LEGO Co-builder: Exploring Fine-Grained Vision-Language Modeling for Multimodal Assembly Assistants

Haochen Huang*
Vrije University Amsterdam
Amsterdam, Netherlands
University of Amsterdam
Amsterdam, Netherlands
Centrum Wiskunde & Informatica
Amsterdam, Netherlands
peterdavinci@yahoo.com

Yue Su*
Vrije University Amsterdam
Amsterdam, Netherlands
y.su@vu.nl

Xin Sun
National Institute of Informatics
Tokyo, Japan
University of Amsterdam
Amsterdam, Netherlands
xsun@nii.ac.jp

Moonisa Ahsan
Centrum Wiskunde & Informatica
Amsterdam, Netherlands
moonisa@cwi.nl

Mohammad Aliannejadi
University of Amsterdam
Amsterdam, Netherlands
m.aliannejadi@uva.nl

Irene Viola
Centrum Wiskunde en Informatica
Amsterdam, Netherlands
irene@cwi.nl

Zhaochun Ren
Leiden University
Leiden, Netherlands
z.ren@liacs.leidenuniv.nl

Chuang Yu
UCL Interaction Centre
University College London
London, United Kingdom
chuang.yu@ucl.ac.uk

Aneta Lisowska
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
a.j.lisowska@vu.nl

Artem Belopolsky
Vrije Universiteit Amterdam
Amsterdam, Netherlands
a.belopolskiy@vu.nl

Koen Hindriks
Vrije Universiteit Amterdam
Amsterdam, Netherlands
k.v.hindriks@vu.nl

Pablo Cesar
Centrum Wiskunde & Informatica
Amsterdam, Netherlands
Delft University of Technology
Delft, Netherlands
p.s.cesar@cwi.nl

Junxiao Wang
Guangzhou University
Guangzhou, China
junxiao.wang@gzhu.edu.cn

Jiahuan Pei†
Vrije Universiteit Amsterdam
Amsterdam, Netherlands
j.pei2@vu.nl

Abstract

Vision-language models (VLMs) are facing the challenges of understanding and following multimodal assembly instructions, particularly when fine-grained spatial reasoning and precise object state detection are required. In this work, we explore **LEGO Co-builder**, a hybrid benchmark combining real-world LEGO assembly logic with programmatically generated multimodal scenes. The dataset captures stepwise visual states and procedural instructions, allowing controlled evaluation of instruction-following, object detection, and state detection. We introduce a unified framework and assess leading VLMs such as GPT-4o, Gemini, and Qwen-VL, under zero-shot and fine-tuned settings. We also evaluated the framework using a reasoning-focused model, GLM-4.1-thinking. Our results show that while object detection achieved high performance (98.16% with fine-tuned InstructBLIP), fine-grained scene understanding and assembly state detection remain challenging: Fine-tuned MiniGPT-v2 reached only 37.52% F1 for identifying theme entities, and even advanced models such as GPT-4o achieved just 40.54% F1 on state detection. This highlights gaps in fine-grained visual understanding among existing models. We release the benchmark, codebase, and generation pipeline to support future research on multimodal assembly assistants grounded in real-world workflows.[1]

ACM Reference Format:
Haochen Huang, Yue Su, Xin Sun, Moonisa Ahsan, Mohammad Aliannejadi, Irene Viola, Zhaochun Ren, Chuang Yu, Aneta Lisowska, Artem Belopolsky,

*First co-authors.
†Corresponding author.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

ICMI '26, Napoli, Italy

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2318-6/26/10
https://doi.org/10.1145/3776574.3831119

[1]https://github.com/peterhuang-coding/LEGO-Co-builder

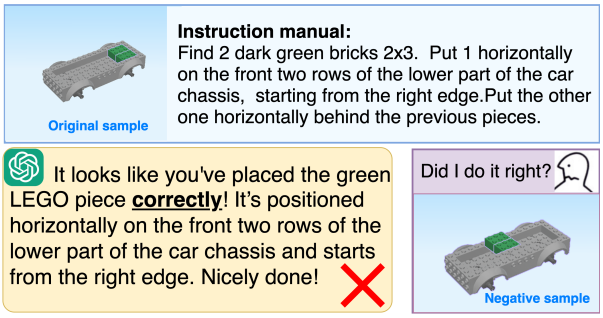


Figure 1: An example of incorrect state detection by GPT-4o during LEGO assembly. Given an image and text instructions, the model fails to accurately recognize an incorrectly placed LEGO part (highlighted in green), demonstrating the challenge of fine-grained vision-language alignment.

Koen Hindriks, Pablo Cesar, Junxiao Wang, and Jiahuan Pei. 2026. LEGO Co-builder: Exploring Fine-Grained Vision-Language Modeling for Multimodal Assembly Assistants. In *INTERNATIONAL CONFERENCE ON MULTIMODAL INTERACTION (ICMI '26), October 05–09, 2026, Napoli, Italy*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3776574.3831119

1 Introduction

Multimodal instruction-following assistants are gaining increasing relevance in domains requiring precise procedural understanding, such as furniture construction [53], automotive manufacturing [5], and industrial product assembly [18]. These tasks demand step-by-step reasoning, spatial awareness, and accurate interpretation of visual and textual instructions—capabilities that current AI systems still struggle to reliably deliver.

Traditional computer vision research has tackled various purely visual tasks, including action segmentation, recognition, anticipation, and object state detection [15, 20, 32, 50]. Procedural activity understanding has also been explored using video datasets with or without textual narration [26, 40, 46, 59]. Some methods incorporate textual alignment, such as narration-video grounding or instruction-conditioned generation [33, 35], but often rely on separate architectures for each task type.

Vision-language models (VLMs) offer a promising solution by bridging visual and textual modalities, enabling models to align procedural language with visual observations. Recent advances in large-scale models such as GPT-4o [1], LLaVa [30], Qwen-VL [3], BLIP-2 [28], MiniGPT-v2 [8], and more recent reasoning-augmented variants like VLM-R1 [41], GLM-4.1v-thinking [22], and Gemini2.5 [13], have demonstrated strong performance on a range of general-purpose multimodal tasks. While most existing benchmarks focus on coarse-grained capabilities like object recognition and captioning, they often fail to evaluate the detailed, procedural understanding required for step-by-step manual assembly. While recent VLMs research [6, 10, 24] aim to unify vision-language reasoning under a single architecture, challenges remain—particularly for tasks that require dense and fine-grained perception [9, 38, 51].

In this work, we address the need for instruction-following benchmarks that evaluate a model's ability to reason over both visual and linguistic input in detail. We focus on LEGO brick assembly as a representative manual task that is richly structured, visually complex, and procedurally grounded. Figure 1 illustrates the limitations of current models in such tasks. When prompted with a visual scene and corresponding instruction, GPT-4o fails to detect a misplaced LEGO piece, highlighting the challenge of fine-grained visual scene understanding and object state recognition. This motivates the need for benchmarks that go beyond high-level visual understanding and evaluate precise spatial compliance with procedural steps.

To address this gap, we present **LEGO Co-builder**, a benchmark and dataset designed to evaluate fine-grained instruction-following capabilities in LEGO assembly tasks. Our dataset includes multimodal sequences consisting of visual snapshots, object states, and procedural steps derived from human-crafted manuals. We introduce a unified task formulation and evaluate nine leading VLMs in both zero-shot and fine-tuned settings. Our contributions are summarized as follows: (1) We develop a unified vision-language architecture to benchmark fine-grained, multimodal, instruction–following capabilities in procedural manual-guided assembly tasks. (2) We evaluate prevailing VLMs on a dataset of LEGO assembly sequences with grounded visual and textual supervision, under both zero-shot and fine-tuned settings. (3) We release the dataset, benchmarking code, and a modular synthetic data generation pipeline to support future research in multimodal instruction-following.

Societal Impact. This work advances the development of multimodal AI assistants that can transform how people learn and perform complex physical tasks. By enabling fine-grained vision-language understanding, our benchmark supports the creation of multimodal educational tools that combine visual input, language, and potentially augmented reality (AR) to enhance hands-on learning experiences. These systems can improve training in domains like education, surgical practice, and industrial training. Moreover, the ability to interpret and verify procedural steps visually opens up promising avenues for assistive technologies—particularly for blind or visually impaired learners—by providing real-time, AI-driven guidance through tasks that were previously inaccessible. In this way, our research supports both innovation in teaching modalities and greater inclusivity in skill development.

2 Fine-Grained Vision-Language Modeling

2.1 Task Definition

We investigate manual-guided LEGO assembly and define the following three vision-language tasks (See in Figure 2):

(**T1**) **Scene Understanding**. Given the current step's image V_t and the task-specific query Q_t, the model outputs a response containing the scene description for the current assembly step I_t. Q_t is the task description contextualized with the previous step's textual instruction I_{t-1}. It assesses the model's ability to accurately describe the current assembly step from an instruction manual, including but not limited to generating object representations, their properties, and the assembly procedure.

(**T2**) **Object Detection**. Given a seeking step's image $V_{t'}$ ($t' = t-1$) and task-specific query $Q_{t'}$, the model outputs a response containing the positional text $P_{t'}$, which includes identified object and its

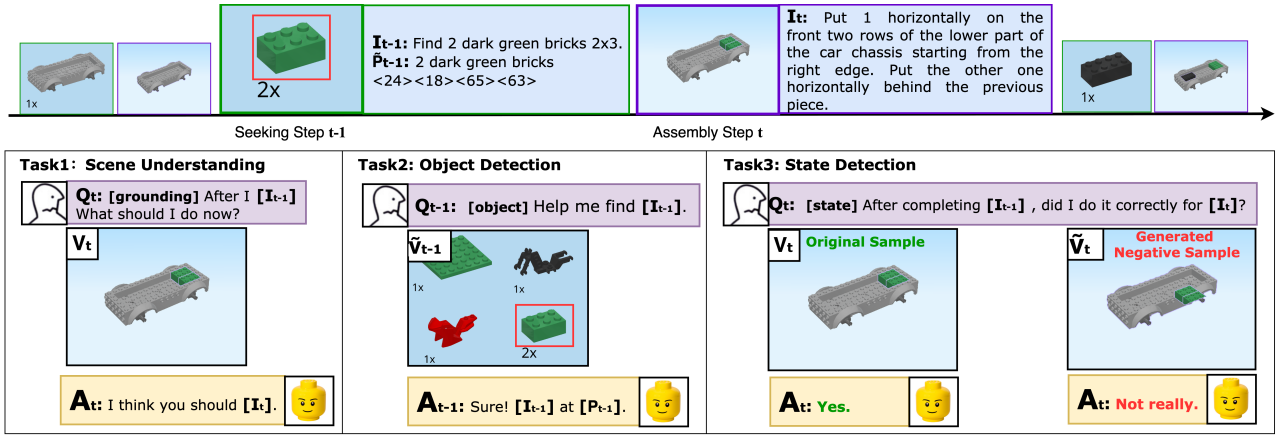


Figure 2: An illustration of the proposed three vision-language model (VLM) tasks, highlighting the core capabilities required by the one-for-all VLM architecture for LEGO assembly based on the procedural instruction manual.

coordinate formatted as "[Object] [Xleft] [Ytop] [Xright] [Ybottom]". The query $Q_{t'}$ is the task description contextualized with the corresponding textual instruction $I_{t'}$. It assesses the model's ability to accurately identify and locate objects in the scene, ensuring they are positioned correctly for the next assembly action.

(T3) State Detection. Given an assembly step's image, either the original manual image V_t or a generated negative sample $\tilde{V}_t$, and task-specific query Q_t, the model outputs a response indicating a correct or incorrect state. It evaluates the model's ability to verify the accuracy of the assembly progression, determining if the assembly action has been correctly completed.

These tasks have been crafted to rigorously test and demonstrate the capabilities of VLMs in LEGO Assembly, focusing on their ability to understand and interpret complex multimodal inputs.

2.2 Unified Vision-Language Architecture

We investigated existing VLMs and derived a universal architecture for the proposed tasks, as illustrated in Figure 3. Given an image, either from manuals or real-world scenes, and a task-specific query, a VLM based on this architecture generates a textual response as the output. This architecture integrates a *vision encoder*, a *large language model*, and a *vision-language projector* as core modules, along with a task-specific *query* for task adaptation. Specifically, the vision encoder processes visual inputs, the large language model interprets and generates textual information, and the vision-language projector aligns visual and textual data for seamless task execution. The query can be customized using the following key components: (1) *Task-specific token:* Special tokens such as "[grounding]", "[object]", and "[state]" are introduced for task T1, T2, T3, respectively, to enhance task focus and accuracy. (2) *Instruction:* Relevant manual instructions or task directives are incorporated to provide the model with context, ensuring responses align with task requirements. (3) *Format-directive:* A directive to clarify the expected output format, ensuring outputs are precise and directly applicable.

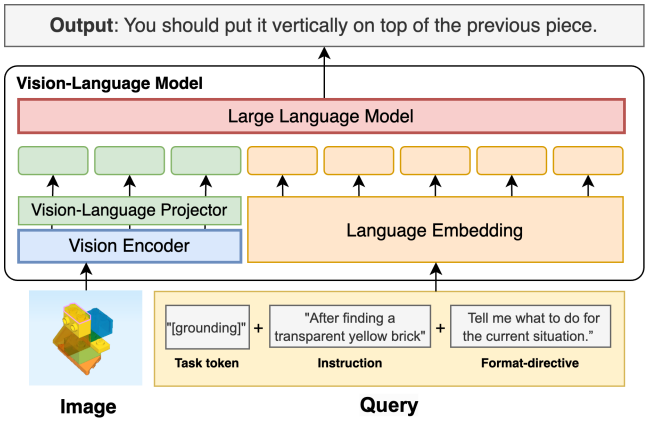


Figure 3: The architecture for one-for-all vision-language modeling. It integrates a vision encoder, a large language model, and a vision-language projector as core functional modules, with a task-specific query for task adaptation.

3 Dataset Creation for Task Design

3.1 Manual Crawling and Scene Data Matching

We outline the procedure for creating the dataset for task **T1**. First, we collected 65 official LEGO instruction manuals [25], designed to help blind and visually impaired users assemble LEGO sets accurately. Figure 4 shows an example of a part of the instruction manual we crawled from the official LEGO website. Each instruction manual comprises several elements: (1) step-by-step textual instructions; (2) corresponding image for each step; and (3) tags provided by LEGO include instruction and image tags. Then, we split a full manual into several assembly sessions, each session includes two types of steps: object seeking and object assembly. Each seeking step is followed by an assembly step , forming a pair. Next, we iterated through all sessions to generate the scene understanding dataset, $D_{T1} = \{(Q_t, V_t, A_t)\}|_{t=0}^{|D_{T1}|}$, where each element is a

triplet of (query, image, text), and the query Q_t is constructed by filling the query template with the task token "[grounding]" and the previous instruction I_{t-1} as context.

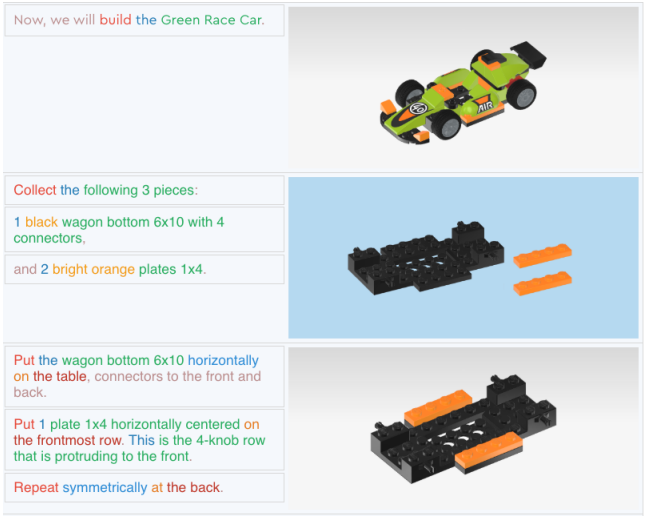


Figure 4: An example of LEGO instruction manual with three sequential instruction steps. Each step includes textual instructions paired with corresponding images to guide the assembly process.

3.2 Object Position Inference

This subsection outlines the procedure for creating the dataset for task **T2** as illustrated in Figure 5. First, we iterated through all object-

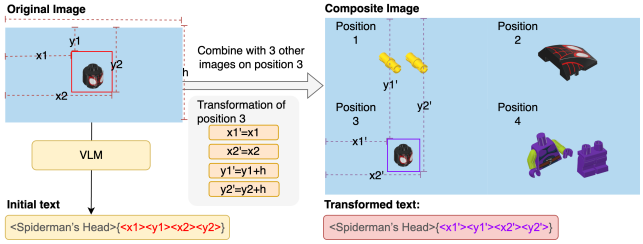


Figure 5: An illustration of object detection data creation.

seeking steps, where users are asked to find specific objects. Second, we iterated all images to generate the object detection dataset, $D_{T2} = \{(Q_{t'}, \tilde{V}_{t'}, A_{t'})\}|_{t'=0}^{|D_{T2}|}$, where each element is a triplet of (query, image, text) for a seeking step $t' = t - 1$. The query $Q_{t'}$ is constructed by filling the query template with the task token "[object]" and the current instruction $I_{t'}$. The response $A_{t'}$ contains a positional text $\tilde{P}_{t'}$ formatted as "[Object] [Xleft] [Ytop] [Xright] [Ybottom]". This is initially generated by querying the image using MiniGPT-v2 [8], followed by rule-based filtering and manual spot checks. Importantly, test-set annotations are fixed and not adapted per model, preventing reward hacking or model-specific bias. The composite image $\tilde{V}_{t'}$ is created by combining the current image $V_{t'}$ with three randomly sampled images. Last, the initial coordinates are adjusted to fit the composite image.

3.3 Variant State Generation

This subsection outlines the procedure for creating the dataset for task **T3**, as illustrated in Figure 6. First, we iterated through

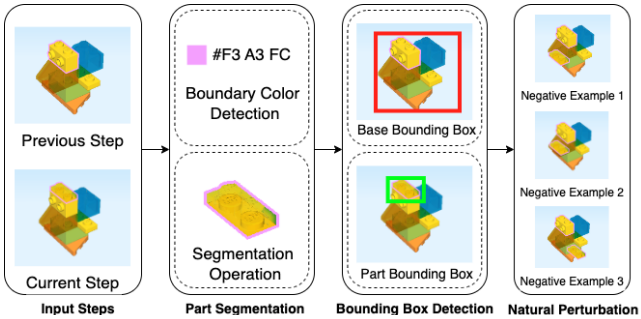


Figure 6: An illustration of synthesized negative samples for state detection.

all object-assembly steps along with their corresponding previous steps, in which users are trained to assemble the parts identified in the prior steps. Second, we conducted part segmentation by detecting the boundary colors and segmenting the parts to be assembled. In each assembly session, the objects to be assembled in the current step are tagged in the previous step, with their boundaries highlighted in distinct colors that vary between sessions. The highlighted colors are selected based on predominant color detection using K-means clustering [44], followed by Hue filtering [12] and manual correction for accuracy. Third, we added bounding boxes to the segmented objects. Fourth, we applied a natural perturbation to the part to be assembled by randomly shifting it within the background box, ensuring it moves by at least 5%. We treated each state from the manual as a positive sample and generated three variant states as negative samples. Then, we constructed the state detection dataset, $D_{T3} = \{(Q_t, V_t, A_t, \tilde{V}_t, \tilde{A}_t, \tilde{V}_t', \tilde{A}_t', \tilde{V}_t'', \tilde{A}_t'')\}|_{t=0}^{|D_{T3}|}$, where the query Q_t is constructed by filling the query template with the task token "[state]", the current instruction I_t, and the previous instruction I_{t-1} as context. The variant states $\{\tilde{V}_t, \tilde{V}_t', \tilde{V}_t''\}$ are natural pertubations of V_t. The responses A_t and $\{\tilde{A}_t, \tilde{A}_t', \tilde{A}_t''\}$ indicate the correct and incorrect statuses of the assembly state, respectively. Note that perturbation variants are first generated per session, after which samples are split for train and test sets.

3.4 Dataset Statistics and Partition

We summarize the statistics of LEGO-VLM dataset in Table 1. It is generated from 65 LEGO instruction manuals and divided into 397 sessions, covering 5,612 scenes, 4,784 objects, and 2,716 states. It consists of 5,614 instruction steps, with each step containing an image and corresponding textual instructions. Out of these, 4,814 steps focus on object seeking, while 5,614 steps involve parts assembly, covering 3,172 states with 222 distinct boundary colors. Overall, the dataset includes 35,612 vision-language data samples: 5,612 for T1, 19,136 for T2, and 10,864 for T3, respectively. Note

that all test data are strictly held out, with no overlap with training data, ensuring no data leakage.

# **Manual**	# **Session**	# **Scene**	# **Object**	# **State**
65	397	5,612	4,784	2,716
# **Step**	Overall	Identification	Assembly	
	10,428	4,814	5,614	
# **Sample**	All Tasks	T1	T2	T3
Overall	35,612	5,612	19,136	10,864
Training	26,709	4,209	14,352	8,148
Testing	8,903	1,403	4,784	2,716

Table 1: Statistics of LEGO-VLM dataset.

4 Experimental Setup

4.1 Benchmark Models

We benchmark the following nine VLMs: (1) **mPLUG-OWL2 [52]** replaces attention with a modality adapter in a large language model (LLM) decoder. (2) **BLIP2 [28]** uses a two-stage pre-trained Q-Former between an image encoder and a LLM to bridge the vision-language modality gap. (3) **LLaVa [30]** combines a visual CLIP encoder and a language decoder Vicuna for general-purpose visual language understanding. (4) **Qwen-VL [3]** connects a LLM with a visual encoder using a position-aware vision-language adapter towards fine-grained visual understanding. (5) **InstructBLIP [14]** explores general-purpose vision-language instruction tuning based on the pretrained BLIP2. (6) **MiniGPT-v2 [8]** links a frozen ViT visual encoder with Llama-2 via a projection layer, applicable for diverse tasks via task-specific multimodal instructions. (7) **Otter [26]** is tuned on the OpenFlamingo, conditioning the language model on images for multi-modal perception and reasoning. (8) **MiniGPT-4 [58]** integrates a frozen visual ViT&Q-Former encoder and Vicuna via a projection layer, unlocking advanced multimodal capabilities like GPT-4. (9) **GPT-4o** [34] is a widely used commercial model accessible via APIs. (10) **Qwen-VL-2.5** update Qwen-VL with the latest Qwen LLM [36]. (11) **Gemini-2.5-flash** [13] is a thinking model, designed to tackle increasingly complex problems. (12) **GLM-4.1-thinking** [22] is designed to explore the upper limits of reasoning.

4.2 Evaluation Metrics

We consider multiple metrics for comprehensively evaluating specific tasks. *Scene understanding:* (1) **F1-Theme** is the identified theme entities' F1 score that measures the harmonic mean of precision and recall. It evaluates the accuracy of correctly mentioned theme entities in the generated instructions compared to the reference instructions, obtained from instruction manuals. (2) **BLEU** measures precision, which measures the ratio of 1-grams in the generated responses that match those in the reference responses. (3) **ROUGE** measures recall, which calculates the ratio of 1-grams in the reference responses that are captured by the generated responses. *Object Detection:* (1) **F1-Object** is identified object entities' F1 score. It evaluates the accuracy of identified object entities in the generated output compared to the reference data. (2) **Intersection over union (IOU)** measures the overlap between the predicted and reference bounding boxes. *State Detection:* (1) **F1-State** score is a metric that combines precision and recall, measuring the model's accuracy in identifying the correct states. It considers both precision and recall, effectively capturing the performance across both the minority (original state) and majority (generated state) classes. (2) **False positive rate (FPR)** measures the ratio of incorrectly classified negative instances as positive, highlighting the issue of incorrect training content.

5 Outcomes

5.1 Benchmarking Results

We compare the performance of nine prevailing VLMs on the proposed LEGO-VLM dataset, without and with finetuing, as shown in Table 2.

First of all, existing models struggle with fine-grained assembly tasks in AR. While the F1-Object score for task T2 using the fine-tuned InstructBLIP is high at 98.16%, the IOU is only 47.20%, indicating minimal overlap between the predicted object positions and the ground truth, which is still unsatisfactory. Even the commercial model GPT-4o achieves only 66.67% for the F1-Object score and 21.68% on task T2. This might also caused by the difficulty of understanding scenes. For example, fine-tuned LlaVA shows the best overlap with reference instructions, scoring 42.97% on ROUGE and 17.73% on BLEU. However, the fine-tuned MiniGPT-v2, despite being the top performer in theme entity identification, achieves only 37.52%. This indicates that understanding and generating theme entities, such as LEGO parts and their properties, remains a significant challenge. For state detection in task T3, the fine-tuned InstructBLIP achieves a 0.00% of FPR, meaning it almost perfectly avoided incorrectly classifying negative instances as positive. However, the F1-State score is just 0.02%, indicating a significant failure in identifying the correct states. These results highlight the challenges of the proposed fine-grained vision tasks, reinforcing the need to advance this area and its resources as a valuable research topic.

Second, open-resource VLMs with fine-tuning generally outperform or are comparable to commercial models. For one hand, top-performing open-resource VLMs are more informative with identified accurate entities. For example, MiniGPT-v2 with fine-tuning achieves an 11.71% higher F1-Theme score on task T1; InstructBLIP with fine-tuning achieves a 31.48% higher F1-Object score, while Qwen-VL shows only a 1% decrease, compared with the commercial GPT-4o. This may be due to the proposed dataset enhancing the model's understanding of domain-specific knowledge, such as LEGO parts and their properties. On the other hand, top-performing open-resource VLMs can generate assembly instructions that closely align with the provided manuals. For example, the fine-tuned LlaVA achieves ROUGE and BLEU scores that are 2.30 and 8.87 times higher than those of GPT-4o, respectively. Besides, fine-tuning generally improves the performance of VLMs, as indicated by the results marked with the superscript "▲". This highlights the significant difference between the proposed dataset and those used to train general commercial models.

Last, task difficulty varies significantly, with the biggest challenge being the alignment of positional information in images with the corresponding textual information from the query. Specifically,

Model	T1: Scene understanding						T2: Object detection				T3: State detection			
	F1-Theme ↑		ROUGE ↑		BLEU ↑		F1-Object ↑		IOU ↑		F1-State ↑		FPR ↓	
PEFT (LoRA)	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
mPLUG-OWL2	23.16	29.70$^{\blacktriangle}$	25.23	32.75$^{\blacktriangle}$	3.04	8.46$^{\blacktriangle}$	77.94	93.05$^{\blacktriangle}$	14.23	34.88$^{\blacktriangle}$	36.82	15.00$^{\blacktriangledown}$	63.41	35.00$^{\blacktriangle}$
BLIP2	27.85	32.65$^{\blacktriangle}$	29.55	40.35$^{\blacktriangle}$	6.40	12.50$^{\blacktriangle}$	77.35	84.48$^{\blacktriangle}$	**34.25**	40.57$^{\blacktriangle}$	24.71	28.50$^{\blacktriangle}$	100.00	94.87$^{\blacktriangle}$
LlaVA	32.10	35.88$^{\blacktriangle}$	13.63	**42.97**$^{\blacktriangle}$	0.99	**17.73**$^{\blacktriangle}$	54.39	72.32$^{\blacktriangle}$	∅	**60.98**$^{-}$	25.04	30.00$^{\blacktriangle}$	99.41	50.00$^{\blacktriangle}$
Qwen-VL	34.84	37.00$^{\blacktriangle}$	**29.57**	39.08$^{\blacktriangle}$	5.19	13.13$^{\blacktriangle}$	78.56	89.15$^{\blacktriangle}$	25.60	30.08$^{\blacktriangle}$	39.77	**39.53**$^{\blacktriangledown}$	97.99	96.82$^{\blacktriangle}$
InstructBLIP	32.85	32.85^{-}	29.20	36.87$^{\blacktriangle}$	4.91	11.52$^{\blacktriangle}$	79.76	**98.16**$^{\blacktriangle}$	∅	47.20^{-}	0.00	0.02$^{\blacktriangle}$	**1.22**	**0.00**$^{\blacktriangle}$
MiniGPT-v2	33.06	**37.52**$^{\blacktriangle}$	34.72	32.56$^{\blacktriangledown}$	**9.81**	8.28$^{\blacktriangledown}$	84.95	85.91$^{\blacktriangle}$	26.98	25.94$^{\blacktriangledown}$	36.76	38.64$^{\blacktriangle}$	60.42	80.72$^{\blacktriangledown}$
Otter	11.49	/	12.12	/	1.28	/	72.39	/	∅	/	35.33	/	75.88	/
MiniGPT-4	34.11	/	15.05	/	1.88	/	87.09	/	30.20	/	37.19	/	67.37	/
GPT-4o	25.81	/	18.67	/	2.00	/	66.67	/	21.68	/	**40.54**	/	43.06	/
Qwen-VL-2.5	19.13	/	15.34	/	2.70	/	75.55	/	10.63	/	33.85	/	36.70	/
GLM-4.1-thinking	35.73	/	19.44	/	2.31	/	70.64	/	4.40	/	39.61	/	38.53	/
Gemini-2.5-flash	**37.35**	/	28.28	/	6.21	/	**93.12**	/	13.16	/	28.81	/	8.72	/

Table 2: Benchmarking vision-language models on LEGO-VLM dataset. The upper part represents open-resourced vision-language models, both without (/wo) and with (/w) parameter-efficient fine-tuning (PEFT) using low rank adaptation (LoRA). The lower part presents results obtained via API calls to the latest vision-language models. The bold font indicates the highest score in each column. Symbols ↑ and ↓ denote that higher and lower values are better, respectively. Symbol "-" indicates the model is not applicable for fine-tuning. Symbol "∅" denotes a meaningless zero as the model fails to generate output as instructed. The superscripts "▲", "▽", and "$^{-}$" indicate an increase, decrease, or inapplicability in the evaluation score after fine-tuning, respectively.

in comparison to task T1, tasks T2 and T3 present greater challenges in instruction following. For example, several VLMs (i.e., LlaVA, InstructBLIP, Otter), without fine-tuning, fail to follow instructions to generate positional information for evaluating IOU, indicated by the symbol "∅." Additionally, all evaluated VLMs have F1-State scores below 50%, indicating that their predictions are even worse than random guessing. One potential reason task T3 is complex, and relies on task T2 is that it requires positional information to detect states, as well as insights from task T1 to understand the LEGO components and their relationships in the scene. We will explore this in future work, as this study primarily focuses on proposing the tasks rather than complex modeling.

5.2 Data Quality Assessment

To ensure the quality of the generated data for tasks T2 and T3, we sampled 100 data samples and conducted quality assessments for each task. We added bounding boxes based on the coordinates and asked three annotators to evaluate the generated data quality, focusing on coordinates, entities, and negative samples, based on the following criteria: (1) **Entity disambiguity** measures how clearly a target entity is relevant to the scene in T2, with scores of 0, 1, and 2 indicating low, medium, and high disambiguation, respectively. (2) **Boundary precision** measures how accurately a bounding box encloses a target object in T2, with scores of 0, 1, and 2 indicating low, medium, and high precision, respectively. (3) **State relevance** measures whether the generated parts in an image are a relevant variation of the original image in T3. (4) **State identifiability** measures whether the generated parts in an image are a recognizable variation of the original image in T3.

As shown in Figure 7, the average entity disambiguation and boundary precision scores are 1.74 and 1.25 out of 2 with the Kappa values [39] of 0.60 and 0.61, respectively.

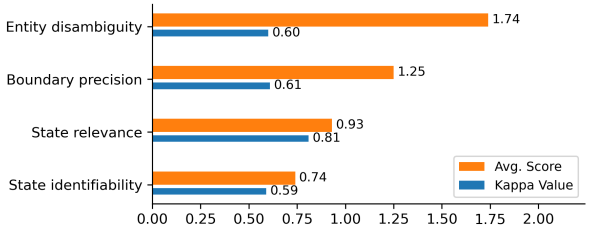


Figure 7: Human assessment of data quality.

This suggests that annotators reached a moderate agreement on the clarity of the evaluated objects in relation to the corresponding scenes. The average state relevance and identifiability scores are 0.93 and 0.74 out of 1.00, respectively, with the Kappa scores 0.81 and 0.59. This indicates that annotators reached near-perfect agreement on the relevance of the generated states to the original while showing moderate agreement on the recognizable variation of those states. Besides, the first five data points mentioned in the guidelines were used as a sanity check to ensure that the annotators correctly understood and applied the evaluation criteria. This ensures the quality of the generated states.

6 Analysis

6.1 Impact of Query Construction

We explore the impact of query construction on the QWEN-VL model in the early phase to identify the optimal configuration for achieving the best inference results.

To this end, we conducted experiments on various query settings to determine which components most significantly influence performance. The model was chosen for this study due to its strong results under our base setting, which included both instruction directives and in-context examples.

Setting		T1			T2		T3	
Instruction	Example	T_ACC ↑	ROUGE ↑	BLEU ↑	E_ACC ↑	IOU ↑	F1-State ↑	FPR ↓
✓	✓	**11.90**	**29.57**	**5.19**	**32.41**	**25.60**	**39.77**	97.99
✗	✓	6.84	24.63	4.58	5.56	22.01	34.00	97.50
✓	✗	10.43	23.00	4.24	26.37	15.55	34.50	97.20
✗	✗	5.27	11.02	1.30	3.70	0.10	25.50	**98.40**

Table 3: A study of query construction results across various tasks and metrics. The symbol ↑ indicates higher is better, ↓ indicates lower is better, and values in bold represent the highest results.

As shown in Table 3, our findings highlight the critical role of query design across different tasks. The results indicate that providing both instructions and examples yields the best overall performance, demonstrating their complementary roles in guiding inference. In the scene understanding task (T1), the presence of instructions plays a critical role in improving accuracy. When instructions are included, theme accuracy (T_ACC) increases significantly, suggesting that clear guidance helps the model better understand scene context and thematic elements. Furthermore, both ROUGE and BLEU scores, which evaluate the structure and coherence of generated outputs, improve when at least one of the two components (instructions or examples) is present. The lowest performance in this task occurs when neither component is provided, confirming that structured input is essential for meaningful inference.

In the object detection task (T2), the results suggest that examples contribute more to entity recognition and output formatting than instructions alone. The entity accuracy (E_ACC) drops sharply from 32.41% to 5.56% when examples are removed, even when instructions remain. Similarly, IoU, which measures how well the output adheres to the expected format, significantly decreases in the absence of examples. This highlights that examples provide concrete reference points, enabling the model to generate well-structured and accurate entity labels. In the state detection task (T3), despite improvements in other metrics with enhanced query settings, the false positive rate (FPR) remains consistently high (above 97%) across all conditions. This suggests that the model has a strong affirmative bias, likely defaulting to generating positive responses regardless of query refinements. While instructions and examples improve structured outputs, they do not sufficiently mitigate the issue of excessive false positives.

These findings emphasize the importance of query design in optimizing model performance. Combining instructions with instruction directives and in-context examples is crucial for achieving high accuracy, structured outputs, and reliable entity recognition. However, the persistent issue of high FPR suggests the need for further refinements, such as introducing negative examples, confidence calibration, or additional fine-tuning strategies to reduce overconfidence in affirmative responses. Future work should explore these approaches to enhance the model's robustness and reliability in diverse inference tasks.

6.2 Case Study of Fine-grained VLM Challenges

In this section, we present illustrative examples to highlight the capabilities and challenges in fine-grained vision-language understanding, particularly in achieving overlap with the reference and accurately recognizing entities and their properties, such as size, color, and etc.

Table 4 presents an example showcasing the intuitive results of the prevailing VLMs on scene understanding. Compared with the reference, existing VLM models face key challenges, including: (1) *Vague and generic assembly instructions.* The mPLUG-OWL2 and BLIP2 models produce broad, non-specific placement directives (e.g., "Place the mailbox front 2x2 next to the block on the table" or "Put the mailbox front 2x2 horizontally on the table, clasp to the back"), falling short of the precise, step-by-step guidance found in the reference. The Otter's response is even more superficial, merely stating "build structure" without actionable detail. (2) *Struggles with fine–grained entity recognition and attribute grounding.* While models like Qwen-VL and MiniGPT-v2 correctly mention the "transparent mailbox front 2x2," other models such as InstructBLIP, overlook critical attributes (e.g., referring only to "it" or "the mailbox front"). The LlaVA model, despite its verbosity, hallucinates details (e.g., "red liquid inside" and "blue sky") absent from the scene, highlighting confusion in entity identification and property association. (3) *Tendency to hallucinate or misinterpret assembly steps.* The MiniGPT-v2 and GPT-4o models introduce extraneous or incorrect actions, such as "aligning with the short side of the plate" or "inserting the small shaft into the round brick 1x1," which are not part of the actual assembly process. This reveals a shallow grasp of both scene context and the logical flow of assembly operations. Overall, while some models can partially identify the correct entities and occasionally mention relevant properties, none consistently generate instructions that are both factually accurate and contextually appropriate for fine-grained assembly tasks. To address this gap, we benchmark the VLM tasks and provide comprehensive resources to advance research in this field.

6.3 Non-VLM Baseline Comparison

Regarding the object detection task (T2), classical perception-only models (e.g., the YOLO series) achieve strong performance on predefined, fixed label sets (IoU 0.62–0.73 on 50 classes). However, they are less suitable for our setting, where the space of descriptive labels is large and each class has limited training examples. In contrast, our benchmark model achieves comparable performance (IoU 0.61) while supporting descriptive object outputs without restricting the label space (see an example in Table 5). Besides, our setting requires models to jointly handle all tasks, integrating perception, reasoning, and the generation of spatial information in a language-compatible format, thereby encouraging the training of a unified architecture.

6.4 Fine-tuning Performance Degradation

While fine-tuning improves performance in most settings, we observe degradation in some cases, such as Qwen-VL on T3 and MiniGPT-v2 on T2/T3 (see Table 2). We attribute this to *over-specialization to surface patterns* and *catastrophic forgetting* in models not designed for precise spatial supervision. LEGO tasks require pixel-level alignment and relational consistency, which can conflict

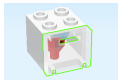
Query: [*grounding*] After I find *1 transparent mailbox front 2x2*. Tell me what to do for the current situation.  **Reference of Assembly Instruction:** Insert the two tabs at the bottom of the mailbox casing into the two small notches on the mailbox front.
mPLUG-OWL2: Place the *mailbox front 2x2* next to the block on the table .
BLIP2: Put the *mailbox front 2x2* horizontally on the table, clasp to the back. Put the mailbox back 2x2 vertically on top, slope to the left.
LlaVA: In the image, there is a small Lego-like object with a red liquid inside of it. The object is placed in a blue sky, and it appears to be a 3D model. To create a transparent mailbox, you need to find a *transparent mailbox front that is 2x2* in size. Once you find it, you can use it as the base for your transparent mailbox. You can then place the small Lego-like object inside the mailbox to complete the scene.
Qwen-VL: Put the *transparent mailbox front 2x2* on the right side of the white arch.
InstructBLIP: Place it horizontally on the table, clasp to the back.
MiniGPT-v2: Here's what you need to do with the pieces you have collected: 1. Place the *transparent mailbox front 2x2* on top of the plate 1x2, aligned with the short side of the plate. Now you have completed the current situation. Let me know if you need help with anything else!
Otter: In view of the reasoning 'A mailbox is small and square with a see through window.', the optimal answer should be: build structure.
MiniGPT-4: Attach the *transparent mailbox front* to the base structure, aligning the tabs with the notches.
GPT-4o: Ensure the *transparent mailbox front 2x2* is securely attached by inserting the small shaft into the round brick 1x1 before proceeding with the next step.

Table 4: An example for understanding VLM challenges.

Query	Please identify and highlight the positions of the following 2 pieces: 1 olive green pair of legs, and 1 dark green, mossy green and bright yellowish green upper part of the body.
Output	1 olive green pair of legs {<13><17><24><37>} 1 dark green, mossy green and bright yellowish green upper part of the body {<24><17><39><37>}

Table 5: An example of language-compatible object detection.

with language-dominant pretraining objectives. Similar effects have been reported in prior VLM fine-tuning studies [54].

7 Related Work

7.1 Vision-Language Datasets for Assembly

Various datasets have been developed to enhance assembly tasks by supporting key capabilities T1, T2, and T3. A comparative summary is presented in Table 6.

Dataset	T1	T2	T3	Size
COIN	✓	✗	✓	11,827
HowTo100M	✓	✗	✓	23,611
TEACh	✓	✗	✓	3,215
MIMIC-IT	✓	✗	✗	2.8M
HoloAssist	✓	✗	✓	350
EPIC-KITCHENS	✗	✓	✓	89,977
Assembly101	✓	✗	✓	4,321
Cross-task	✗	✓	✓	4,713
RareAct	✗	✓	✓	7,607
LEGO-VLM (Ours)	✓	✓	✓	35,612

Table 6: Compariable datasets for assembly tasks towards AR training regarding the cabablities of T1, T2, T3. The symbols ✓and ✗indicate the presence or absence of each capability.

COIN provides a strong foundation for sequential task analysis with comprehensive annotations, aiding research in multimodal learning [46]. HoloAssist captures egocentric human-AI interactions using mixed-reality headsets, offering valuable real-world insights [50]. HowTo100M, with its extensive collection of 136 million video clips and transcribed narrations, is highly effective for text-to-video retrieval but lacks fine-grained object annotations [33]. TEACh focuses on interactive dialogues in domestic environments, improving dialogue modeling but not prioritizing detailed object detection [35].

While some datasets excel in action recognition, they often lack the procedural depth necessary for effective training. Assembly101, with over 4,000 toy assembly videos, does not include step-by-step instructional details [40]. CrossTask facilitates weakly supervised learning by leveraging narrations and step lists but lacks temporal annotations to clarify action sequences [59]. EPIC-KITCHENS provides extensive annotations on unscripted kitchen activities, yet it does not offer structured procedural guidance [15]. RareAct presents unique interactions, challenging models to interpret complex actions without explicit instructions [32]. Stanescu et al. [42] introduce a state-aware prior that significantly improves object detection in assembly tasks like furniture and Lego construction. However, it lacks textual descriptions of object states and does not address fine-grained detection [43]. Meanwhile, MIMIC-IT contributes 2.8 million multimodal instruction-response pairs, enriching conversational modeling but lacking scenario-specific descriptions that would enhance task guidance [26].

To bridge these gaps, we conduct a comparative analysis evaluating dataset coverage of essential components (objects, states, and scenes) critical for developing AR-guided assistants. Additionally, we introduce a new simulated dataset featuring nearly 400 objects, systematically alternating between object detection and assembly functions. This dataset is designed to optimize the training and evaluation of advanced VLMs, improving their ability to handle fine-grained assembly tasks.

7.2 Fine-grained Vision-Language Models

Recent advancements in vision-language models (VLMs) have broadened their applications, enhancing accessibility, search indexing, and interactive content retrieval [16, 17, 19, 55]. These models generate captions, describe images [23, 57], facilitate visual QA [4]. For instance, they can accurately identify objects in images based on phrases like "the red car" [45]. Furthermore, these models facilitate conversational interactions about visual elements [7], which makes virtual assistants more adept at handling inquiries related to images and videos, which makes virtual assistants more adept at handling inquiries related to images and videos. Additionally,

VLMs can support multilingual content [21], benefiting assistive technologies and education [11, 47].

Despite progress, VLMs often lack fine-grained modeling for scene understanding, object recognition, and error detection. While CLIP demonstrates strong generalization [37], models like EfficientVLM [48], MiniGPT-v2 [8], and Qwen-VL [3] struggle with intricate tasks. OSCAR [29] and VisionLLM [49] improve image-text alignment but face challenges in context-specific adaptation. The Otter model advances sequential task management yet falls short in detailed analysis [27].

Efforts in procedural video representation [56] and instructional task graphs [2] highlight potential VLM integration but reveal gaps in error correction. Hence, we propose a vision-language architecture optimized for fine-grained tasks, advancing training and benchmarking of VLMs.

8 Conclusion

Vision-language models have achieved significant success in general vision-language tasks, but they often struggle with fine-grained understanding, particularly in structured instructional scenarios. This research explores fine-grained vision-language modeling for manual-guided LEGO assembly tasks, focusing on scene understanding, object detection, and state detection. We created a specialized multimodal dataset from LEGO instruction manuals, designing fine-grained tasks to evaluate VLM performance in tracking assembly sequences and interpreting instructions. Using a unified architecture, we assessed several models and found that they struggle with fine-grained scene understanding and assembly state detection, highlighting the need for further benchmarking and improvement. Beyond technical advancements, our research aims to empower blind and visually impaired individuals by enabling AI-driven learning tools, promoting greater accessibility and independence.

Safe and Responsible Innovation Statement

Implications. The implications span beyond research: such systems could democratize hands-on learning by enabling visually impaired users to access assembly tasks through multimodal feedback, and could revolutionize technical education and vocational training by offering consistent, language-grounded support. Our findings further highlight the current limitations of VLMs in precise reasoning and verification, pointing toward future directions in responsible AI development that benefit society at large—including learners, workers, and underserved communities. The architecture is intentionally lightweight, serving as a unified evaluation wrapper that enables fair comparison across diverse VLMs without introducing confounding model-specific designs. Our motivation is not to propose a new architecture, but to expose capability gaps in existing multimodal assistants under a consistent interface.

Limitations. We also acknowledge the limitations, including: First, simulated evaluation scenarios. While T1 is constructed from real-world data, T2 and T3 are generated using rule-based programs, rather than diverse real-world assembly scenarios. Moreover, we only consider 2D settings rather than 3D environments, which represents a major gap between our benchmarks and real-world scenarios. Second, annotation bias. In T2, we only use model-assisted annotation to create a scalable benchmark dataset without full-scale human correction. Although we sampled 100 data instances for human assessment of data quality, human correction was not performed on the entire dataset. Last, fidelity of evaluation metrics. While BLEU and ROUGE provide complementary measures of lexical overlap and semantic alignment, they do not fully capture procedural correctness or the functional equivalence of assembly instructions.

Future work. We plan to extend this benchmark towards more realistic scenarios, including the construction of datasets collected from users interactions with LEGO assembly assistants in XR environments. We also plan to develop more comprehensive evaluation metrics that better reflect task-level performance, including procedural accuracy, functional validity, and user-centered effectiveness.

Ethical Considerations. We recognize the ethical implications of developing VLMs for user interactions, so addressing these concerns is essential. To ensure ethical standards, we benchmark open-source VLMs and prioritize user-centric design, such as inclusiveness for diverse users.

Acknowledgments

We would like to thank all reviewers for their time, effort, and constructive comments on our paper. We appreciate their feedback and have addressed their concerns in the camera-ready version.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Kumar Ashutosh, Santhosh Kumar Ramakrishnan, Triantafyllos Afouras, and Kristen Grauman. 2024. Video-mined task graphs for keystep recognition in instructional videos. In *NeurIPS*. arXiv:2307.08763 [cs.CV]

[3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A versatile vision-language model for understanding, localization. *Text Reading, and Beyond 2* (2023).

[4] Yakoub Bazi, Mohamad Mahmoud Al Rahhal, Laila Bashmal, and Mansour Zuair. 2023. Vision-language model for visual question answering in medical imagery. *Bioengineering* 10, 3 (2023), 380.

[5] Fahmi Bellalouna, Mika Luimula, Panagiotis Markopoulos, Evangelos Markopoulos, and Franco Zipperling. 2020. FiAAR: an augmented reality firetruck equipment assembly and configuration assistant technology. In *CogInfoCom*. IEEE, 000237–000244.

[6] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. 2024. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *CVPR*. 14455–14465.

[7] Feilong Chen, Duzhen Zhang, Xiuyi Chen, Jing Shi, Shuang Xu, and Bo Xu. 2022. Unsupervised and pseudo-supervised vision-language alignment in visual dialog. In *MM*. 4142–4153.

[8] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023. Minigpt-v2: Large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478* (2023).

[9] Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. 2024. Are we on the right way for evaluating large vision-language models? *NeurIPS* 37 (2024), 27056–27087.

[10] An-Chieh Cheng, Hongxu Yin, Yang Fu, Qiushan Guo, Ruihan Yang, Jan Kautz, Xiaolong Wang, and Sifei Liu. 2024. Spatialrgpt: Grounded spatial reasoning in vision-language models. *NeurIPS* 37 (2024), 135062–135093.

[11] Ta-Chung Chi, Minmin Shen, Mihail Eric, Seokhwan Kim, and Dilek Hakkani-Tur. 2020. Just ask: An interactive learning framework for vision and language navigation. In *AAAI*. 2459–2466.

[12] M. Chu, Y. Tsuji, and S. Kato. 2014. Hue-based object detection in color images. In *CVPR*. 148–153.

[13] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen,

et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261* (2025).
[14] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. arXiv:2305.06500 [cs.CV]
[15] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2018. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*.
[16] Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. 2022. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936* (2022).
[17] Xiaoran Fan, Tao Ji, Shuo Li, Senjie Jin, Sirui Song, Junke Wang, Boyang Hong, Lu Chen, Guodong Zheng, Ming Zhang, et al. 2024. Poly-visual-expert vision-language models. In *COLM*.
[18] Markus Funk, Andreas Bächler, Liane Bächler, Thomas Kosch, Thomas Heidenreich, and Albrecht Schmidt. 2017. Working with augmented reality? A long-term analysis of in-situ instructions at the assembly workplace. In *PETRA*. 222–229.
[19] Zhe Gan, Linjie Li, Chunyuan Li, Lijuan Wang, Zicheng Liu, Jianfeng Gao, et al. 2022. Vision-language pre-training: Basics, recent advances, and future trends. *Foundations and Trends® in Computer Graphics and Vision* 14, 3–4 (2022), 163–352.
[20] Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. 2024. Physically grounded vision-language models for robotic manipulation. In *ICRA*. IEEE, 12462–12469.
[21] Jeremy Gwinnup and Kevin Duh. 2023. A survey of vision-language pre-training from the lens of multimodal machine translation. *arXiv preprint arXiv:2306.07198* (2023).
[22] Wenyi Hong, Wenmeng Yu, Xiaotao Gu, Guo Wang, Guobing Gan, Haomiao Tang, Jiale Cheng, Ji Qi, Junhui Ji, Lihang Pan, et al. 2025. GLM-4.1v-thinking: Towards versatile multimodal reasoning with scalable reinforcement learning. *arXiv preprint arXiv:2507.01006* (2025).
[23] Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. 2022. Scaling up vision-language pre-training for image captioning. In *CVPR*. 17980–17989.
[24] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. 2024. What matters when building vision-language models? *NeurIPS* 37 (2024), 87874–87907.
[25] LEGO. 2024. LEGO audio instructions. https://legoaudioinstructions.com/instructions. Accessed: 12-12-2024.
[26] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Jingkang Yang, Chunyuan Li, and Ziwei Liu. 2023. Mimic-it: Multi-modal in-context instruction tuning. *arXiv preprint arXiv:2306.05425* (2023).
[27] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. 2023. Otter: A multi-modal model with in-context instruction tuning. arXiv:2305.03726 [cs.CV]
[28] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*. arXiv:2301.12597 [cs.CV]
[29] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, et al. 2020. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *ECCV*. Springer, 121–137.
[30] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. In *NeurIPS*.
[31] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *ICLR*. https://arxiv.org/abs/1711.05101
[32] Antoine Miech, Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2020. RareAct: A video dataset of unusual interactions. *arxiv:2008.01018* (2020).
[33] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *CVPR*. arXiv:1906.03327 [cs.CV]
[34] OpenAI. 2024. GPT-4o. https://openai.com/index/hello-gpt-4o/.
[35] Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. 2021. TEACh: Task-driven embodied agents that chat. In *AAAI*. arXiv:2110.00534 [cs.CV]
[36] QwenTeam. 2024. Qwen2.5: A party of foundation models. https://qwenlm.github.io/blog/qwen2.5/
[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *ICML*. PMLR, 8748–8763.
[38] Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. 2024. Vision language models are blind. In *ACCV*. 18–34.
[39] Scikit-learn. 2024. Fleiss Kappa. https://www.statsmodels.org/dev/generated/statsmodels.stats.inter_rater.fleiss_kappa.html.
[40] 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 *CVPR*. 21096–21106.
[41] Haozhan Shen, Peng Liu, Jingcheng Li, Chunxin Fang, Yibo Ma, Jiajia Liao, Qiaoli Shen, Zilun Zhang, Kangjia Zhao, Qianqian Zhang, et al. 2025. Vlm-r1: A stable and generalizable r1-style large vision-language model. *arXiv preprint arXiv:2504.07615* (2025).
[42] Ana Stanescu, Peter Mohr, Mateusz Kozinski, Shohei Mori, Dieter Schmalstieg, and Denis Kalkofen. 2023. State-aware configuration detection for augmented reality step-by-step tutorials. In *ISMAR*. IEEE, 157–166.
[43] Ana Stanescu, Peter Mohr, Mateusz Kozinski, Shohei Mori, Dieter Schmalstieg, and Denis Kalkofen. 2023. State-aware configuration detection for augmented reality step-by-step tutorials. In *ISMAR*. 157–166. doi:10.1109/ISMAR59233.2023.00030
[44] Statsmodels. 2024. KMeans. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html.
[45] Sanjay Subramanian, William Merrill, Trevor Darrell, Matt Gardner, Sameer Singh, and Anna Rohrbach. 2022. ReCLIP: A strong zero-shot baseline for referring expression comprehension. In *ACL*.
[46] 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 *CVPR*. 1207–1216.
[47] Bryan Wang, Yuliang Li, Zhaoyang Lv, Haijun Xia, Yan Xu, and Raj Sodhi. 2024. LAVE: LLM-powered agent assistance and language augmentation for video editing. *arXiv preprint arXiv:2402.10294* (2024).
[48] Tiannan Wang, Wangchunshu Zhou, Yan Zeng, and Xinsong Zhang. 2023. EfficientVLM: Fast and accurate vision-language models via knowledge distillation and modal-adaptive pruning. In *ACL*. 13899–13913.
[49] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. 2024. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *Advances in Neural Information Processing Systems* 36 (2024).
[50] 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. 2023. HoloAssist: An egocentric human interaction dataset for interactive ai assistants in the real world. In *CVPR*. arXiv:2309.17024 [cs.CV]
[51] Haoran Wei, Lingyu Kong, Jinyue Chen, Liang Zhao, Zheng Ge, Jinrong Yang, Jianjian Sun, Chunrui Han, and Xiangyu Zhang. 2024. Vary: Scaling up the vision vocabulary for large vision-language model. In *ECCV*. Springer, 408–424.
[52] Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Anwen Hu, Haowei Liu, Qi Qian, Ji Zhang, and Fei Huang. 2024. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration. In *CVPR*. 13040–13051.
[53] Yingchao You, Ze Ji, Xintong Yang, and Ying Liu. 2022. From human-human collaboration to human-robot collaboration: automated generation of assembly task knowledge model. In *ICAC*. IEEE, 1–6.
[54] Yuexiang Zhai, Shengbang Tong, Xiao Li, Mu Cai, Qing Qu, Yong Jae Lee, and Yi Ma. 2024. Investigating the catastrophic forgetting in multimodal large language model fine-tuning. In *Conference on Parsimony and Learning*. PMLR, 202–227.
[55] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. 2024. Vision-language models for vision tasks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).
[56] Yiwu Zhong, Licheng Yu, Yang Bai, Shangwen Li, Xueting Yan, and Yin Li. 2023. Learning procedure-aware video representation from instructional videos and their narrations. In *CVPR*. arXiv:2303.17839 [cs.CV]
[57] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and vqa. In *AAAI*. 13041–13049.
[58] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. 2023. MiniGPT-4: Enhancing vision-language understanding with advanced large language models. arXiv:2304.10592 [cs.CV]
[59] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. 2019. Cross-task weakly supervised learning from instructional videos. In *CVPR*. arXiv:1903.08225 [cs.CV]

A System Implementation Details

During data synthesis, we utilized MiniGPT4-v2 [8] as the VLM, employing straightforward rules and human correction to create vision-language data. To evaluate the performance of benchmark models, we randomly sampled 25% of the dataset to serve as the test set. All inference and fine-tuning experiments were conducted on NVIDIA A100 SXM4 40GB GPUs. We set the learning rate to 1e-5 and the batch size to 16 for our experiments. The AdamW optimizer [31] was employed to facilitate efficient training, which spanned 3 epochs.

B Query Templates

We investigate three fine-grained vision-language tasks and have devised specific query templates for each, as shown in Table 7.

T1: Scene Understanding **Token:** [grounding] **Selection Rule:** [step_symbol] with [step]. Instruction from this step. **Format-Directive:** Tell me what to do for the current situation.
T2: Object Detection **Token:** [object] **Selection Rule:** [step_symbol] with [eop], Word starts with "find" or "collect". Instruction from this step. **Format-Directive:** Providing the positions in the format: {[object] [Xleft][Ytop][Xright][Ybottom]}, with X and Y coordinates normalized to [0,100]. [Xleft] and [Ytop] for the top-left corner. [Xright] and [Ybottom] for the bottom-right corner.
T3: State Detection **Token:** [state] **Selection Rule:** [step_symbol] with [step]. Previous [step_symbol] is [eop]. Instruction from this step and the previous step. **Format-Directive:** Just tell me Yes or No.

Table 7: Query Templates for Fine-Grained VLM Tasks.

```
{
  "instruction_id": 95,
  "text": [ "Find 2 transparent red flat tiles 1x2." ],
  "VLM": {
    "img_path": "...",
    "parts_img_path": "None",
    "id_sign": "Car",
    "step_num": 28,
    "step_class": "eop",
    "other_sign": "AG",
    "background_rect": "None",
    "color_list": [],
    "bound_color": [],
    "task_label": "object",
    "query": "Find 2 transparent red flat tiles 1x2.",
           "MiniGPTv2_output":  "<p>2  transparent  red  flat  tiles</p>
{<28><17><73><50>}"
  },
  "img": "...",
  "entities": [
    { "this_line": "Find 2 transparent red flat tiles 1x2." },
    { "verb": "Find" },
    { "det": "2" },
    { "colorinfo": "transparent red" },
    { "theme": "flat tiles 1x2" },
    { "suffix": "." } ]
}
```

Table 8: An example of a JSON schema for integrated multimodal data.

C Data Schema

This section presents the JSON data schema for integrating multimodal data (visual and textual) and fine-tuning VLM. The schema defines key components that structure and organize data efficiently, ensuring seamless preprocessing before final compilation. The key fields related to VLM in the JSON schema include:

- **img_path**: Stores the local path of images downloaded via web scraping, pointing to the original files fetched from the web.
- **id_sign**: Extracted from the image filename, this field identifies the sub-assembly session (e.g., "CAR" in this example).
- **step_num**: Represents the sequence number of the assembly step (e.g., step 28), tracking the assembly's progression.
- **step_class**: A classification tag (e.g., "eop" for "end of part") indicating an object-finding step in the assembly process.
- **MiniGPTv2_output**: Stores processed output from MiniGPTv2, containing annotations or modifications essential for subsequent dataset creation.
- **parts_img_path**, **bound_color**, **color_list**: Reserved for assembly steps classified as "step" and crucial for state detection dataset generation: (1) **color_list**: Records all possible boundary colors used throughout the assembly session. (2) **bound_color**: Consolidates the most probable boundary colors through a voting mechanism, ensuring consistency across the dataset.

To support fine-tuning, our default JSON format follows the Llava model's fine-tuning data structure, ensuring an optimized and adaptable framework for general use cases. An example of the JSON schema for VLM fine-tuning is illustrated in Table 9. The full

```
{
  "id": "lego-60263-ocean-mini-submarine-readscr-12",
  "image": "sm01-island-0_step_Step%20%23%23E.png",
  "conversations": [
    {
      "from": "human",
       "value": "[grounding] After I find 1 reddish brown chest
bottom 2x4.
      Tell me what to do for the current situation."
    },
    {
      "from": "gpt",
      "value": "Put it horizontally on the table."
    } ]
}
```

Table 9: An example of a JSON schema for VLM fine-tuning.

dataset, comprising JSON and PNG files, occupies approximately 11.8 GB. For specialized requirements, such as those for the Otter model, we convert all text and image data into Parquet files.

D Case Study of Object Detection

Table 10 present an example of object detection task. We obtain the following observations: First, only InstructiBLIP and Otter generate the full description of the object "1 light purple brick 2x6" without disambiguation. Second, none of the models accurately predict the precise positions as outlined in the reference.

E Examples in Data Quality Assessment

Figure 8 and 9 illustrate examples of the quality assessment for generated data in object detection (T2) and state detection (T3), respectively. We randomly selected 100 data examples per task to ensure a fair assessment. Three annotators were provided with comprehensive guidelines on how to evaluate each dataset based on multiple criteria, accompanied by relevant examples. The first five data points mentioned in the guidelines were used as a sanity check to ensure that the annotators correctly understood and applied the evaluation criteria. If annotators fail the sanity check, they undergo retraining until they pass. We evaluate annotations using the average scores of four criteria (i.e., entity disambiguity, boundary precision, state relevance, and state identifiability) and calculate Kappa values to assess annotation agreement.

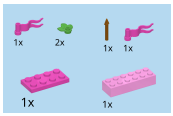
	Text	Image
Query	[object] Please identify and highlight the positions of 1 light purple brick 2x6. Providing the positions in the format: object [Xleft,Ytop,Xright,Ybottom], with X and Y coordinates normalized to [0,100]. Xleft and Ytop for the top-left corner. Xright and Ybottom for the bottom-right corner.	
Reference	1 light purple brick [58,15,92,35]	
mPLUG-OWL2	1 light purple brick[125,48,45,97]	
Fine-tuned	1 light purple brick [60,24,69,35]	
BLIP2	1 light purple brick [57,15,82,35] and 1 dark purple brick 2x6 [82,25,93,36]	
Fine-tuned	1 light purple brick [58,15,92,35]	
LLaVA	The light purple brick is located at the center of the image, surrounded by yellow and black bricks.	
Fine-tuned	1 light purple brick [62,15,87,35]	
QWEN-VL	[20,61,30,78]	
Fine-tuned	1 light purple brick[57,15,82,35]	
InstructiBLIP	1 light purple brick 2x6 [57,15,82,35]	
Fine-tuned	1 light purple brick [58,15,92,35]	
MiniGPT-v2	1 light purple brick [62,20,88,40]	
Fine-tuned	1 light purple brick [57,17,85,37]	
Otter	The 1 light purple brick 2x6 is on the left side of the image.	
MiniGPT-4	Sure! Here are the positions of the light purple brick in the image: Xleft = 0, Ytop = 25, Xright = 30, Ybottom = 50.	
GPT-4o	from PIL import Image # Load the image image_path =. . .	

Table 10: An example of object detection results. The object is highlighted in light purple, and its position is highlighted in blue.

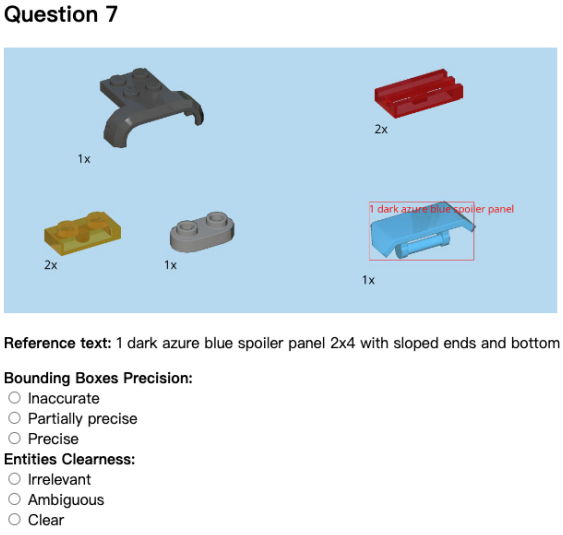


Figure 8: An example of evaluating data quality of task T2.

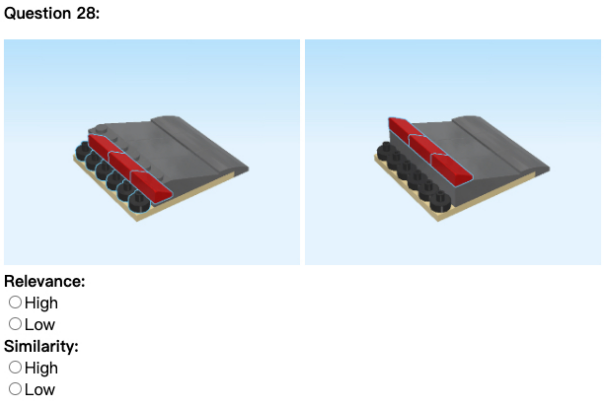


Figure 9: An example of evaluating data quality of task T2.