Appendix Contents

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).

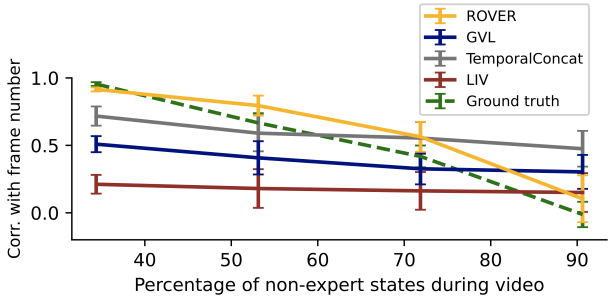

Figure 8: Mean and standard error of correlation between video frame number and the frame-level progress values predicted by each method stratified across proportion of non-expert states in video.

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).

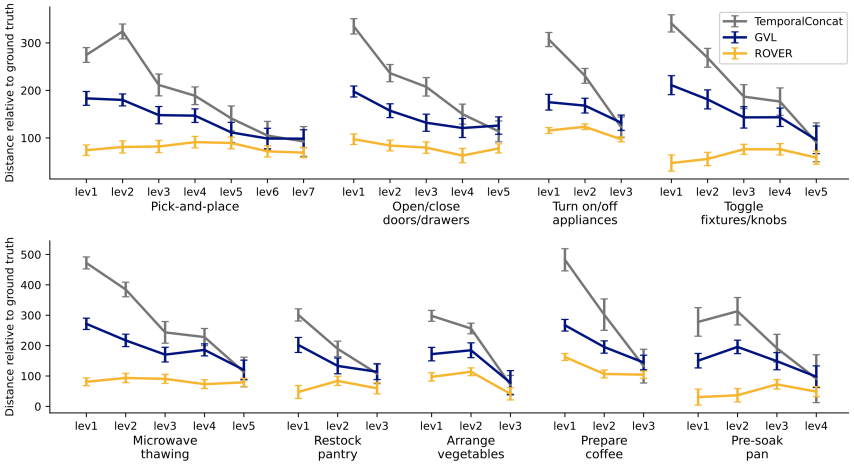

Figure 9: Mean and standard error of distance between model-predicted and ground-truth task progress estimates stratified across trajectory level.

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

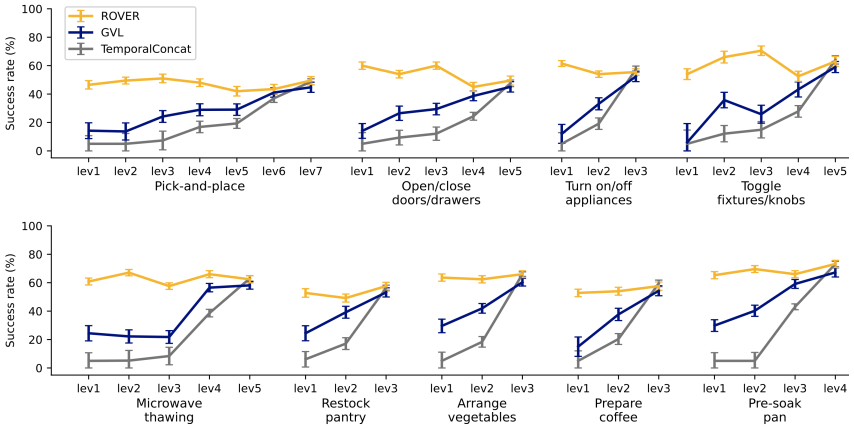

Figure 10: Mean and standard error of frame-level reasoning success rate (percentage of frames model states something that is verifiably correct relative to ground-truth information about simulator state) for videos stratified across trajectory level.

A.3 Task 3: Video QA

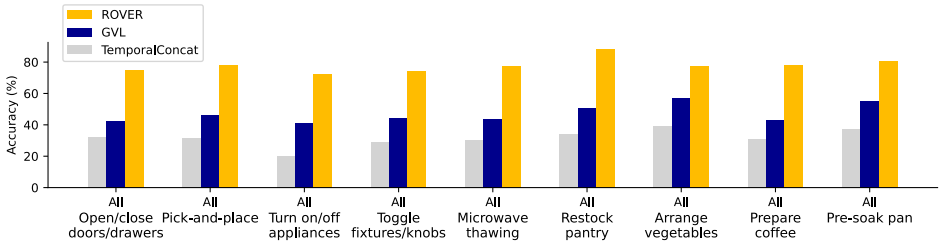

Figure 11: Video QA accuracy (the percentage of questions for which the model correctly identifies whether the event occurred) across all task groups.

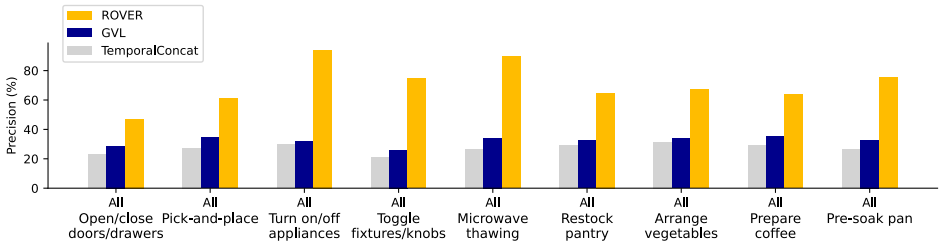

Figure 12: Video QA precision (the percentage of predicted occurrences that actually occurred in the video) across all task groups.

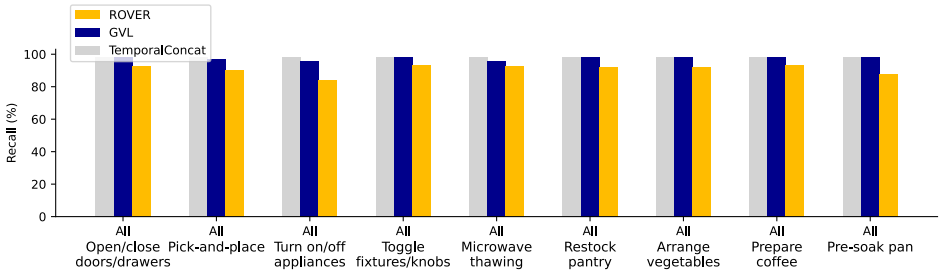

Figure 13: Video QA recall (the percentage of actual occurrences correctly identified by the model) across all task groups.

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.

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, $Y = \mathrm{Window}(Y + o_{next})$, the window extracts three frames: the first frame of the current line of reasoning and the most recent previous frame (each 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

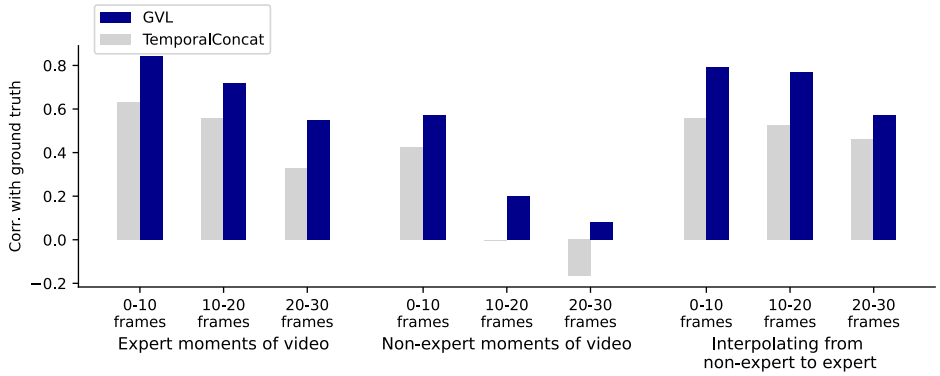

Figure 16: Average correlation GVL and TemporalConcat achieve with ground-truth value estimates in the frame-level progress prediction task when reasoning over frame sequences of different lengths.
.

F.2 Frame-level natural language reasoning

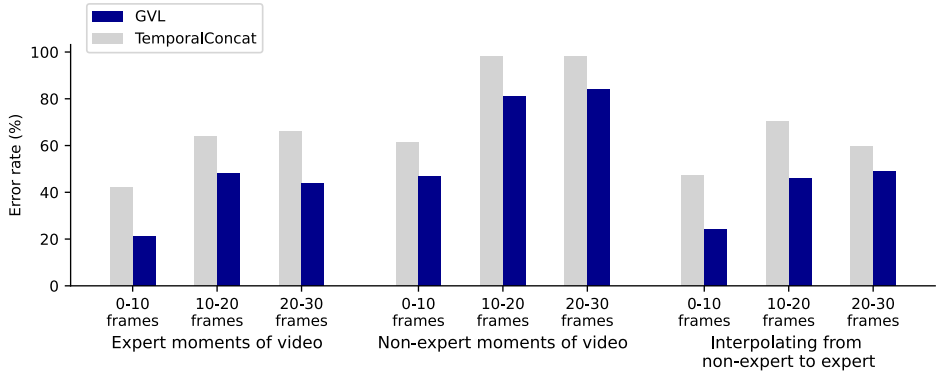

Figure 17: Average error rate of GVL and TemporalConcat with the frame-level natural language reasoning task when reasoning over frame sequences of different lengths.

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.

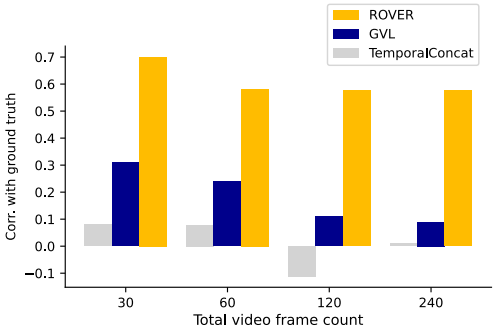

Figure 18: Average correlation each method achieves with ground-truth value estimates in the frame-level progress prediction task when increasing the total frame count from 30 to 240 (by increasing the frame rate) for the 10 pick-and-place tasks.

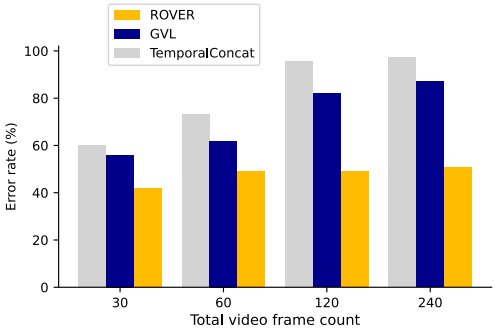

Figure 19: Average error rate of each method with the frame-level natural language reasoning task when increasing the total frame count from 30 to 240 (by increasing the frame rate) for the 10 pick-and-place tasks.

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.

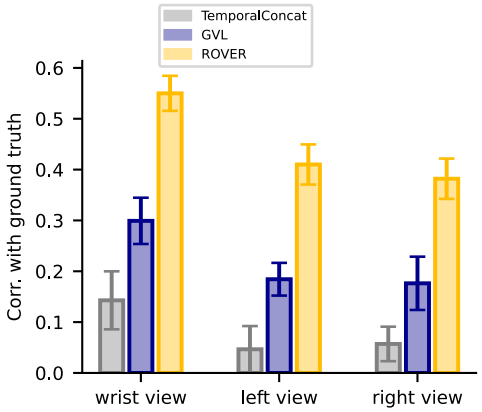

Figure 20: Mean and standard error of correlation between ground-truth value estimates and progress values predicted by each method stratified across different camera views.

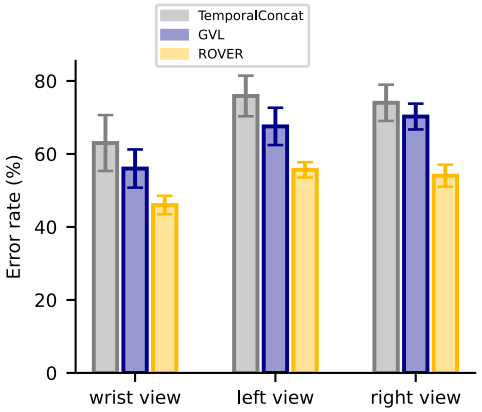

Figure 21: Mean and standard error of error rate in frame-level natural language reasoning of each method stratified across different camera views.

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.

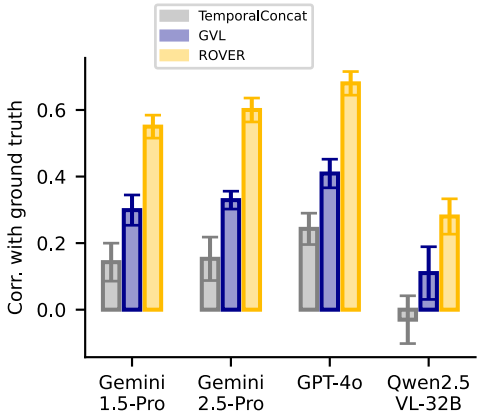

Figure 22: Mean and standard error of correlation between ground-truth value estimates and progress values predicted by each method stratified across different backbone VLMs.

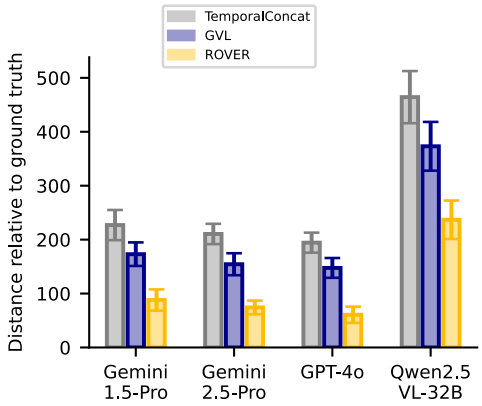

Figure 23: Mean and standard error of distance between ground-truth value estimates and progress values predicted by each method stratified across different backbone VLMs.

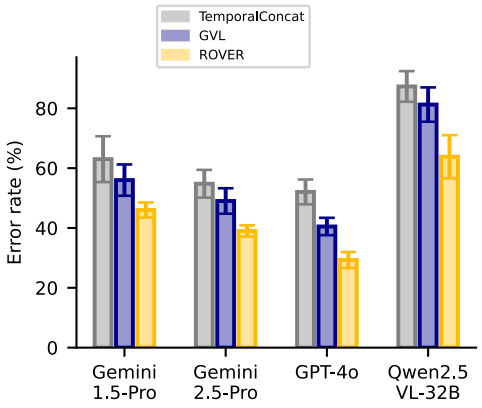

Figure 24: Mean and standard error of error rate in frame-level natural language reasoning of each method stratified across different backbone VLMs.

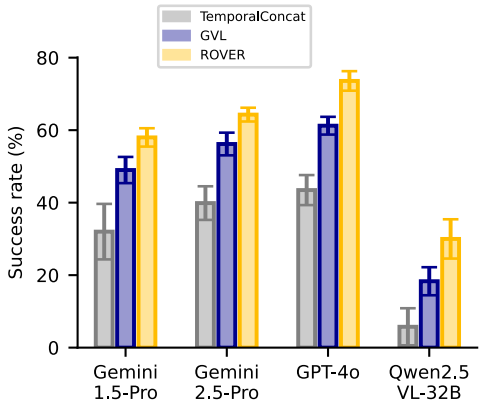

Figure 25: Mean and standard error of success rate in frame-level natural language reasoning of each method stratified across different backbone VLMs.

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).

Task: “pick the apple from the bowl”

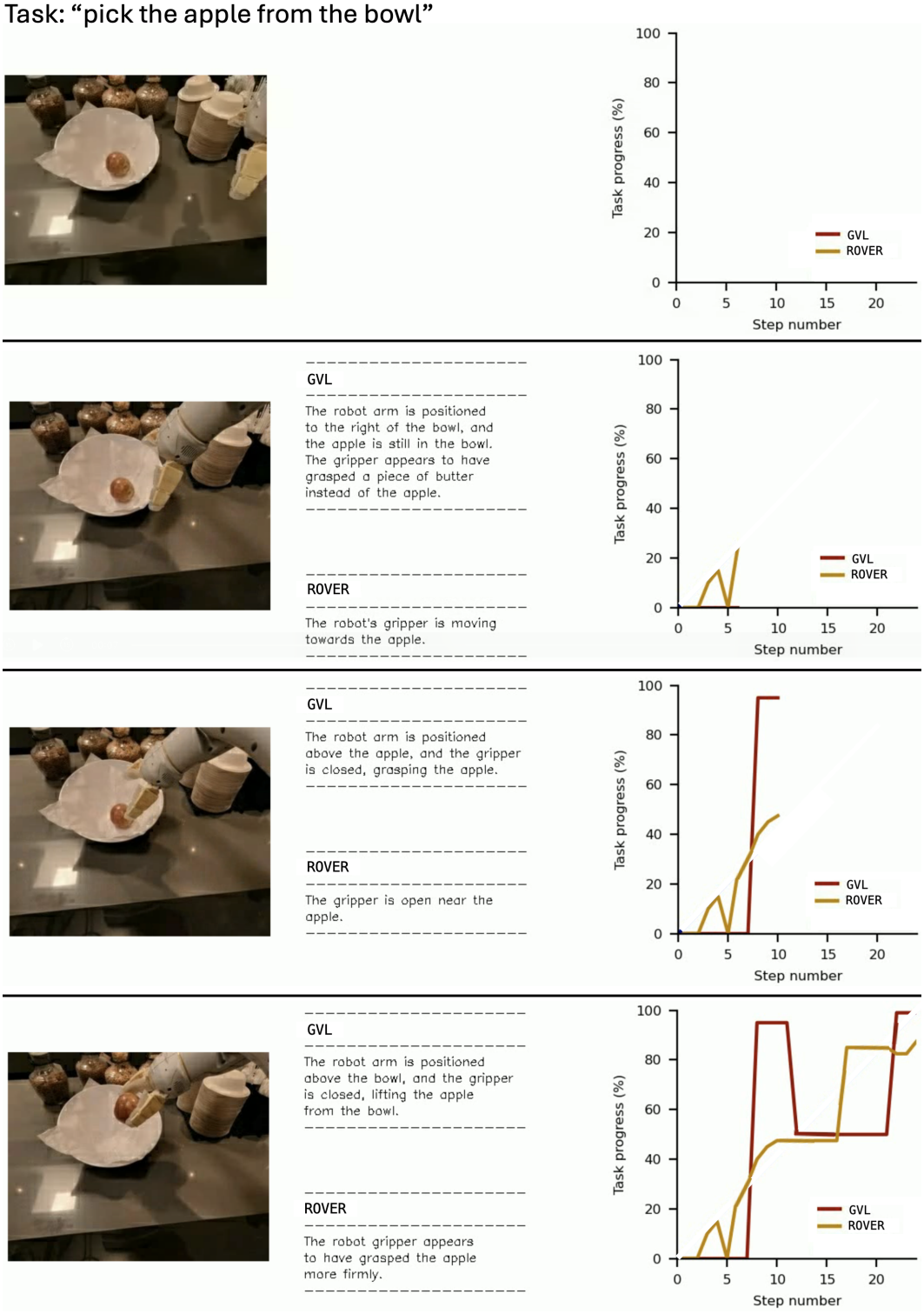

Figure 26: Examples of frame-level reasoning and progress prediction from GVL (red) versus ROVER (yellow) on real-world videos.

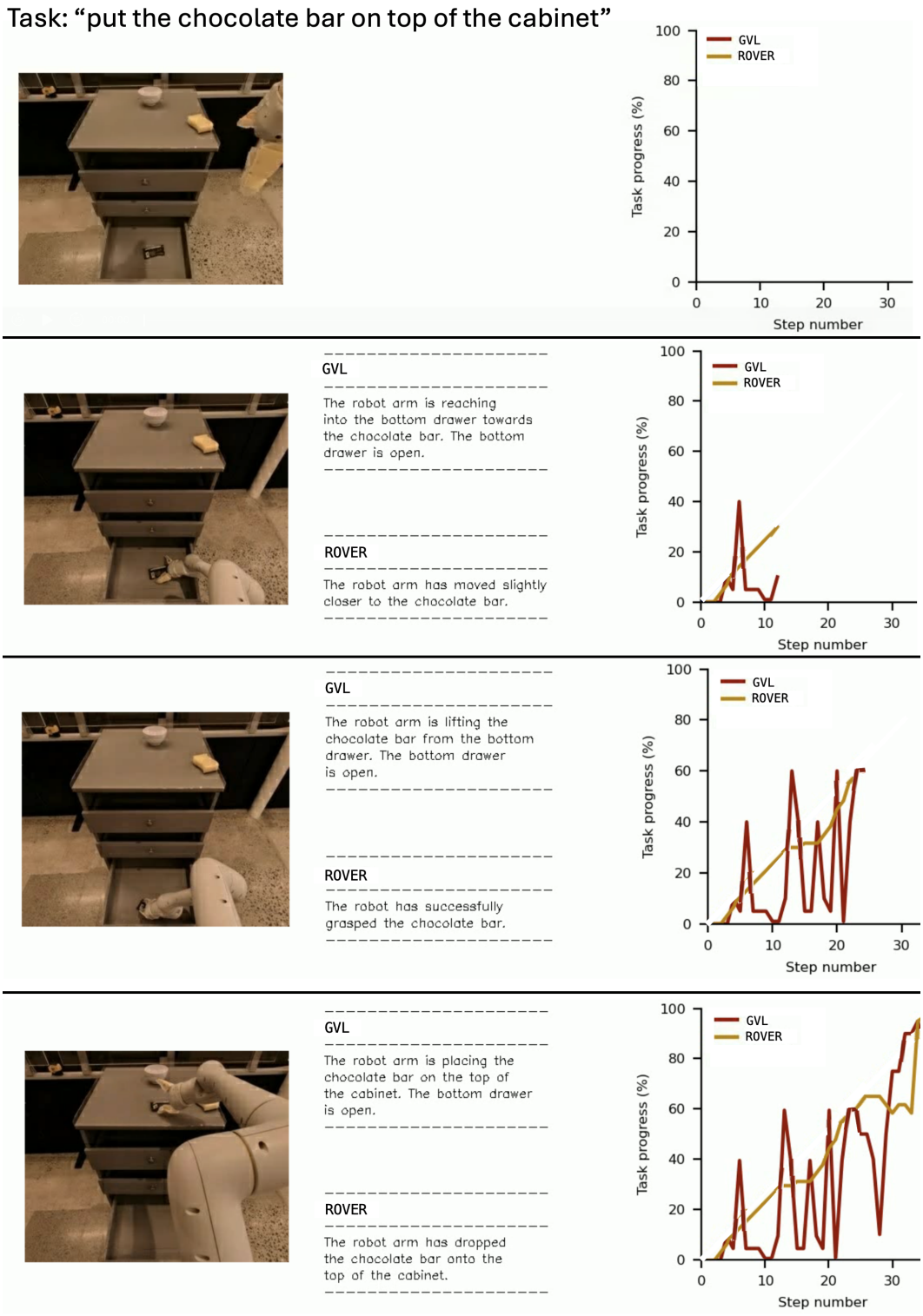

Figure 27: Examples of frame-level reasoning and progress prediction from GVL (red) versus ROVER (yellow) on real-world videos.

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.

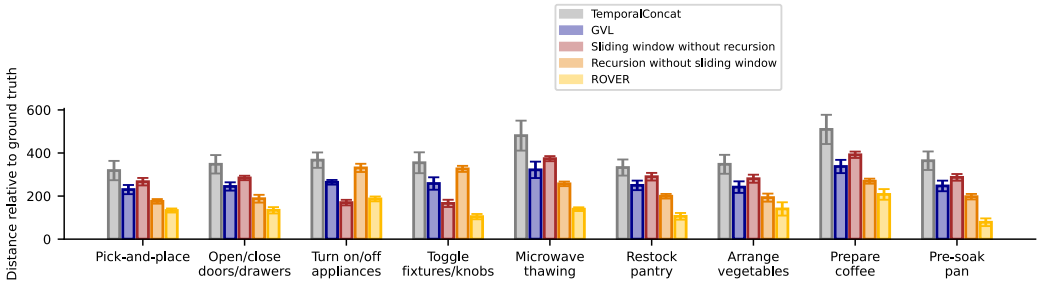

Figure 28: Mean and standard error of distance between ground-truth value estimates and progress values predicted by baselines and ablated versions of ROVER stratified by task group.

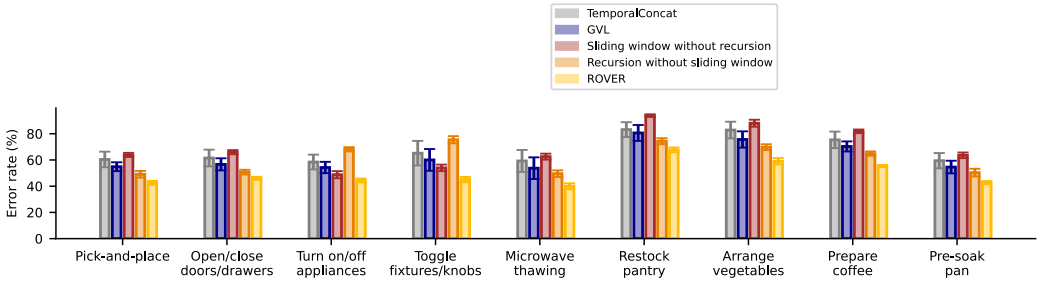

Figure 29: Mean and standard error of error rate in frame-level natural language reasoning of baselines and ablated versions of ROVER stratified by task group.

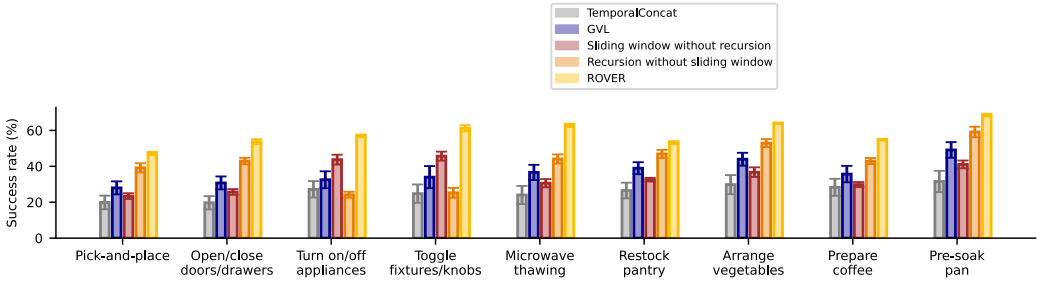

Figure 30: Mean and standard error of success rate in frame-level natural language reasoning of baselines and ablated versions of ROVER stratified by task group.

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.