ProMQA-Assembly: Multimodal Procedural QA Dataset on Assembly

Kimihiro Hasegawa[1] **Wiradee Imrattanatrai**[2] **Masaki Asada**[2] **Susan Holm**[1]
Yuran Wang[1] **Vincent Zhou**[1] **Ken Fukuda**[2] **Teruko Mitamura**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]National Institute of Advanced Industrial Science and Technology (AIST)
kimihiro@cs.cmu.edu

Abstract

Assistants on assembly tasks show great potential to benefit humans ranging from helping with everyday tasks to interacting in industrial settings. However, evaluation resources in assembly activities are underexplored. To foster system development, we propose a new multimodal QA evaluation dataset on assembly activities. Our dataset, ProMQA-Assembly, consists of 646 QA pairs that require multimodal understanding of human activity videos and their instruction manuals in an online-style manner. For cost effectiveness in the data creation, we adopt a semi-automated QA annotation approach, where LLMs generate candidate QA pairs and humans verify them. We further improve QA generation by integrating fine-grained action labels to diversify question types. Additionally, we create 81 instruction task graphs for our target assembly tasks. These newly created task graphs are used in our benchmarking experiment, as well as in facilitating the human verification process. With our dataset, we benchmark models, including competitive proprietary multimodal models. We find that ProMQA-Assembly contains challenging multimodal questions, where reasoning models showcase promising results. We believe our new evaluation dataset contributes to the further development of procedural-activity assistants.

Keywords: Multimodal, QA, Procedural Activity, Assembly, Evaluation

1. Introduction

Assembly tasks abound in everyday life from do-it-yourself (DIY) (Ben-Shabat et al., 2021; Liu et al., 2024; Jang et al., 2019) to industrial settings like manufacturing (Ragusa et al., 2021; Moriwaki et al., 2022; Wang et al., 2023a; Schoonbeek et al., 2024). Assistant systems for such procedural activities have the potential to further increase accessibility by providing on-point, situated feedback—similar to how parents teach their children to construct a shelf or how experts provide on-the-job training to beginners learning to repair cars. To facilitate the development of such assistants, we propose a new question-answering (QA) benchmark dataset to assess systems' capabilities in answering online-style questions about assembly procedures that require multimodal understanding.

Supporting procedural activities like assembly often involves reasoning over multiple sources and modalities of information. This includes understanding the correct action order from instructions, comprehending the current status of a target object based on its appearance and the sequence of actions performed so far, and combining the information to identify which steps are complete or incomplete. Additionally, the system must detect potential mistakes and either encourage users to proceed to the next step or make necessary corrections. Figure 1 illustrates an example of an assistant supporting a user on an assembly task.

When a user asks a question out of confusion, an assistant is expected to respond with natural language in an online manner by answering the question based solely on the available information so far, i.e., activity recording up to that point. This online-style setting imitates the practical situation better than an offline-style setting, where a system responds based on access to the whole video. While a model can assume the completion of all steps in the offline-style setting, the online-style setting necessitates a model to identify the completion of steps. Besides, the QA formulation can simulate the application scenario without any alteration, thanks to its expressiveness (Gardner et al., 2019; Rogers et al., 2023). While classification tasks, such as action recognition and temporal segmentation (Kuehne et al., 2014; Tang et al., 2019; Ding et al., 2022), have been widely adopted, they are suboptimal for the end task evaluation, as these classification tasks are subtasks of procedural activity assistance.

Instructions take a variety of forms, e.g., a list of text descriptions as in cooking recipes, a list of images (with a caption) as in IKEA manuals, or instructional videos as found on YouTube. Prior work typically adopts a task graph to represent instructions similar to scripts (Schank and Abelson, 1977; Sakaguchi et al., 2021). An instruction task graph is a partial directed acyclic graph (DAG), where nodes represent steps and edges represent the step order dependencies (Dvornik et al., 2022; Ashutosh et al., 2023; Peddi et al., 2024; Seminara et al., 2024). While instructions are not always available in the

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 9082–9104
11–16 May 2026.

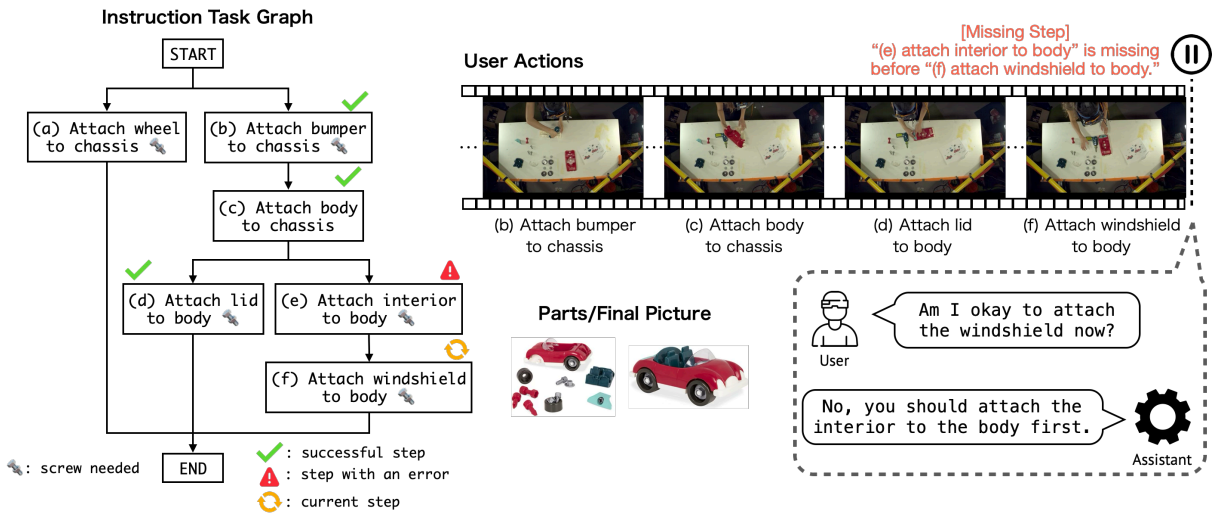


Figure 1: Task example: A user performs actions based on instructions. When the user asks a question, the assistant system responds to it based on the instructions and the past actions, by confirming if the user follows the correct order. In this example, the system needs to point out that the user is attaching the windshield before attaching the interior to the body and tell the user the mistake.

form of task graphs in the real world, the representation is suitable for benchmarking because it can capture even implicit pairwise step dependencies, providing accurate ordering information.

In this work, we introduce a new benchmark dataset, **ProMQA-Assembly** (**Pro**cedural **M**ultimodal **Q**uestion **A**nswering for **Assembly**), following ProMQA (Hasegawa et al., 2025) for cooking. ProMQA-Assembly consists of 646 QA pairs with corresponding video clips and instructions, where multi-view recordings of take-apart toy assembly are sampled from Assembly101 (Sener et al., 2022). ProMQA-Assembly contains notable differences from the original ProMQA (details in § 3.5), providing diversity that complements the original evaluation suite. Furthermore, in our QA annotation, we devise new prompt templates to improve ProMQA's semi-automatic QA annotation approach, i.e., generation by LLMs, and verification by humans, in terms of question diversity for the assembly data. Under controlled experiment settings, we found that integrating fine-grained action labels can diversify questions while maintaining the characteristics of multimodal procedural questions. Additionally, we annotate 81 task graphs, which is three times larger in size than those of a previous work (Peddi et al., 2024). These graphs serve as accurate representations of instructions for each toy in benchmarking experiments, as well as assisting human annotators in the QA verification process. Finally, we conducted benchmarking experiments, where we evaluated text-only models, open-weight, and proprietary multimodal models, using LLM-as-a-judge (Zheng et al., 2023). The result supports that ProMQA-Assembly contains challenging multimodal questions even for strong proprietary models, which lag significantly behind human performance.

Our contributions are threefold: First, we develop a new QA evaluation dataset for multimodal procedural activities. We focus on assembly, which is underexplored in the existing studies. By effectively incorporating fine-grained action labels in the question generation process, we obtained 646 diverse high-quality QA pairs. Second, we propose an annotation approach to create instruction task graphs for the assembly tasks. Our 81 graphs, which are three times larger than prior work, are used in benchmarking and facilitate our QA annotation process. Third, we benchmark existing models to provide baselines and insights for further research on methodologies. The results demonstrate the challenging nature of our dataset and the promising yet insufficient results of reasoning models. We believe that our ProMQA-Assembly enriches the evaluation suite and encourages system development on procedural-activity assistants.[1]

2. Related work

Our work is positioned at the intersection of multimodal QA, procedural activity understanding, and instruction task graphs.

Multimodal QA: Prior work has explored multimodal QA (Duan et al., 2024), including image-centric benchmarks like MMMU (Yue et al., 2024) and video-centric benchmarks, such as Video-MME (Fu et al., 2024) and LVBench (Wang

[1]Code and data are available in https://github.com/kimihiroh/promqa-assembly

Dataset Name	Video+Text Input	Procedural	Assembly	Task Graph	Multi View	QA	Open Vocab	LLM Scoring
Assembly101 (Sener et al., 2022)	✗	✓	✓	✗	✓	✗	✗	✗
IndustReal (Schoonbeek et al., 2024)	✗	✓	✓	✗	✗	✗	✗	✗
CaptainCook4d (Peddi et al., 2024)	✗	✓	✗	✓	✗	✗	✗	✗
EgoSchema (Mangalam et al., 2023)	✓	✗	✗	✗	✗	✓	✗	✗
GazeVQA (Ilaslan et al., 2023)	✓	✓	✓	✗	✓	✓	✗	✗
OpenEQA (Majumdar et al., 2024)	✓	✗	✗	✗	✗	✓	✓	✓
ProMQA (Hasegawa et al., 2025)	✓	✓	✗	✓	✗	✓	✓	✓
ProMQA-Assembly (Ours)	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Our dataset vs relevant datasets

et al., 2024). While the multiple-choice format is popular due to its ease for evaluation, some datasets like OpenEQA (Majumdar et al., 2024) and ProMQA (Hasegawa et al., 2025) adopt open-vocabulary settings. The third-person (i.e., exocentric) view is common in the existing datasets, but the first-person (i.e., egocentric) view has been getting more attention, e.g., EgoSchema (Mangalam et al., 2023), which is based on e.g., Ego4D (Grauman et al., 2022). In ProMQA-Assembly, we target a multimodal QA dataset with both exocentric and egocentric views in an open-vocabulary setting, considering its practical applications.

Procedural activity understanding: Procedural activities are ubiquitous across human contexts: from everyday tasks such as cooking (Kuehne et al., 2014; Stein and McKenna, 2013; Damen et al., 2020; Peddi et al., 2024; Huang et al., 2024) and DIY(Ben-Shabat et al., 2021; Liu et al., 2024; Jang et al., 2019) to industrial settings, e.g., manufacturing (Ragusa et al., 2021; Bansal et al., 2022; Wang et al., 2023a; Schoonbeek et al., 2024), research, e.g., biological experiment (Yagi et al., 2024), or specialized work like medical operations (Beyer-Berjot et al., 2016; Jang et al., 2023). Understanding procedural activities has been explored through instructional videos or recordings of people performing tasks. Popular task formulations include classification tasks such as action recognition and temporal segmentation. While the classification formulation has practical value, applying classifiers to an assistant system is challenging when the assistant is required to provide more fine-grained and context-aware feedback. Recently, mistake detection in procedural activities has been getting more attention (Sener et al., 2022; Peddi et al., 2024; Haneji et al., 2024; Flaborea et al., 2024) due in part to the need for automatic assistants to identify mistakes in procedural activities. To better support the use of procedural activity understanding in assistant systems, our dataset adopts a QA formulation, where outputs are expressed in natural language. In addition, we focus on recordings that contain mistakes in order to assess systems in terms of recognizing errors, tracking task progress, and generating context-aware feedback.

QA	#Pairs	646
	#Answers per question	2.2
	Question Length (word)	11.5
	Answer Length (word)	14.7
Recording	#Unique recordings	227
	Duration (min)	2m2.3s
	#Steps per recording	5.8
Instruction	#Unique instructions	81
	#Steps per instruction	10.4

Table 2: Dataset statistics

Instruction task graph: In parallel to activity understanding, there is a line of work that constructs procedural instructions from videos (Dvornik et al., 2022; Ashutosh et al., 2023; Peddi et al., 2024). The task is to identify the order of steps in a procedure, given a set of videos, and typically represent them as a partial DAG (Schank and Abelson, 1977; Sakaguchi et al., 2021). In our work, we manually created the instruction task graph for Assembly 101 and used it for our QA creation pipeline and in the benchmarking experiments.

3. Dataset

The goal of ProMQA-Assembly is to provide an evaluation testbed to track the progress of multimodal models on procedural activity understanding, especially in the domain of assembly. Table 1 shows the comparison among ProMQA-Assembly and existing datasets. Our dataset features multimodal input, procedural assembly activity, instruction task graphs, multi-view, and open-vocab QA with LLM-as-a-judge. Table 2 shows its statistics.

3.1. Task formulation

ProMQA-Assembly employs QA as its task formulation to reflect real-world use cases while keeping simplicity as an evaluation benchmark. Given an instruction task graph g, a parts image i, a recording v, and a question q, a model is tasked to generate an answer a. The instruction task graph g is a DAG,

Category	Size	Target	Typical Example Question
Process-level			
Next	152	Ask about future steps	*What is the next step now?*
Missing	107	Ask about steps missed in the past	*Did I miss any steps so far?*
Order	60	Ask about step orders in the past	*Was it correct to attach X before this?*
Misadjustment	56	Ask about the correct/incorrect step adjustment	*Should I reattach X?*
Past	51	Ask about any other mistakes made in the past	*Have I made any mistakes?*
Others	22	Ask about process-level other things	N/A (e.g., *Why did I have to remove X?*)
Step-specific			
Location	149	Ask about the location of a part	*Did I attach X in the correct location?*
Others	49	Ask about other things about each step	N/A (e.g., *Should I screw this part now?*)

Table 3: Question categories, the number of examples, target phenomenon, and typical examples.

Aspect	ProMQA	ProMQA-Assembly (Ours)
Main Stats (QA, Instr., Rec)	(401, 24, 231)	(646, 81, 227)
Domain (Source)	Cooking (CaptainCook4D)	Assembly (Assembly 101)
Goal	Not always achieved	Always achieved
Step Dependency	high	relatively low
Step Type	correct, mistake	correct, mistake, correction
Step Annotation	coarse	coarse & fine-grained
Trial & Error	some steps are irreversible	always allowed
Action Type	various (e.g., cut, measure, heat)	attach (detach)
Tool	various (e.g., knife, spoon, pot)	screwdriver
Object Type	rigid/non-rigid/transformable	only rigid
Object Visibility	Ingredients are not always visible	Remaining parts are always visible
Video View	ego-centric	both ego-centric and exo-centric
Video Length (avg.)	6.5 min	2.0 min
Instruction	task graph	parts/final picture & task graph

Table 4: Comparison between the original ProMQA and our new dataset

where a node represents a step s, and an edge represents a step dependency that the two connected nodes need to follow. The recording consists of a video, i.e., a pile of frames, where a user performs a sequence of m steps, $S = \{s_0, s_1, \ldots, s_{m-1}\}$. The parts image shows parts, a target assembly image, and/or an exploded view. The question and answer are written in natural language.

3.2. QA pair

We target multimodal procedural questions, i.e., questions that require the understanding of both instructions and recordings about procedures. The questions in ProMQA-Assembly are grouped into two coarse categories, process-level and step-specific, which are further divided into specific types. Process-level questions orient towards the understanding of each procedure, consisting of *next*, *missing*, *order*, *misadjustment*, and one fallback category, *other*. Step-specific questions focus on each specific step, including *location* and one catch-all category, *other*. Table 3 shows a typical example of each type of question. As for the answer format, natural language (i.e., open vocabulary) is employed to resemble usual dialogue. While existing work often uses the multiple-choice style for its evaluation simplicity, that style poses other challenges, such as difficulty of creating good negative choices (Wang et al., 2023b) and potential biases (Pezeshkpour and Hruschka, 2024). We use English in this work. More details about the QA annotation process are in § 4.

3.3. Instruction task graph

Instruction task graphs are DAGs, where each node represents one step in the instructions, and each edge represents the step dependency one must follow for correct assembly, as illustrated in Figure 1. We chose this representation to accurately define step orders (Peddi et al., 2024). More specifically, a node can be executed once all of its prerequisite nodes (i.e., nodes with incoming edges to it) have been completed. These instruction graphs are used as an input to provide the accurate step order requirements in our benchmarking and QA annotations. Annotation process is detailed in § 5.

3.4. Assembly Activity

Our dataset consists of the activity of assembling take-apart toys. We chose the activity because it possesses fundamental characteristics for assembly. Toy assembly shares commonalities with other complex scenarios of DIY (e.g., flatpack furniture, camping tent) and industrial settings (e.g., automotive, construction): steps with order dependencies, parts that are rigid and decomposable, a general reversibility of steps, and use of spe-

cific tools, like screwdrivers. We believe that these similarities, despite any other differences, make ProMQA-Assembly a meaningful initial dataset for the community to explore assembly assistants.

3.5. Comparison with ProMQA

While ProMQA-Assembly resembles ProMQA in task formulation and QA annotation approach, there are notable differences as shown in Table 4. Specifically, ProMQA-Assembly contains multiple forms of instructions, i.e., parts image and task graphs, and multiple angles of the same videos. The fundamental difference in the activity domains (cooking vs assembly) necessitates the difference in types of actions and their target objects. Another distinctive difference is the correction actions, which are not included in ProMQA, partially because of the irreversible nature of the cooking activity, as well as the experimental design choice. As a proxy for step dependency complexity, we calculate the percentage of edges between step nodes, the nodes that are not "START" or "END." The ratio is 0.6 for ProMQA-Assembly and 0.8 for ProMQA. Additionally, there are two more differences from the QA task perspective. The first concerns multimodality strictness, where the extent of instruction visibility is different. While the videos in both datasets are sufficient sources for instructions, the parts images in ProMQA-Assembly are more visible compared to the textual step descriptions in ProMQA. The second difference from the QA task perspective regards the importance of step histories, especially the last frame(s), due to the types of objects being recorded and the visibility of the components. In ProMQA, the remaining cooking ingredients and the ingredients already added are not always captured in the last frame(s), e.g., sugar melts and becomes invisible. In contrast, the assembled parts and the remaining parts are more visible in the last frames(s) in ProMQA-Assembly.

In short, ProMQA-Assembly is based on simpler activities, while it has more variations in terms of video views and instructions. However, this simplicity is mainly from a human perspective, and it is unclear if the same applies to systems. We believe that this additional domain, along with the variations that ProMQA-Assembly provides, can facilitate multi-faceted analysis to understand the capabilities of multimodal models.

4. QA annotation

We employ a semi-automatic approach for our QA annotation to ensure cost-effectiveness while maintaining quality.

4.1. Preprocess

As the source of assembly recordings, we select Assembly 101 (Sener et al., 2022), containing 362 unique recordings of users assembling take-apart toys. In the recordings, users occasionally make mistakes and need to disassemble and reassemble the parts. This "*natural*" behavior makes the dataset suitable for our online-style QA dataset. We first filter the recordings based on the annotation completeness. Among 362 sequences, 244 sequences contain all annotations (mistake labels, coarse action labels, and fine-grained action labels) and all angles of videos (12 views). We also manually corrected some coarse action labels based on our manual inspections.

Next we create candidate examples by cutting the original recordings based on the action temporal segmentation labels. Suppose we have a sequence with n steps, $S = \{s_0, s_1, \ldots, s_{n-1}\}$. From S we create n video segments $v_{0:k} = \{s_0, s_1, \ldots, s_k\}$, where $k \in \{1, 2, \ldots, n\}$. With each video segment we create eight examples by attaching a different target question type, t, for prompting. The target question types reflect the four existing mistake annotations in Assembly 101, including ***order*** for the wrong order, ***past*** for when a previous step is a mistake, ***misadjustment*** for a step that should not have happened, and ***location*** for a part in the wrong position. Additionally, we added ***next*** for asking about next steps and ***missing*** for asking about any missing steps, as well as ***others(process)*** and ***others(step)*** for general questions to increase the diversity of the questions. Together with our instruction task graph g (§ 5) and parts/target assembly image i from Assembly101, one example consists of $\langle v_{0:k}, g, i, t \rangle$. Through this process, we obtained around 22k examples in total.

Due to the time-intensive nature of manual verification, we sampled examples based on two criteria. First, we sample at most two examples for each target question type from one unique sequence. Second, for the question types of *order*, *past*, *misadjustment*, and *location*, we prioritize picking the video segments where their respective mistake label is attached to the last action of each video segment. These conditions help us make use of as many sequences in the original data as possible and generate more relevant questions. For all 8 question types, we sampled at most 100 examples for each type under the conditions, which resulted in 794 examples in total.

4.2. QA generation

We used LLMs to generate candidate question-answer pairs to reduce annotation costs, following prior work (Mangalam et al., 2023; Hasegawa et al., 2025). As a starting point, we first evaluated

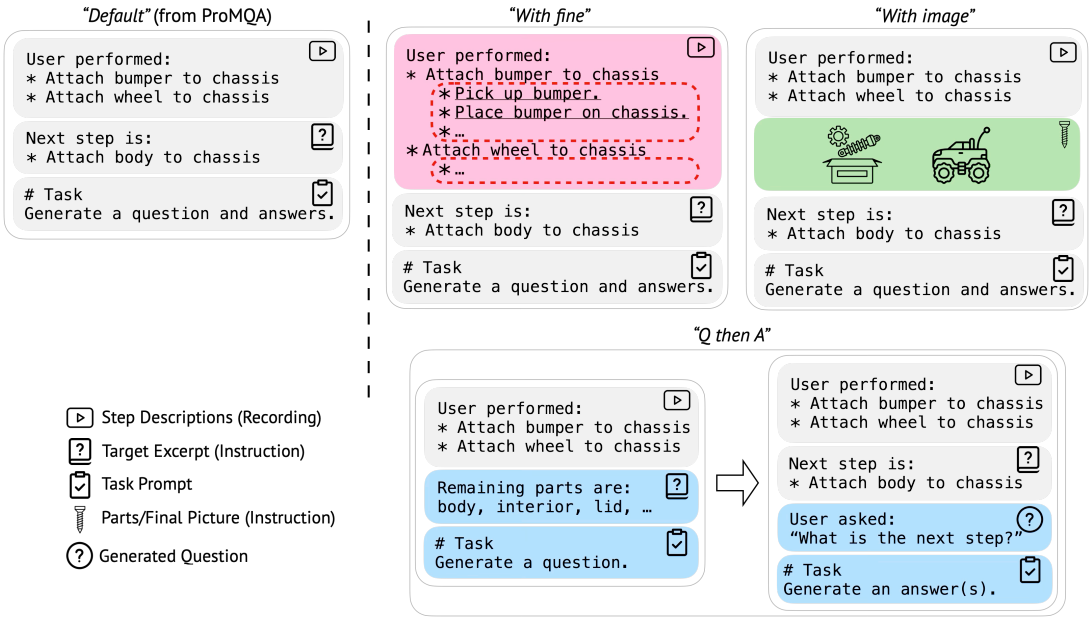


Figure 2: QA generation prompts

Prompt				Generator		
Default	Q then A	w/ img	w/ fine	Gemini	Claude	o3-mini
0.78	0.63	0.39	0.78	0.67	0.52	0.50

Table 5: Prompt & QA generator comparison. The number is the percentage of multimodal procedural questions among all generated questions.

whether ProMQA's optimal prompt was effective for our assembly domain; in addition, we tested new prompt variants under the controlled experimental settings. We also compared four candidate QA generators. All generated QA pairs underwent human verification to ensure quality (§ 4.3).

Prompt comparison: Figure 2 shows the abridged versions of prompt templates including *"Default"*, *"With fine"*, *"With image"*, and *"Q then A"*. *"Default"* represents ProMQA's optimal prompt template, which mainly consists of three components: (1) step descriptions: textual descriptions of steps already performed to inform LLMs of the context of the questions to be generated, (2) target excerpt: an answer hint, extracted from instructions or mistake annotations, and (3) task prompt: including a target question type to be generated. New prompt variants change one aspect at a time under the controlled experiment settings. *"With fine"* is a prompt template where fine-grained step descriptions and fastening information are added on top of the existing coarse step descriptions. *"With image"* is the prompt where we feed a parts/target assembly image as additional input to LLMs. *"Q then A"* represents two-step prompting where questions are first generated, followed by answer generation, which tries to direct LLMs to focus on each step one at a time. Using 46 samples, we compared these four prompt templates by generating question-answer pairs with one fixed LLM and manually checked them. Table 5 shows the rate of our target multimodal procedural question (criteria in § 4.3) out of all samples. According to the result, *"Default"* and *"With fine"* generate our target questions the most. By comparing them qualitatively, we found that the *"With fine"* template successfully integrated the fine-grained action information in the generated questions such as "What should I do after putting down the partial toy?" and "Do I need to put a screw on this part?", while generating coarse action-level questions as well. Considering that the *"Default"* generates only coarse action-level questions, we chose the *"With fine"* template in this study.

Model selection: In addition to the default GPT-4o (OpenAI, 2024), we considered the following models as candidate QA generators: Gemini 2.5 Pro (Google, 2025), Claude 3.7 Sonnet (Anthropic, 2025), and o3-mini (OpenAI, 2025b). Based on the experimental results, shown in Table 5, we decided to use GPT-4o as our QA generator.

Manual QA creation: In addition to automatic QA generation, we also ask a human annotator to create QA pairs for a small set of examples (50 examples) as a comparison. The annotation took more than 12 times longer than our automatic generation (45 mins vs 3.5 mins). These human-created QA pairs were merged with machine-generated ones and underwent human verification.

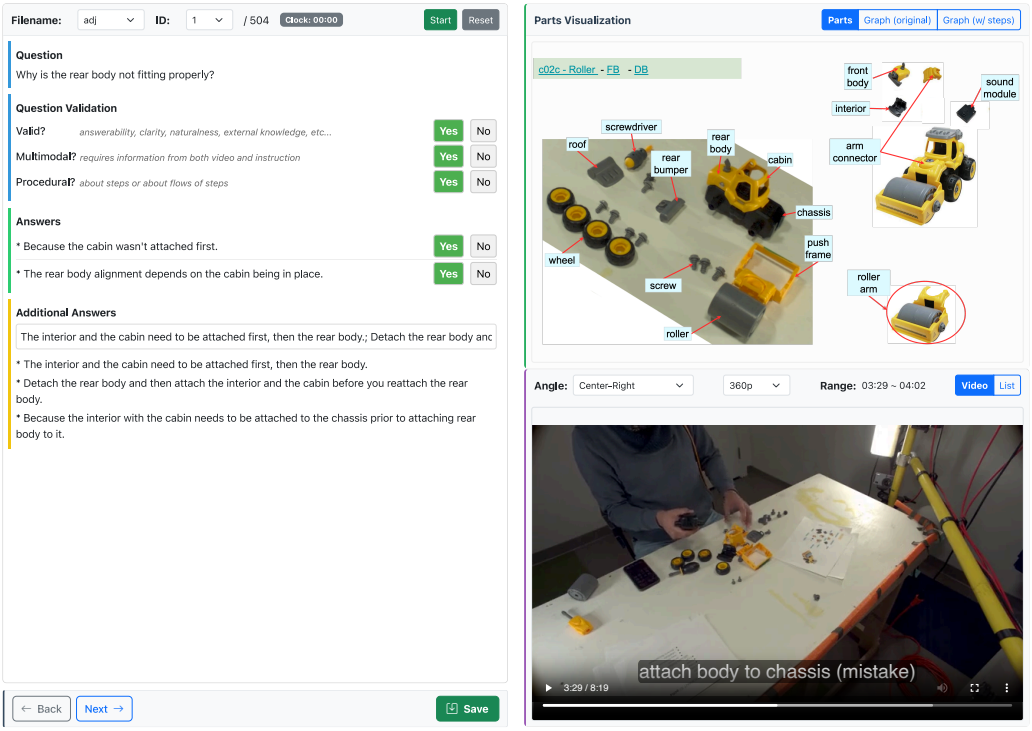


Figure 3: QA verification interface. An annotator verifies the question and answers (left panel) based on the instruction (upper right) and the recording (lower right). If needed, they write answers in the comment.

4.3. QA verification

We performed human verification on ProMQA-Assembly to ensure its quality as an evaluation dataset. Our choice prioritizes quality over quantity, unlike automatic methods which tend to prioritize quantity over quality. Human annotators check each QA example by referring to its corresponding instructions (both task graph and parts image) and video recording. Questions were verified based on three aspects: (1) **Valid**: whether or not a question is answerable and makes sense. If not, annotators can skip to the next example, (2) **Multimodal**: whether or not a question requires information from both its instructions and recording, and (3) **Procedural**: whether or not a question is about a step or a sequence of steps. Answers were simply checked to see whether they were correct or not. When candidate answers were incorrect or incomplete, annotators provided the correct/missing answer(s). For each example, we assign two annotators to independently verify each QA example. When two annotations conflict in any of the aspects, another annotator acts as an adjudicator to resolve any annotation conflicts. After a trial annotation, five annotators and one adjudicator performed the verification task using the annotation guideline developed during the trial. All annotators and the adjudicator have CS-related graduate degrees. We calculated the inter-annotator agreement (IAA) based on the percent agreement: 0.75 for the valid aspect, 0.83 for the multimodal aspect, 0.96 for the procedural aspect, and 0.77 for the answer correctness. After the adjudication, we were left with 646 valid questions, of which 538 were multimodal procedural questions, and 47 were human-created questions.

5. Instruction task graph annotation

In addition to QA pairs, we annotated instruction task graphs for the toys in Assembly101 motivated by three reasons. First, the task graphs represent accurate step order requirements, which are essential in benchmarking experiments. Second, the graphs are used in our QA generation process (§ 4.2), specifically to provide answer hints as target excerpts. Third, they can facilitate the human verification process. During our preliminary QA verification, we found the task challenging if we could only watch the recording and look at the parts' image for the question, especially when recall is important. For instance, "What is the next step now?" should have all possible next steps as answers, and even humans may miss listing all the next steps if they are unable to refer to a verified task graph.

5.1. Annotation procedure

Figure 4 shows the annotation interface. The initial sets of nodes were collected based on the coarse action labels in Assembly101, with two extra nodes,

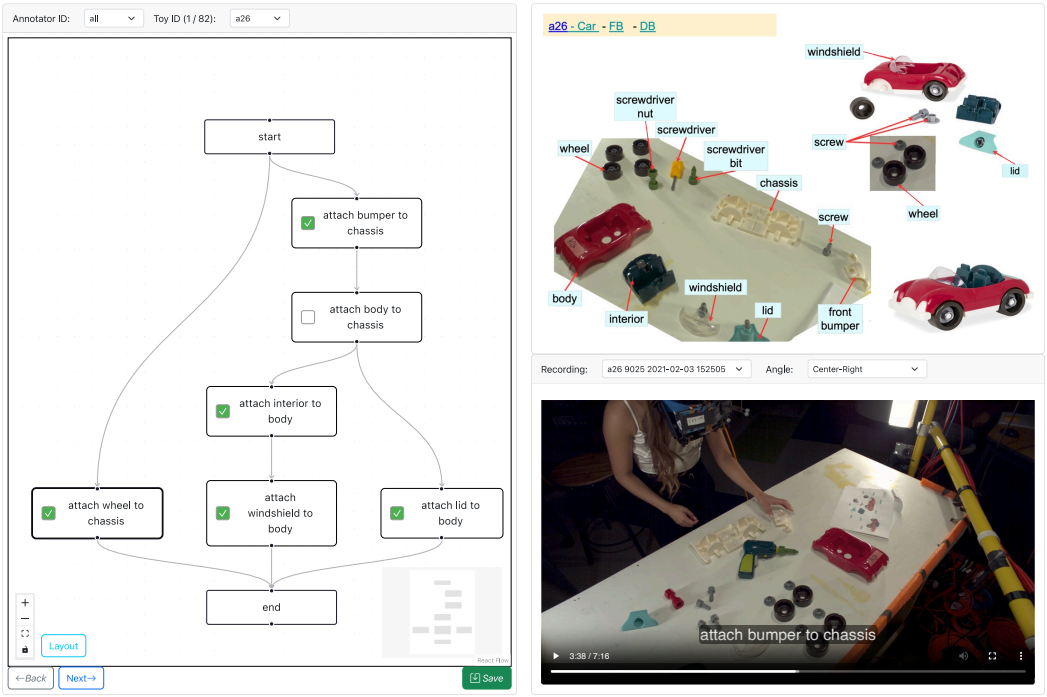


Figure 4: Task graph annotation interface. An annotator checks the part image (upper right) and the recordings (lower right) to identify the step dependencies and add edges by dragging on the graph annotation interface (left).

"START" and "END," for each graph. Annotators can watch a set of recordings of multiple users assembling the same toy, often in different orders.

Given the pre-loaded nodes and a set of recordings, annotators add edges between nodes by identifying a specific order of steps to assemble the toy correctly. Additionally, we ask annotators to check if a step requires a fastening (i.e., a screw) to provide further enriched step information in QA generation. We assign two people to annotate each graph independently, and if any conflicts happen, an adjudicator resolves them. Five people participated as annotators, and one person served as an adjudicator. In total, we annotated 81 instruction task graphs, which is three times more than the task graphs in CaptainCook4D. To measure IAA, we calculated pairwise F1 scores (UzZaman and Allen, 2011) for the edge labeling and percent agreements for fastening checks. The average IAA for edges is 0.74, and that for fastening is 0.89. While these manually curated instruction task graphs are used as inputs in this study, they can be used to evaluate systems for task graph construction as well (Seminara et al., 2024). We encourage the researcher to test their graph construction systems on our 81 task graphs, in addition to the 24 graphs in CaptainCook4D.

6. Benchmarking

We provide benchmarking results on ProMQA-Assembly to give insights to the research community on how well existing models perform.

6.1. Experimental Setup

We consider three types of models. One is a text-only model, where we feed only an instruction task graph and a question without any vision inputs to obtain the lower bound for multimodal-input models. We tested Llama 3.3 (72B) (Grattafiori et al., 2024) and DeepSeek-R1 (DeepSeek-AI, 2025). For the DeepSeek-R1, we used the distilled version with Qwen (32B), due to our computational resources. Next, we chose open(-weight) multimodal models including Qwen2.5-VL (72B) (Bai et al., 2025), a smaller version of InternVL3 (8B) (Zhu et al., 2025), and LLaVA-Video (72B) (Zhang et al., 2024). Finally, we also benchmarked proprietary multimodal models, namely, GPT-4o (OpenAI, 2024), GPT-5 (OpenAI, 2025a), Gemini 2.5 Pro (Google, 2025), and Claude 3.7 Sonnet (Anthropic, 2025), with zero-shot chain-of-thought (*CoT*) (Kojima et al., 2022) or their reasoning modes (*think*). Additionally, we evaluated some multimodal models under the text-only setting to highlight the importance of the vision and multimodal inputs. As a comparison, we also obtained human performance by asking three of the annotation participants to answer questions that they had not checked in the verification stage. Due to cost constraints, we used only a subset of 20 different questions out of the full 646 questions.

The outputs of all open-weight models are ob-

Model	All	Multi&Proc		Process-level						Step-specific		Source	
		Yes	No	next	miss	order	misadj	past	others	loc	others	ma	hum
Text-only													
Llama 3.3 72B	30.6	27.6	45.4	39.5	36.0	43.3	29.5	29.4	27.3	13.4	31.6	30.5	31.9
DeepSeek R1 (Qwen 32B)	44.0	39.6	65.7	47.0	51.9	60.8	42.9	30.4	27.3	34.6	48.0	44.1	42.6
Qwen2.5-VL	39.7	37.5	50.9	48.0	57.5	42.5	39.3	31.4	31.8	19.1	46.9	39.4	43.6
GPT-4o	34.4	31.1	50.9	40.5	46.7	49.2	33.9	29.4	22.7	15.1	40.8	34.7	30.9
GPT-5 w/ think	47.1	42.1	72.2	49.7	63.1	65.0	44.6	33.3	31.8	29.2	61.2	47.2	46.8
Claude 3.7 Sonnet	36.4	31.5	60.6	45.1	45.3	50.0	38.4	37.3	29.5	13.8	41.8	36.3	37.2
Gemini 2.5 Pro	36.3	30.8	63.9	45.7	47.7	62.5	31.2	21.6	25.0	15.8	38.8	35.9	41.5
Multimodal													
Qwen2.5-VL 72B Instruct	45.4	44.4	50.0	46.1	53.7	40.0	42.9	39.2	38.6	43.0	51.0	44.1	61.7
LLaVA-Video 72B (Qwen2)	47.4	46.5	51.9	47.0	57.5	31.7	40.2	50.0	31.8	49.3	52.0	47.1	51.1
InternVL3 8B	38.2	37.2	43.5	38.5	27.6	41.7	34.8	36.3	25.0	45.0	48.0	37.7	44.7
GPT-4o	47.8	46.5	54.2	44.1	43.5	59.2	45.5	49.0	36.4	50.3	53.1	48.1	43.6
GPT-4o w/ CoT	45.8	41.1	69.4	48.0	41.6	65.8	42.9	37.3	38.6	41.9	51.0	45.2	54.3
GPT-5	48.2	46.3	57.9	48.4	41.1	56.7	41.1	44.1	34.1	53.4	56.1	47.7	55.3
GPT-5 w/ think	**58.0**	**53.2**	**81.9**	53.3	**65.9**	**72.5**	48.2	48.0	38.6	**55.7**	**74.5**	**58.2**	55.3
Claude 3.7 Sonnet	47.8	45.2	60.1	52.9	43.5	63.1	36.0	**55.8**	32.6	42.0	45.0	47.4	51.0
Claude 3.7 Sonnet w/ CoT	48.4	43.8	71.3	51.3	52.3	65.8	40.2	49.0	31.8	39.9	51.0	48.2	50.0
Claude 3.7 Sonnet w/ think	48.5	44.1	69.9	53.3	51.9	66.7	39.3	46.1	29.5	40.9	48.0	47.9	55.3
Gemini 2.5 Pro	51.0	45.4	78.7	56.6	59.3	66.7	39.3	34.3	18.2	46.3	56.1	50.1	62.8
Gemini 2.5 Pro w/ CoT	54.2	49.5	77.3	**58.9**	60.7	**72.5**	**49.1**	42.2	**40.9**	44.0	58.2	53.3	**66.0**
Human*	(70.7)	—	—	—	—	—	—	—	—	—	—	—	—

Table 6: Benchmarking result. "Multi&Proc" represents multimodal procedural questions. Categories under "Process-level" and "Step-specific" are question types. "Source" shows the creators of questions, either machine ("ma") or human ("hum"). All numbers are averages over examples for each category. *: Human performance is based on the sampled 20 examples.

tained using at most four A6000 GPUs. We sample 20 frames uniformly from each recording from the above angle with resizing to 640x360, and feed them with a corresponding text prompt and a parts image. Task graphs are fed in DOT format. For evaluation, we use LLM-as-a-judge (Zheng et al., 2023) with GPT-4o, following ProMQA. The evaluator provides scores in a three-point Likert scale, and we scale them to 0-100 by multiplying by 50.

6.2. Result

Table 6 shows the benchmarking result. Based on this result, we focused on two points: the confirmation of our dataset's value and a promising approach with room for improvement. As for the former point, models perform consistently better under the multimodal setting than under the text-only setting, which supports the necessity of visual understanding for ProMQA-Assembly. For instance, the same proprietary models perform even more than 10 points better with visual inputs compared to text-only settings (51.0 with multimodal vs 36.3 with text-only settings for Gemini 2.5 Pro). In addition, the general performance gap between multimodal procedural understanding questions (*Yes*) and other valid questions (*No*) strengthens the value of the challenge that our dataset poses. Specifically, GPT-5 (*w/ think*) answers our valid, yet non multimodal-procedural questions (81.9) much easily, compared to our target questions (53.2). Concerning the second point, models with reasoning capability stand out in each setting, i.e., DeepSeek-R1, GPT-5 (*w/ think*), and Gemini 2.5 Pro (*w/ CoT*), among models with mostly comparable performance. This indicates the potential of inference-time scaling methods and reasoning-oriented training. While GPT-5 (*w/ think*) shows the best performance amongst the ones we tested, it still lags far behind the reference human performance. Our benchmark experiments show promising results of reasoning techniques, and we would encourage the research community to explore more to further advance the multimodal procedural activity understanding by utilizing our ProMQA-Assembly dataset.

7. Conclusion

In this work, we propose a new multimodal QA evaluation dataset, ProMQA-Assembly, which specifically targets assembly tasks. During the development phase, we annotate QA pairs efficiently using the combination of LLM generation and human verification, and we also create instruction task graphs. Our benchmarking results show that current multimodal models underperform humans, which suggests great room for improvement. We believe that our benchmark dataset sheds light on the missing capabilities of multimodal models and facilitates the development of further helpful systems for humans.

8. Ethical Statement

In our dataset development we used LLMs, which are pretrained on a massive web-scraped corpora. Considering that LLMs may introduce prejudiced, offensive, or biased content in our dataset, we carefully checked potentially inappropriate questions and answers in our verification process.

9. Limitations

We acknowledge several limitations of our work. First, the size of our dataset is relatively small. This is based on our design choice of quality over quantity, due to the careful curation by the human verification. While our dataset can be reliably used to evaluate models, which is the primary purpose of this dataset, when it comes to training models, one may need to resort to automatic verification with LLMs. Since our approach produces 22k examples in total at the preprocessing time, effective automatic verification has the potential to scale the dataset.

The second point concerns real-world applicability due to the target objects of take-apart toys. Take-apart toy assembly may be simpler for humans than other practical assembly tasks. However, as we discussed in § 3.4, take-apart toys possess fundamentally common characteristics with broader assembly tasks. Hence, the findings based on our dataset, together with our annotation approach, can be generalized to broader scenarios. At the same time, developing QA datasets on higher complexity tasks is beneficial for further practicality, although few resources in the current research community contain error-included real recordings of assembly tasks. Furthermore, much greater variability exists in real-world use cases, e.g., insufficient light, vision obstacles, third-person interactions, evolving to a dialog-style setting, and/or replacing the task graphs with normal assembly instructions, which requires the identification of implicit step dependencies (Lal et al., 2024). We envision our work encouraging the community to invest in resource development on higher-complexity assembly tasks in more practical environments.

Third, the dataset contains only English. With the increase of multilingual multimodal models, the English capability on procedural activity understanding can be indicative of the capabilities with other languages on similar tasks. Yet, considering the ubiquitous demand for assembly assistant systems, we hope to contribute to developing datasets in non-English languages. Finally, we note that our dataset is released for evaluation purposes only, not for training, in compliance with OpenAI's terms of use.[2]

[2]https://openai.com/policies/row-terms-of-use/

10. Acknowledgment

This work is partially supported by (1) Programs for Bridging the gap between R&D and the IDeal society (society 5.0) and Generating Economic and social value (BRIDGE) / Practical Global Research in the AI × Robotics Services, implemented by the Cabinet Office, Government of Japan, and (2) a project, JPNP25006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

11. Bibliographical References

Anthropic. 2025. Claude 3.7 sonnet and claude code.

Kumar Ashutosh, Santhosh Kumar Ramakrishnan, Triantafyllos Afouras, and Kristen Grauman. 2023. Video-mined task graphs for keystep recognition in instructional videos. *Advances in Neural Information Processing Systems*, 36:67833–67846.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Siddhant Bansal, Chetan Arora, and CV Jawahar. 2022. My view is the best view: Procedure learning from egocentric videos. In *European Conference on Computer Vision*, pages 657–675. Springer.

Yizhak Ben-Shabat, Xin Yu, Fatemeh Saleh, Dylan Campbell, Cristian Rodriguez-Opazo, Hongdong Li, and Stephen Gould. 2021. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 847–859.

Laura Beyer-Berjot, Stéphane Berdah, Daniel A Hashimoto, Ara Darzi, and Rajesh Aggarwal. 2016. A virtual reality training curriculum for laparoscopic colorectal surgery. *Journal of surgical education*, 73(6):932–941.

Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro,

Toby Perrett, Will Price, et al. 2020. The epic-kitchens dataset: Collection, challenges and baselines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):4125–4141.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Guodong Ding, Fadime Sener, and Angela Yao. 2022. Temporal action segmentation: An analysis of modern techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:1011–1030.

Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. 2024. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201.

Nikita Dvornik, Isma Hadji, Hai Pham, Dhaivat Bhatt, Brais Martinez, Afsaneh Fazly, and Allan D Jepson. 2022. Flow graph to video grounding for weakly-supervised multi-step localization. In *European Conference on Computer Vision*, pages 319–335. Springer.

Alessandro Flaborea, Guido Maria D'Amely di Melendugno, Leonardo Plini, Luca Scofano, Edoardo De Matteis, Antonino Furnari, Giovanni Maria Farinella, and Fabio Galasso. 2024. Prego: Online mistake detection in procedural egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18483–18492.

Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. 2024. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*.

Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. Question answering is a format; when is it useful? *arXiv preprint arXiv:1909.11291*.

Google. 2025. Gemini 2.5: Our most intelligent ai model.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. 2022. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012.

Yuto Haneji, Taichi Nishimura, Hirotaka Kameko, Keisuke Shirai, Tomoya Yoshida, Keiya Kajimura, Koki Yamamoto, Taiyu Cui, Tomohiro Nishimoto, and Shinsuke Mori. 2024. Egooops: A dataset for mistake action detection from egocentric videos with procedural texts.

Kimihiro Hasegawa, Wiradee Imrattanatrai, Zhi-Qi Cheng, Masaki Asada, Susan Holm, Yuran Wang, Ken Fukuda, and Teruko Mitamura. 2025. ProMQA: Question answering dataset for multimodal procedural activity understanding. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11598–11617, Albuquerque, New Mexico. Association for Computational Linguistics.

Yifei Huang, Guo Chen, Jilan Xu, Mingfang Zhang, Lijin Yang, Baoqi Pei, Hongjie Zhang, Lu Dong, Yali Wang, Limin Wang, et al. 2024. Egoexolearn: A dataset for bridging asynchronous ego-and exo-centric view of procedural activities in real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22072–22086.

Muhammet Ilaslan, Chenan Song, Joya Chen, Difei Gao, Weixian Lei, Qianli Xu, Joo Lim, and Mike Shou. 2023. GazeVQA: A video question answering dataset for multiview eye-gaze task-oriented collaborations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10462–10479, Singapore. Association for Computational Linguistics.

Youngkyoon Jang, Brian Sullivan, Casimir Ludwig, Iain Gilchrist, Dima Damen, and Walterio Mayol-Cuevas. 2019. Epic-tent: An egocentric video dataset for camping tent assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*.

Yunseok Jang, Sungryull Sohn, Lajanugen Logeswaran, Tiange Luo, Moontae Lee, and Honglak Lee. 2023. Multimodal subtask graph generation from instructional videos. *arXiv preprint arXiv:2302.08672*.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Hilde Kuehne, Ali Arslan, and Thomas Serre. 2014. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yash Kumar Lal, Vanya Cohen, Nathanael Chambers, Niranjan Balasubramanian, and Ray Mooney. 2024. CaT-bench: Benchmarking language model understanding of causal and temporal dependencies in plans. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19336–19354, Miami, Florida, USA. Association for Computational Linguistics.

Sicong Leng, Hang Zhang, Guanzheng Chen, Xin Li, Shijian Lu, Chunyan Miao, and Lidong Bing. 2024. Mitigating object hallucinations in large vision-language models through visual contrastive decoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13872–13882.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Yunong Liu, Cristobal Eyzaguirre, Manling Li, Shubh Khanna, Juan Carlos Niebles, Vineeth Ravi, Saumitra Mishra, Weiyu Liu, and Jiajun Wu. 2024. IKEA manuals at work: 4d grounding of assembly instructions on internet videos. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Sasha Sax, and Aravind Rajeswaran. 2024. Openeqa: Embodied question answering in the era of foundation models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. 2023. Egoschema: A diagnostic benchmark for very long-form video language understanding. In *Advances in Neural Information Processing Systems*, volume 36, pages 46212–46244. Curran Associates, Inc.

Kosuke Moriwaki, Gaku Nakano, and Tetsuo Inoshita. 2022. The brio-ta dataset: Understanding anomalous assembly process in manufacturing. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1991–1995. IEEE.

OpenAI. 2024. Hello gpt-4o.

OpenAI. 2025a. Introducing gpt-5.

OpenAI. 2025b. Openai o3-mini.

Arjun Panickssery, Samuel Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802.

Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Bhavya Gouripeddi, Qifan Zhang, Jikai Wang, Vasundhara Komaragiri, Eric Ragan, Nicholas Ruozzi, Yu Xiang, and Vibhav Gogate. 2024. Captain-cook4d: A dataset for understanding errors in procedural activities. In *Advances in Neural Information Processing Systems*, volume 37, pages 135626–135679. Curran Associates, Inc.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.

Francesco Ragusa, Antonino Furnari, Salvatore Livatino, and Giovanni Maria Farinella. 2021. The meccano dataset: Understanding human-object interactions from egocentric videos in an industrial-like domain. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1569–1578.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension. *ACM Comput. Surv.*, 55(10).

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proScript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Roger C Schank and Robert P Abelson. 1977. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum.

Tim J Schoonbeek, Tim Houben, Hans Onvlee, Fons van der Sommen, et al. 2024. Industreal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 4365–4374.

Luigi Seminara, Giovanni Maria Farinella, and Antonino Furnari. 2024. Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. 2022. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21096–21106.

Sebastian Stein and Stephen J. McKenna. 2013. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, page 729–738, New York, NY, USA. Association for Computing Machinery.

Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. 2019. Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Naushad UzZaman and James Allen. 2011. Temporal evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 351–356, Portland, Oregon, USA. Association for Computational Linguistics.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Shiyu Huang, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. 2024. Lvbench: An extreme long video understanding benchmark.

Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, Neel Joshi, and Marc Pollefeys. 2023a. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20270–20281.

Zhecan Wang, Long Chen, Haoxuan You, Keyang Xu, Yicheng He, Wenhao Li, Noel Codella, Kai-Wei Chang, and Shih-Fu Chang. 2023b. Dataset bias mitigation in multiple-choice visual question answering and beyond. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8598–8617, Singapore. Association for Computational Linguistics.

Takuma Yagi, Misaki Ohashi, Yifei Huang, Ryosuke Furuta, Shungo Adachi, Toutai Mitsuyama, and Yoichi Sato. 2024. Finebio: a fine-grained video dataset of biological experiments with hierarchical annotation. *arXiv preprint arXiv:2402.00293*.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9556–9567.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. 2024. Video instruction tuning with synthetic data.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, et al. 2025. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*.

A. QA annotation

A.1. Preprocess

To make full use of fine-grained action labels, the preprocessing for *"With fine"* is slightly different from others in terms of video cutting points. For *"With fine"*, we first cut a recording into video segments based on coarse actions, then we slightly move the end timestamp backward based on the fine action labels to add diversity in end points. Specifically, suppose the latest step s_k in a video segment $v_{0:k}$ contains l fine-grained steps, $s_k = \{s_{k,0}, \ldots, s_{k,l-1}\}$. We randomly choose one fine-grained step $s_{k,j}$, and move the end timestamp.

As for manual coarse-action correction, one annotator manually skimmed through and corrected the labels. Representative modifications include typos, normalizing steps descriptions (e.g., "attach interior to chassis" in one recording and "attach chassis to interior" in another recording for the same toy), and misspecification (e.g., "attach body to base," where there are "front body" and "rear body"). On average, 1-2 corrections were made for each toy. We release the modifications, in addition to other data, for reproducibility.

A.2. QA generation

In our prompt comparison experiment, we compared *"Default"*, *"With fine"*, *"With image"*, and *"Q then A"*. The full prompt examples of *"Default"*, *"With fine"*, and *"With image"* are shown in Figure 6, 7, and 8, while Figure 9 and 10 present the full prompt examples for generating questions and answers for *"Q then A"*, respectively.

In the qualitative comparison between *"Default"* and *"With fine"*, we found that *"With fine"* generated 25 questions (out of 46) that mentioned fine-grained step information, while 2 questions by *"Default"* mentioned fine-level step information. Furthermore, we calculated the distinct-1, i.e., type-token ratio, and distinct-2 (Li et al., 2016) to measure the lexical diversity. We first removed stop words and calculated these scores. The *"Default"* showed distinct-1 and distinct-2 scores of 0.30 and 0.77, respectively, while *"With fine"* improved these metrics to 0.36 and 0.80.

Our manual investigation of the *"Q then A"* results revealed that this template generated "should" questions more frequently than the above two templates. These questions tend to require only the instruction information in most cases (e.g., "Should I attach the roof before the light?"). While the template may allow a model to focus on each stage of generation one by one, knowing the target answer in the question generation stage may help a model to generate a question that requires both what has been done and what is supposed to be done to answer. Additionally, we found that *"Q then A"* generated more incorrect answers, which would increase the burden on annotators. One typical reason was because a model generated a question that was different from the specified type in its prompt, hence, the provided target excerpt, i.e., answer hint, became noise for the answer generation. Suppose a prompt type is "missing," where "attach body to chassis" needs to be done before the latest action "attach interior to chassis." In this case, the target excerpt is the missing steps, "attach body to chassis." However, if the generated question is about next steps, e.g., "What should I do next?", the target excerpt misleads the answer generation. In this case, the correct answer would be "You should detach the interior first and attach the body first before the interior," but the model is likely to generate "The next step is to attach the body to the chassis." Moreover, in the *"Q then A"* setting, a model often generated both yes and no answers for the same question and/or paraphrases of the same answers, presumably because the model tried to increase the recall. As we keep only the correct answers, this recall-oriented behavior increases the workload in the verification stage.

As for *"With image"*, we suspect that the input image is distracting for a model, rather than helpful, possibly due to its tendency to draw high attention to irrelevant visual tokens (Leng et al., 2024).

In our preliminary experiments, we used the following versions of the models: GPT-4o (`gpt-4o-2024-11-20`), Gemini 2.5 Pro (`gemini-2.5-pro-exp-03-25`), Claude 3.7 Sonnet (`claude-3-7-sonnet-20250219`), and o3-mini (`o3-mini-2025-01-31`).

In the manual QA creation, we provide a similar interface to that for QA verification, specifically, a recording with a textual action list and a part image. The annotator writes a question, its answer(s), and the timestamp for the question to be asked. The annotation guideline for the manual QA creation contains the interface description and the explanation of our target multimodal procedural questions, which are similar to what a model receives in QA generation. One difference is that we did not specify question types so that the annotator can decide on a question type depending on the context.

A.3. QA verification

Figure 5 shows another view of our QA verification interface: the task graph with step information and the textual list.

Considering that the human annotator who created QAs manually is one of the annotators in the verification stage, we split the manually-created QAs among the annotators who did not create QAs.

We select the adjudicator among the annotators based on their familiarity with the task. The ad-

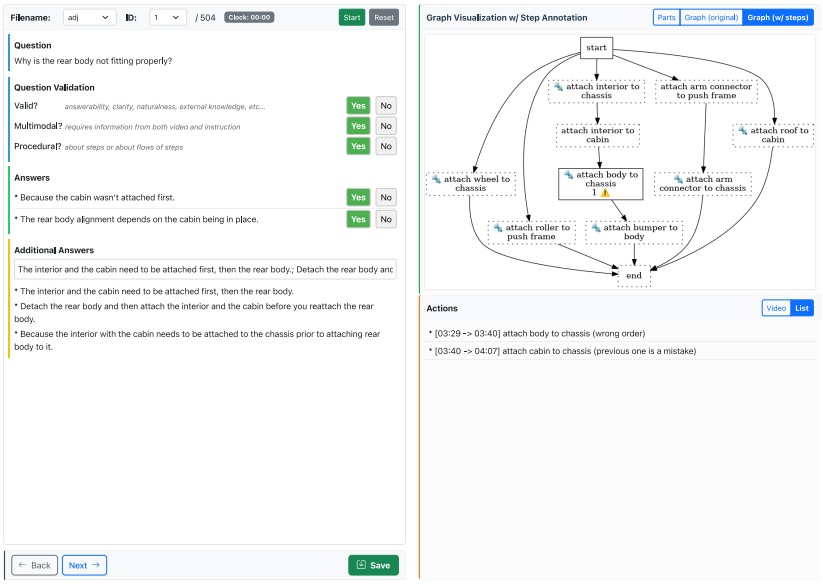


Figure 5: The different view of QA verification interface.

judicator is the one who participated in the development of our guideline. All annotators read the guideline first and annotate 10 examples, followed by our feedback, before completing the remaining ones.

Among the 646 questions, 599 were generated by LLMs, and 47 were created by the human annotator. IAA for "Multimodal," "Procedural," and answers are calculated only for questions that are marked as valid by two annotators.

B. Instruction task graph annotation

The IAA for edges is calculated as a pairwise F1 score. The recall and the precision are defined as follows (UzZaman and Allen, 2011):

$$\text{Precision} = \frac{|E_1 \cap E_2|}{|E_1|}, \quad \text{Recall} = \frac{|E_1 \cap E_2|}{|E_2|}$$

, where E_1 and E_2 denote the sets of edges from two independently annotated task graphs. The guideline for the task graph annotation is also included in our supplemental material.

Four of the five annotators were the same for both the task-graph annotation and QA verification, with one unique annotator assigned to each task, due to their availability.

C. Benchmarking

C.1. Experimental setup

In our benchmarking experiment, we used the following versions of the models: DeepSeek R1 (`deepseek-ai/DeepSeek-R1-Distill-Qwen-32B`), Qwen2.5-VL (`Qwen/Qwen2.5-VL-72B-Instruct`), LLaVA-Video (`lmms-lab/LLaVA-Video-72B-Qwen2`), InternVL3 (`OpenGVLab/InternVL3-8B`), GPT-4o (`gpt-4o-2024-11-20`), GPT-5 (`gpt-5-2025-08-07`), Claude 3.7 Sonnet (`claude-3-7-sonnet-20250219`), and Gemini 2.0 Flash (`gemini-2.0-flash-001`). Proprietary models are based on their API services, and open-weight models are downloaded from the HuggingFace Hub[3] and run locally. For the reasoning mode, we use `medium` for GPT-5 and 4098 as budget tokens for Claude 3.7 Sonnet. For the DeepSeek model, we used the temperature 0.6 based on the recommendation by the official repository.

We feed task graphs as text by representing them in DOT format.[4] In the frame sampling, we sampled uniformly from the last so that the last frame is always included. This is based on our insight that the last frame typically contains informative information for the QA task. Figure 11 shows the basic example of the prompt in benchmarking. We slightly adapted the prompt for each model. Refer to our code attached for the details. It took up to a few hours to run the models, and the APIs cost less than 1 USD for each. Figure 12 shows an example prompt for our LLM-as-a-judge evaluation.

C.2. Result

In the human performance experiment, a participant is given a question, instructions (parts' image

[3] https://huggingface.co/

[4] https://en.wikipedia.org/wiki/DOT_(graph_description_language)

Table 7: Bias analysis result: generator-predictor

Predictor	Generator		
	GPT	Claude	Gemini
GPT	47.5	42.5	47.5
Claude	35.0	27.5	37.5
Gemini	27.5	30.0	37.5

Table 8: Bias analysis result: predictor-evaluator

Predictor	Evaluator			Human
	GPT	Claude	Gemini	
GPT	44.2	40.0	50.8	45.8
Claude	33.3	30.8	39.2	33.3
Gemini	30.8	30.8	36.7	31.7

and task graph), and a recording (video segments, instead of sampled frames), and answers the question. We asked three participants to answer questions, followed by the same LLM-as-a-judge, and then took the average. Due to the cost, we used only the subset of 20 questions out of the full 391 questions. As the annotators are the ones who performed the QA verification, we sampled different sets of 20 questions for each so that participants answered questions that they did not see in the verification stage. The average score was 67.5. Based on our manual inspection, humans make mistakes typically by failing on step status tracking and overlooking small, yet salient visual details. While the task is even difficult for humans who are familiar with the task, there is a gap between the human average score and the top-performing model's score, which shows room for improvement for models.

C.3. Qualitative Analysis

Table 9 and 10 show predictions from some of the models we evaluated. One typical failure case is that models pay more attention to instructions when frames also contain critical information. Interestingly, we found that the text-only model, DeepSeek-R1, sometimes predicted, or more precisely, *guessed*, correctly, only based on instructions, even though its reasoning process explicitly stated that it did not have access to the information of what the user was doing.

C.4. Bias Analysis

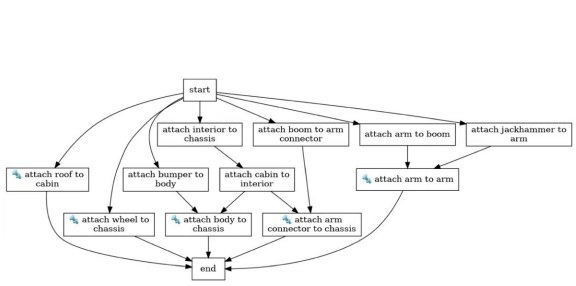
We also conducted self-preference bias analysis. Self-preference bias is a bias that LLMs favor their outputs over others (Panickssery et al., 2024). Since our work employs LLMs in multiple places, this type of bias may unexpectedly affect the benchmark results. As conducted in ProMQA, we investigate two types of biases to justify the quality of ProMQA-Assembly as a benchmark dataset. One is the bias between a QA generator and a predictor. Minor styles or characteristics of questions generated by a model may make it easier for the same model to predict the answers. The other is the bias between a predictor and an evaluator. This is the same setting as the original self-preference bias.

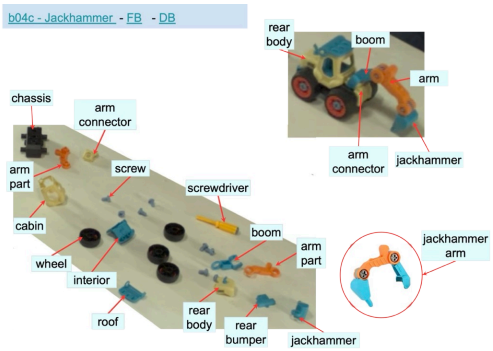
We first sampled a new set of 40 samples obtained in § 4.1, and prompted three generators (GPT4o, Claude 3.7 Sonnet, and Gemini 2.0 Flash) to generate QA pairs. Due to cost constraints, we used only one annotator to verify the questions and, based on the verification results, we obtained 20 valid multimodal procedural questions from each generator. When there were more than 20 target questions, we sampled from them. Next, we use the same set of LLMs to predict answers to each set of 20 questions, followed by manual judgment. This manual judgment used the same guidelines as our LLM-as-a-judge and was conducted by one annotator. According to the result in Table 7, we do not see an indication of the generator-predictor self-preference bias here. For instance, GPT-4o (predictor) performs the best on the GPT-4o-generated data compared to other predictors, but the trend is the same for the generated data by other models. Or, Gemini (predictor) performed the best on its data, but that is the same for other predictors.

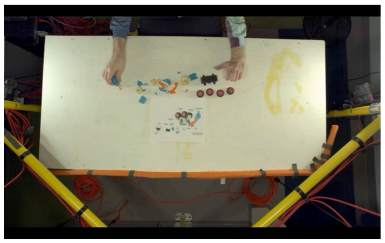
Next, we checked the predictor-evaluator self-preference bias. We prompted the same set of LLMs to evaluate the predicted answers we obtained above, i.e., 20 QAs $\times$ 3 generator $\times$ 3 predictor $= 180$ predictions. The result is shown in Table 8, as well as the human evaluation results. GPT-4o (evaluator) gave the best score to GPT-4o (predictor), but other evaluators consistently gave the best score to GPT-4o (predictor). Gemini (predictor) received the best score from Gemini (evaluator). Again, however, other models received the highest scores from Gemini (evaluator). Based on this result, we do not see the sign of the predictor-evaluator self-preference bias, either. In this bias analysis, the person who served as an adjudicator in § 4.3 performed the verification and the manual evaluation. We also calculated the judgment correlations between each model and the human evaluator using Kendall's Tau coefficient (Kendall, 1938). The correlations were 0.26 for GPT-4o, 0.27 for Claude 3.5 Sonnet, and 0.24 for Gemini 2.0 Flash. While we used the evaluation prompt and GPT-4o based on ProMQA's results, there remains room for improvement in evaluation methodology, which we leave for future work.[5]

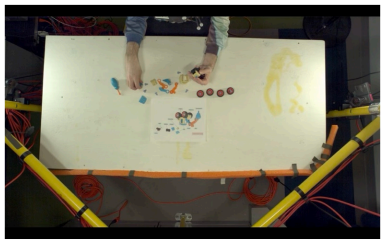
[5]Icons are from Freepik, www.flaticon.com

Table 9: Qualitative Analysis Example. The question requires visual understanding to identify that all the parts have been attached. While Qwen2.5-VL correctly answers, other models fail to answer. One possible reason for GPT-4o and Claude 3.7 Sonnet is that they rely too much on the instructions, instead of the frames, to answer the question, based on their answers.

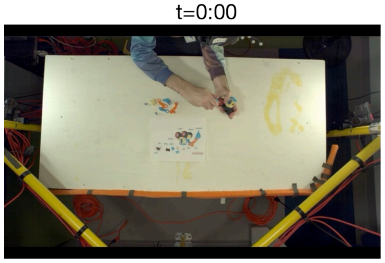


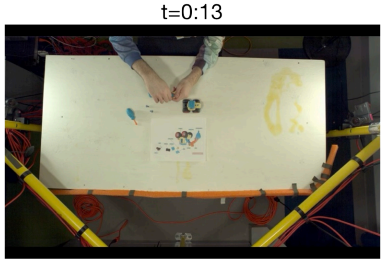


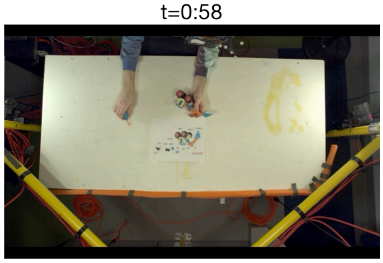

t=0:00


t=0:13


t=0:58

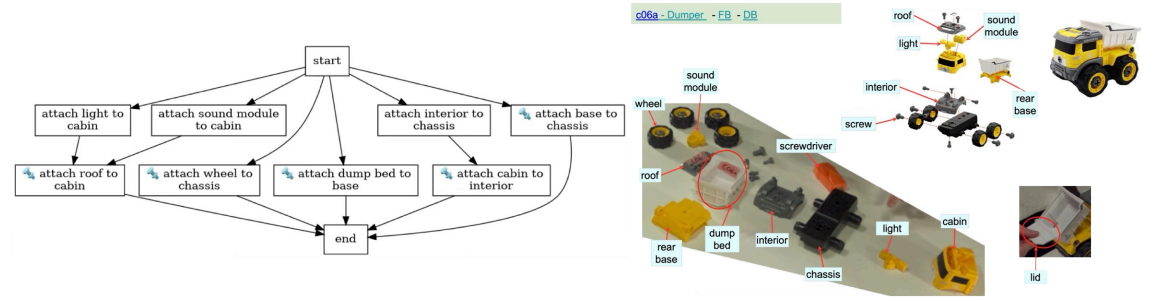

t=1:29

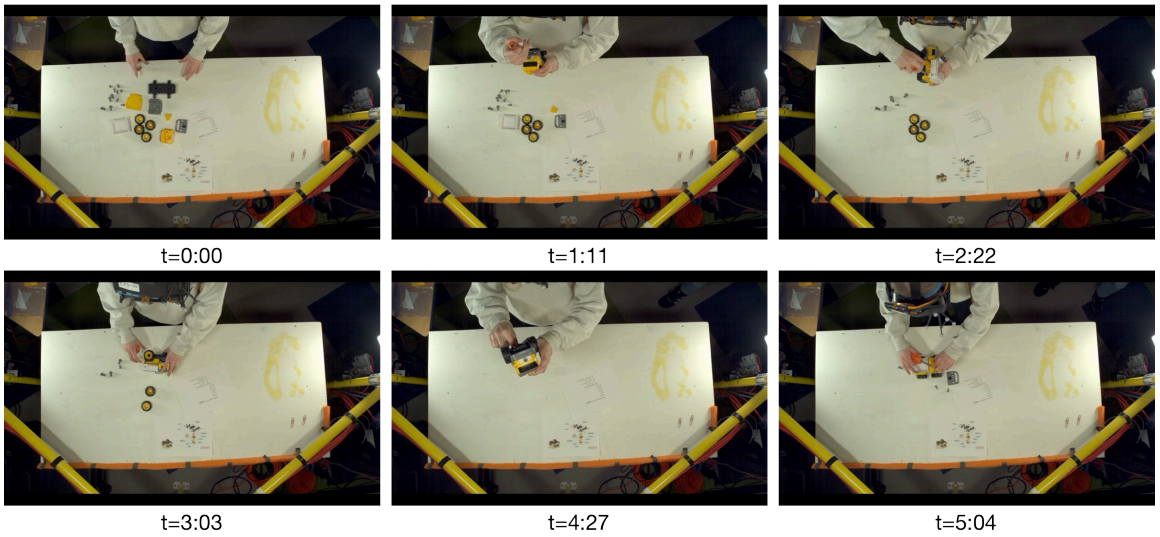

t=2:25


t=3:07

Question:	Am I done, or is there something left to finish?
Gold answer:	You're done! The toy is complete.
DeepSeek-R1:	✗ There are still steps left to complete.
Qwen2.5-VL:	✓ You are done.
GPT-4o:	✗ You are not done yet; you still need to attach the jackhammer to the arm and complete the remaining steps as per the manual.
Claude 3.7:	✗ Based on the assembly recording and the manual, you are not done yet. There are still several steps to complete, including attaching the jackhammer assembly to the main vehicle and connecting various components like the roof to the cabin with screws.

Table 10: Qualitative Analysis Example. All multimodal models fail to identify what the user is doing, i.e., they forgot to secure the interior to the chassis, so they are detaching the cabin. While this question requires visual understanding, DeepSeek guesses somewhat correctly, presumably because attaching the interior is one of the prerequisite steps.





Question:	What should I do next to fix my mistake?
Gold answer:	You should detach the cabin from interior and secure the interior with screw and reattach the cabin. Detach the roof, sound module, and cabin from the interior, so that you can screw in the interior. Then reattach everything in reverse order.
DeepSeek-R1:	(... I need to figure out the correct next step based on the provided DOT graph. ... since the user's question is about fixing a mistake, the most logical next step would be to complete a prerequisite that was skipped. Looking at the graph, after starting, the user should attach the interior to the chassis, which then leads to attaching the cabin to the interior. So maybe they skipped attaching the interior first. ...) ▲ To fix your mistake, attach the interior to the chassis first.
Qwen2.5-VL:	✗ You should attach the wheel to the chassis with a screw.
GPT-4o:	✗ To fix your mistake, identify the last incorrect step in the assembly process by comparing your progress with the manual, disassemble the affected parts, and reassemble them in the correct order.
Claude 3.7:	✗ Based on the recording, you need to attach the light to the cabin before attaching the roof, as you appear to have attached the roof without first installing the light component according to the assembly manual's dependency order.

```
# Instruction
You are playing with a take-apart toy alongside your senior friend,
    who has more experience with it.
You identify your skill level as intermediate.
Here is/are the step(s) you have already performed in the actual order
    :
* attach wheel to chassis
* attach interior to chassis (mistake: wrong position)
* detach interior from chassis
* attach interior to chassis (mistake: wrong position)

Your action history indicates some location errors.
While your friend is aware of these, you may or may not know.

# Task
What question would you ask your senior friend regarding the part
    location of the current (latest) step?

Create three possible QAs as a list. Questions should be something you
     would ask to your friend and the answers are something your friend
     would say.
Follow the format below:
* <question 1>
     * <answer 1>
     * ...
* ...

# Note
* Each question should require information about both A) what you have
     done and B) what you are supposed to do to answer correctly.
* Each question/answer should consist of one short and concise
    sentence/phrase.
* Your friend is watching over you, so you do not have to explain your
     situation, just ask questions directly.
* If multiple correct answers exist, provide all of them.
* Consider different personas you might adopt: careless/careful, child
    /teen/adult, casual/formal, etc.
* Return as diverse QA pairs as possible in terms of tone, wording,
    etc.
* Come up with different question types: yes/no-questions, Wh-
    questions, etc.

# Example
* Is X in the right place?
     * ...
* Where should I put this now?
     * ...
* ...

# Response
```

Figure 6: QA generation prompt example: *"Default"* for "location" type.

```
# Instruction
You are playing with a take-apart toy alongside your senior friend,
    who has more experience with it.
You identify your skill level as intermediate.
Here is/are the step(s) you have already performed in the actual order
     (Note: the indented ones are the detailed, fine-grained steps):
* attach excavator arm to chassis (mistake: wrong position) (no-screw
    step)
     * position excavator arm
* attach arm connector to excavator arm (mistake: previous one is a
    mistake) (screw-required step)
     * Pick up arm connector
     <<<omitted for brevity>>>
* detach excavator arm from chassis (no-screw step)
     * tilt up excavator arm
     <<<omitted for brevity>>>
* detach arm connector from excavator arm (no-screw step)
     * rotate excavator arm
The last step may be incomplete.

You do not have any missing steps either because you have followed the
     correct order or due to your previous mistake(s).
While your friend understand the correct orders, you may or may not
    know.

# Task
What question would you ask your senior friend regarding possible
    missing steps?
<<omitted for brevity>>

# Response
```

Figure 7: QA generation prompt example: *"With fine"* for "missing" type. Note that some fine-grained actions are omitted for brevity.

```
# Instruction
You are playing with a take-apart toy alongside your senior friend,
    who has more experience with it.
You identify your skill level as intermediate.
Attached is the image of parts, final picture, and/or exploded view.
Here is/are the step(s) you have already performed in the actual order
    :
* attach interior to rear body (mistake: wrong position)
* attach cabin to interior (mistake: previous one is a mistake)

The above list contains your previous mistake information, if any.
While your friend is aware of these, you may or may not know.

# Task
What question would you ask your senior friend regarding the possible
    past mistakes?
<<omitted for brevity>>
# Response
```

Figure 8: QA generation prompt example: *"With image"* for "past" type. Note that a corresponding parts' image as shown in the lower middle of Figure 1 is also fed to a model.

```
# Instruction
You are playing with a take-apart toy alongside your senior friend,
   who has more experience with it.
You identify your skill level as expert.
Here is/are the step(s) you have already performed in the actual order
   :
* attach wheel to chassis
* attach rear bumper to chassis
* attach cabin to interior

Below is the remaining part(s):
* bumper
* dump bed/base
* lid

# Task
What question would you ask your senior friend regarding next steps?

Create three possible questions as a list.
Follow the format below:
* <question 1>
* <question 2>
* <question 3>
<<omitted for brevity>>
# Response
```

Figure 9: QA generation prompt example: prompt for question generation in *"Q then A"* for "next" type.

```
# Instruction
You are watching over your younger frined, who is playing with a take-
   apart toy. You are more experienced with it.
They identify their skill level as expert.
Your younger frined has done the following steps in this order:
* attach door to base
* attach transport cabin to base
<<omitted for brevity>>

Any of the steps below can be done as the possible next step:
* attach rear roof to transport cabin
<<omitted for brevity>>

# Task
Suppose you were asked the following question by your yonger friend:
Question: "Should I attach the bumper before the roof?"
What response(s) would you give them?

Create possible answers as a list, like:
* <answer 1>
* ...

# Note
* An answer should consist of one concise sentence/phrase.
* If multiple correct answers exist, provide all of them.

# Response
```

Figure 10: QA generation prompt example: prompt for answer generation in *"Q then A"* for "next" type.

```
# Instruction
This is a multimodal question answering task.
A user is assembling a toy car.

# Parts/Final Picture/Exploded View
This is the image containing the parts, final picture, and/or exploded
    view.
<<<parts image added here>>>

# Assembling Manual
This is the assembling manual as text in the DOT (graph description
   language from Graphviz) format.
Each node represents one step and each edge represents an order
   dependency.
Two nodes connected by an edge must be performed in the specified
   order.
Nodes that are not directly connected can be performed in any order,
   as long as their respective prerequisites have been completed.
digraph G {
start;
end;
"attach body to chassis w/ screw";
"attach interior to chassis w/ screw";
"attach roller to push frame w/ screw";
"attach roof to cabin w/ screw";
"attach wheel to chassis w/ screw";
"attach arm connector to chassis w/ screw";
"attach arm connector to push frame";
"attach bumper to body w/ screw";
"attach interior to cabin";
start -> "attach roller to push frame w/ screw";
start -> "attach interior to chassis w/ screw";
"attach wheel to chassis w/ screw" -> end;
"attach roller to push frame w/ screw" -> end;
start -> "attach roof to cabin w/ screw";
"attach roof to cabin w/ screw" -> end;
start -> "attach wheel to chassis w/ screw";
start -> "attach arm connector to push frame";
"attach arm connector to push frame" -> "attach arm connector to
   chassis w/ screw";
"attach bumper to body w/ screw" -> end;
"attach arm connector to chassis w/ screw" -> end;
"attach body to chassis w/ screw" -> "attach bumper to body w/ screw";
"attach interior to cabin" -> "attach body to chassis w/ screw";
"attach interior to chassis w/ screw" -> "attach interior to cabin";
}

# Recording
These are the sampled frames in sequence from the recording of the
   user's activity.
<<<sampled frames added here>>>
# Task
The user asked the following question. Answer the question in one
   consice sentence, based on the give information above (parts,
   manual, and recording).
[Question]
Why is the rear body not fitting properly?
[Answer]
```

Figure 11: Benchmarking prompt example. Note that the parts image and sampled frames are omitted for brevity.

```
## Instruction ##
This is an evaluation task.
You will be given a question, gold answer(s), and predicted answer.
Your task is to evaluate if the predicted answer matches against the
   gold answer(s).

Here is/are the step(s) they have already performed in the actual
   order:
* attach wheel to chassis
* attach base to chassis
* detach base from chassis
* attach interior to chassis

Give your ternary judge 0, 1, or 2:
* 0 means the predicted answer is wrong (unmatch)
* 1 means the predicted answer is partially correct/wrong (partial
   match)
* 2 means the predicted answer is correct (match)
When multiple gold answers are available (provided as a list), the
   predicted answer is correct/partially correct if it matches/
   partially matches with at least one of the gold answers.

Provide your feedback as follows:
## Feedback ##
[Rationale] (your rationale for the judge, as a text)
[Judge] (your judge, as a number, 0, 1, or 2)

## Note ##
The question is being asked by a user who is playing with a take-apart
    toy.
Gold answer(s) are created by well-trained humans.
Predicted answer is created by a machine, based on the corresponding
   instruction and the frames of the assemblying process recording.

## Task ##
Now, here are the question, gold answer(s), and predicted answer:
[Question]
Am I missing a step before attaching the interior?
[Gold Answer(s)]
- Yes, you need to secure the rear base.
- No, you're on the right track.
[Predicted Answer]
Yes, you need to attach the sound module to the chassis before
   attaching the interior.

## Feedback ##
[Rationale]
```

Figure 12: LLM-as-a-judge prompt example.