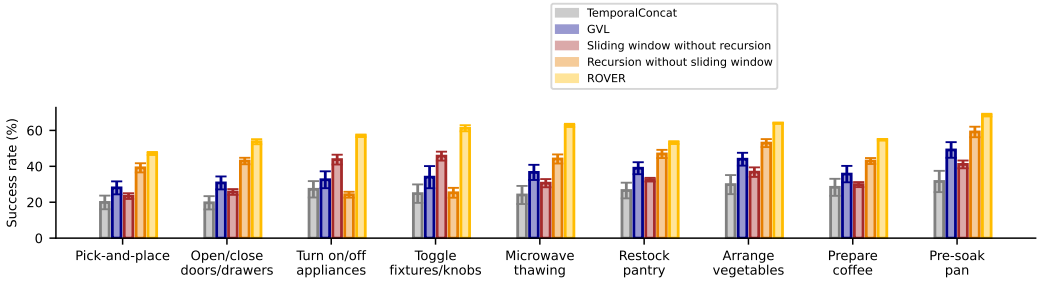

Figure 28: Mean and standard error of success rate in frame-level natural language reasoning of baselines and ablated versions of ROVER stratified by task group.

M Additional details on object-centric subtasks

When generating non-expert trajectories (described in Section 4.1) and computing ground-truth value estimates (described in Section 4.2), we follow the work in MimicGen [36] in defining object-centric subtasks for each task trajectory. Specifically, if we let $E = \{e_1, e_2, ...\}$ be the set of objects in a task $\mathcal{M}$, we assume that each expert demonstration, $\tau^E \in D_{src}$, can be decomposed into object-centric segments $\tau^E = \{\tau_i^E(e_{\tau_i})\}_{i=1}^M$, where the manipulation in each subtask trajectory is relative to a single object's coordinate frame ($e_{\tau_i} \in E$). This sequence of object-centric subtasks is generally straight forward for a human to identify given a task [36]. For example, for the pick-and-place task of "pick the cup from the counter and place it in the sink", the subtasks include first picking up the cup, then placing it in the sink. This task can be broken down into two object-centric subtasks: a cup-grasping subtask (motion is relative to cup) and a cup-placement subtask (motion is relative to the sink basin). Similar to MimicGen [36], we assume access to metrics that allow the end of each subtask to be detected. In the previous example, this would correspond to detecting 1) when the cup has been lifted from the counter and 2) when the cup has been placed within the sink basin.

We generate the non-expert trajectories (by inserting random perturbations within the expert demonstrations as described in Section 4.1) and compute the value estimates (based on the distance measures described in Section 4.2) for each of these object-centric subtasks separately. To create the full non-expert trajectory, $\tau^N = \{\tau_i^N(e_{\tau_i^N})\}_{i=1}^M$, we simply sequence together the subtask trajectories. To form the value estimates for the full trajectory, we sequence together the value estimates corresponding to each subtask, adding the value estimates of each subtask to the final value of the previous subtask and, when complete, rescaling the list of values to the range $[0, 1]$.

O Full related work

Decomposition-based reasoning. ROVER enables more precise and efficient reasoning over video for long-horizon embodied tasks by segmenting the video input into subtask-specific segments through recursive decomposition. Prior work has explored decomposition in various forms, including neural modular architectures [2, 17], multi-hop QA [50], and multi-step reasoning for program synthesis or mathematical problems [37, 16, 28]. More recently, LLMs have been used to decompose problems via in-context learning or fine-tuning for complex tasks such as code generation and planning [55, 24, 64]. Adaptive decomposition methods have also emerged, including feedback-driven re-planning [42, 44] and multi-agent coordination [59]. Unlike these approaches, ROVER performs recursive decomposition over video input with VLMs. Further, we show that ROVER can be used to improve reasoning about physical interactions in embodied settings.

Goal-conditioned value functions and reward modeling. ROVER demonstrates an emerging use of VLMs as per-frame value estimators for embodied video tasks, leveraging their semantic understanding and long-context capabilities. Prior work developed models using robot [46] or human videos with discriminators [8], contrastive learning [4], or offline reinforcement learning [34, 32, 5]. With the advent of foundation models, researchers have begun incorporating language and vision models into downstream robotic applications including planning [1, 48, 12], imitation learning [6, 49], and symbolic reasoning [30, 20, 51].

Kwon et al. [27] and Mahmoudieh et al. [35] leverage language models to assign reward values to reinforcement learning agents. Klissarov et al. [26], Wang et al. [58], and again Kwon et al. [27] utilize these models to deliver preference-based feedback. Works such as Ma et al. [33], Yu et al. [62], and Xie et al. [60] go further by having LLMs generate code. Notably, all these approaches rely solely on the language understanding capabilities of foundation models.

Prior methods leveraging VLMs for reward prediction over video input primarily relied on two extremes: reasoning over a small set of frames (e.g., comparing two individual frames) [57, 25, 54, 63] or attempting to reason over the full video by concatenating all frames into a single context [31]. The former sacrifices global context, while the latter is inefficient (inference cost scales quadratically with video length) and overwhelms the model with visual information that is often redundant or irrelevant. ROVER addresses these limitations by leveraging recursive decomposition to enable high-precision frame-by-frame reasoning (including reward or task progress prediction), without sacrificing efficiency or global context.