IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Robust Task Planning via Failure Detection using Scene Graph from Multi-view Images

Haechan Chong[1], Jongwon Lee[2], Hyemin Ahn[1]

***Abstract*—Recent robot task planners utilize large language models (LLMs) or vision-language models (VLMs) as a failure detector. These methods perform well by leveraging their semantic reasoning capabilities but often assume full environment understanding, which can lead to unreliable planning in complex scenes lacking explicit structural modeling. To address these limitations, we propose a novel multi-view scene understanding framework that explicitly models object-level relationships, enabling failure detection and effective task replanning. Our approach first captures multi-view images for comprehensive coverage, and generates local 2D scene graphs encoding object identities and relational information. Building on this, we introduce a model based on a graph neural network that merges the local 2D scene graphs into a unified representation. This process results in the unified scene graph, used to detect task success and identify failure causes. For each sub-task, our framework compares the unified scene graph with the expected scene graph predicted by the LLM during the task planning stage, identifying potential failure causes based on their deviations. These causes are then fed back into the LLM to facilitate effective replanning, thereby reducing repetitive failures and enhancing adaptability. We evaluate our framework on five real-world benchmark tasks to demonstrate its applicability. Separately, we compare failure detection and reasoning performance with other methods, showing the benefits of combining multi-view perception with explicit graph-based reasoning. More information can be found in https://sites.google.com/view/scrutinize-robot-manipulation**

***Index Terms*—Task and Motion Planning, Failure Detection and Recovery, AI-Enabled Robotics**

I. Introduction

EFFECTIVE robot task planning requires a precise understanding of dynamic and unstructured workspaces. To achieve this, the robot must not only follow high-level instructions, but also continuously interpret the current task context. Based on this, the robot can determine what changes it has made to the environment, whether it has failed to accomplish the task, and if so, identify the reason for the failure. Without such contextual awareness, it will be challenging for the robot to identify the main cause of failure, and to develop an appropriate strategy to correct the failure. This highlights the importance of task space reasoning not merely for execution, but also for robust failure detection and informed replanning.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2023-00211416).
[1] H. Chong and H. Ahn are with the Department of Electrical Engineering (EE), Pohang University of Science and Technology (POSTECH), Pohang, Korea (e-mail: atlantic0924@postech.ac.kr; hmahn@postech.ac.kr).
[2] J. Lee is with the Department of Computer Science and Engineering (CSE), Ulsan National Institute of Science and Technology (UNIST), Ulsan, Korea (e-mail: belluno@unist.ac.kr).

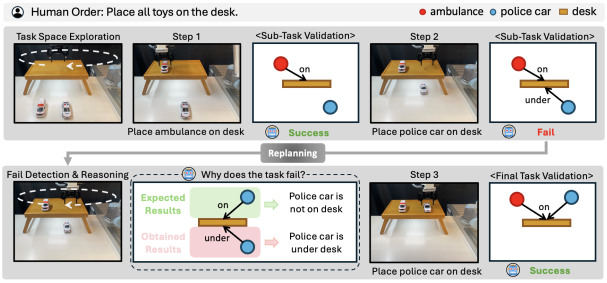

Fig. 1. Given human instruction, the robot explores the task space and starts executing the sub-tasks. After each sub-task, the predicted object relations are compared with the expected object relations to detect failures. If a failure and its reason are detected, the proposed framework completes the task and leverages the LLM to replan based on the failure reason.

To address this challenge, recent studies employed large models, such as large language models (LLMs) and vision-language models (VLMs), to plan robot behavior. To be specific, early large model-based task planners relied on single-view images from fixed cameras [1]–[4], but struggled to capture the full 3D environment, leading to reduced success in occluded or cluttered scenes. To address this, multi-view images were explored for richer spatial understanding [5]–[7], yet these methods often overlook fine-grained inter-object relationships critical for context-aware planning. Recent trends instead shift toward using large pretrained models to implicitly handle failure detection and replanning [8], [9].

To support explicit object-level reasoning, recent studies adopted scene graphs, which encode objects and their spatial, semantic, and functional relationships in a structured form. Several works showed promise in interpreting task outcomes and diagnosing failure, as well as replanning in robotic manipulation scenarios [10], [11]. When integrating scene graphs with LLMs or VLMs, better reasoning and context-aware task execution were available [12]–[14]. However, their reliance on a single-view image remains as a limitation.

In this paper, we propose a robust task planning framework that leverages multi-view scene understanding and an explicit model of inter-object relationships represented as a scene graph. This integrated approach enables accurate failure detection and effective replanning, ensuring reliable task execution in complex environments. Our contributions are as follows:

Multi-view scene understanding: We construct 2D scene graphs from multi-view images, which propose Graph Neural Network (GNN) then fuses into a unified representation. This unified scene graph is key to achieving a comprehensive understanding of dynamic and unstructured task spaces.

Failure detection using inter-object relationships: We accurately detect sub-task failures by comparing (1) the expected object relations predicted by an LLM, with (2) the actual

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

object relationships obtained from the unified scene graph. Note that expected object relations are obtained once at the initial planning stage, to reduce computational cost.

Failure reasoning for replanning: We used the unified scene graph to identify the main cause of the detected failure. The failure reasons are given to LLM, such that replanning can be conducted.

II. Related Work

Early robotic task planning and manipulation methods relied mainly on single-view visual input, such as RGB or RGB-D images. Various methods [1], [2], [15] used single-view images to perform manipulation tasks. While they are efficient and lightweight due to the reliance on a single image and minimal sensors, these approaches lack the spatial context and geometric understanding required for accurate 3D reasoning. To overcome this, recent research shifted towards using multi-view images to improve scene understanding. Some methods [5], [16] directly feed multi-view images into VLMs. Nevertheless, multi-view methods improve spatial and geometric understanding over single-view approaches but still struggle to reason about object relationships and dynamic interactions in complex scenes. To address this, our approach not only uses multi-view images to build a rich 3D scene representation but also explicitly models inter-object relationships via a scene graph, yielding more robust scene understanding and improved performance in robotic manipulation tasks.

Scene graphs provide a structured representation of objects and their spatial, semantic, and functional relationships, enabling robots to better understand and interact with complex environments. Recent works integrated scene graphs with LLMs for robotics [12]–[14], [17], demonstrating how scene-based understanding, when combined with symbolic and linguistic reasoning, can support more interactive and context-aware robotic behavior. In particular, some methods [10], [11] utilized scene graph representations as inputs to LLMs to interpret and verify manipulation results, especially to detect task failures. Compared with direct image input to VLMs, these methods provide structured, task-relevant information, improving planning robustness and failure detection. However, they rely on single-view input and depth-based relation inference using handcrafted rules (e.g., fixed thresholds), which often break down in cluttered or unseen environments. In contrast, modern AI frameworks reason adaptively using learned semantics, providing greater flexibility and robustness. Our framework adopts this approach by utilizing a 2D Scene Graph Generation (2DSGG) model [18], [19] to infer object relations from multi-view images.

Achieving high task success in robotic manipulation requires not only precise execution, but also accurate failure detection and effective replanning. Recent multi-modal systems [2], [5], [6], [15], [20] have improved failure detection by combining visual and linguistic signals, allowing sub-task validation and correction. However, merely retrying failed sub-tasks is often insufficient. Robust systems must reason about failures and adaptively replan with alternative strategies to complete the task. In this context, recent studies [8], [9] proposed multi-modal replanning frameworks that detect failures and revise task plans accordingly. In parallel, scene graph-based approaches [10], [11] also demonstrated potential to support both failure detection and replanning. Despite their effectiveness, these methods often incur considerable computational overhead because they frequently rely on large models, such as LLMs or VLMs, to perform sub-task-level and total-task validation. In contrast, our approach avoids re-invoking the LLM for every sub-task validation. Instead, failure detection is performed efficiently using expected object relationships, comparing the unified scene graph with predictions from a fine-tuned LLM during planning, significantly reducing computational cost.

III. Proposed Method

A. Notations

Let us denote a scene graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, which is obtained from a single 2D image observing the task space. Here, $\mathcal{V}$ is the set of nodes representing objects, $\mathcal{E}$ is the set of directed edges representing relationships between nodes, and $\mathcal{R}$ is the set of possible relation types. Each edge $\langle s, r, o \rangle \in \mathcal{E}$ encodes a relational triplet, where the subject node $s \in \mathcal{V}$ is connected to the object node $o \in \mathcal{V}$ via the relation label $r \in \mathcal{R}$. Let G^+ denote the redundant scene graph obtained by integrating all observed 2D scene graphs $\mathcal{G}$, and let G^* denote the unified scene graph produced through post-processing (Sec. III-E) of G^+. When the robot observes the task space only during a specific time, let us assume that a sub-graph $g \subseteq \mathcal{G}$ can be obtained as $g = (\mathcal{V}_g, \mathcal{E}_g, \mathcal{R}_g)$.

Notations below will be used throughout the section:

- $\mathbf{T}$: **Task Instruction**, language sentence describing the objective or the goal for manipulation.
- $\mathbf{F}$: **Object Feature Set**, defined as $\mathbf{F} = \{f_1, f_2, \ldots, f_n\}$, where each f_i is a feature vector associated with the i-th object in the scene.
- $\mathbf{R}$: **Object Relations**, representing spatial or semantic interactions between all objects in the task space. Each relation $r \in \mathcal{R} = \{$*on, next to, in, under*$\}$ serves as a predicate.

The object relations $\mathbf{R}$ can be represented in two complementary forms:

1) **Symbolic Representation**: A set $\langle s, r, o \rangle$, where each triplet describes a relation instance between two objects.
2) **Binary Tensor Representation**: As a binary tensor $\mathbf{R} \in \{0, 1\}^{n \times |\mathcal{R}| \times n}$. Let us define id_s as the index of the subject s, id_r as the index of relation type r, and id_o as the index of the object o. Then, $\mathbf{R}_{(id_s, id_r, id_o)} = 1$ if the relation $< s, r, o >$ present in the task space.

B. Task Planning Using LLM

To interpret natural language instructions from humans, our framework utilizes pretrained large language models (LLMs), such as LLaMA [21], to decompose a high-level task instruction $\mathbf{T}$ into a sequence of m executable sub-tasks, as shown in Fig. 2. To ensure robustness, we fine-tuned the LLaMA using multiple instruction phrasings for each task scenario,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

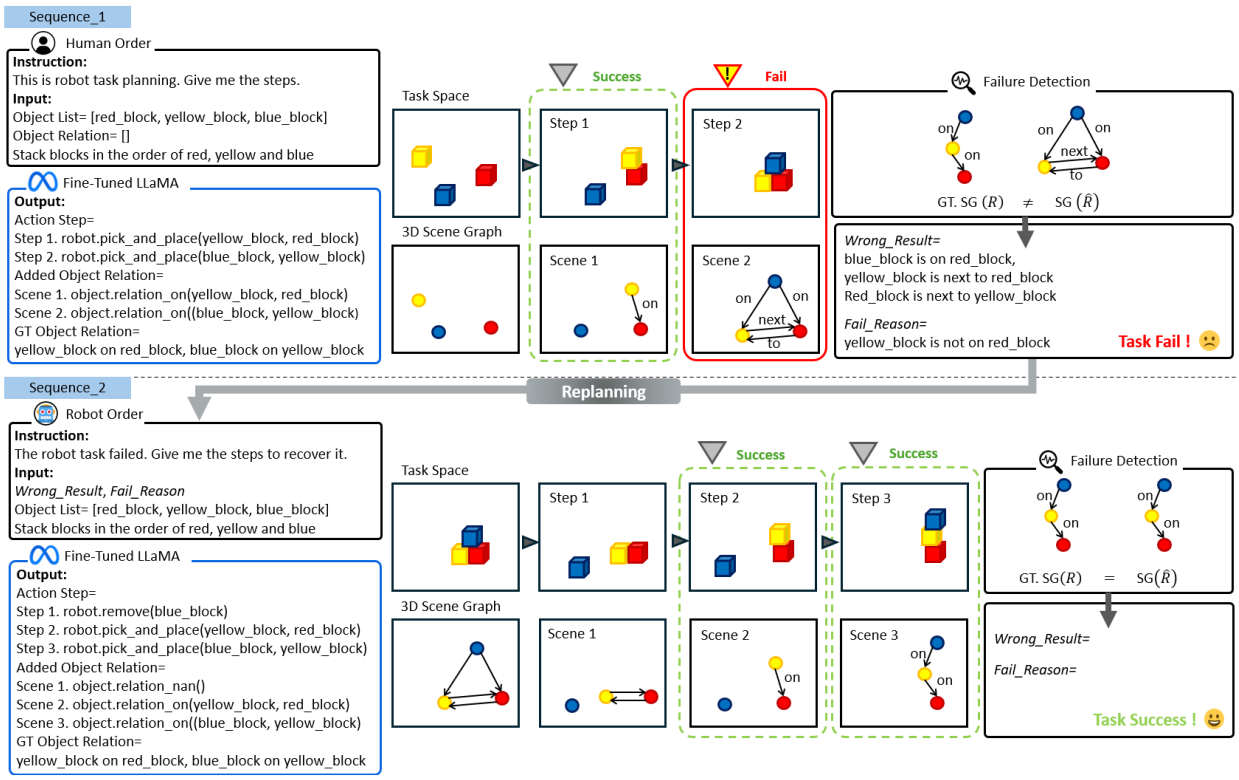

Fig. 2. An overview of our method. For robust task execution using LLM-based planning in sequence 1 and replanning sequence 2. Sequence 1: The robot plans and executes a task based on the human instruction, validating the success of each sub-task step-by-step. A failure is detected at Step 2 by comparing the predicted scene graph with the ground truth. Sequence 2: Using the identified wrong result and failure reason, the robot replans that leads to successful task completion, demonstrating the system's ability to recover from errors.

preventing it from relying on simple keyword-based mapping. The LLaMA model operates as follows:

$$\text{LLaMA}(\mathbf{T}, \mathbf{F}, \mathbf{R}) = [(T_1, R_1), \ldots, (T_m, R_m), R^*], \quad (1)$$

where each (T_m, R_m) pair represents the m-th sub-task and its associated expected object relations, and R^* denotes the final relational state expected after all tasks are completed.

Each sub-task T_m is represented in the text format `robot.{action}("A", "B")`, where `action` denotes a low-level manipulation primitive (e.g., *pick_and_place*), and `"A"` and `"B"` refer to specific objects identified by their class and track ID in the unified scene graph (e.g., `red_block_1`). Similarly, each relation R_m is expressed as `object.{relation}("A", "B")`, where `relation` corresponds to a spatial predicate (e.g., *on*). The final goal R^* is ultimately represented as a list of object-level relations described in natural language, such as `A is {relation} B`, providing an interpretable and human-readable summary of the desired scene configuration.

C. Task Validation and Replanning

Task failure is detected by measuring the deviation between the actual object relations $\hat{R}_m$ and the expected relational goals R_m and R^*, where all are represented as binary relation matrices. The ground-truth goals R_m and R^* are generated by the LLaMA model during planning, while $\hat{R}_m$ is derived from the scene graph predicted by the 2D Scene Graph Generation (2DSGG) model and iteratively refined via Enhanced Relational Graph Convolutional Network (E-RGCN) at each step m which will be more detailed in section III-D and III-E.

1) Task Validation: We validate task progress by comparing the actual and expected object relations across different execution steps.

Sub-Task Validation. To evaluate the success of sub-tasks from step 1 to $m-1$, we compute the element-wise difference between the expected and actual relation matrices at each step: $R_1 - \hat{R}_1, R_2 - \hat{R}_2, ..., R_{(m-1)} - \hat{R}_{(m-1)}$.

Final-Task Validation. To assess the final goal at step m, we compare: $R^* - \hat{R}_m$. Discrepancies in the execution of the task are identified through matrix subtraction shown in Fig. 3, where each element of the resulting matrix refers to the success or failure of a specific object relationship. Here, "kiwi is on the microwave" is **wrong result** (-1). In contrast, "kiwi is not in the microwave" corresponds to a **fail reason** $(+1)$.

2) Task Replanning: When a failure is detected, we extract the wrong result and the failure reason from the matrix subtraction $\mathbf{R} - \hat{R}_m$. This information, along with the list of detected objects $\mathbf{F}$ and the original human instruction $\mathbf{T}$, is compiled into a natural language prompt. This prompt is fed into our LLaMA to generate a revised task plan, and the robot executes the new sequence of actions. If the following action also fails, the entire process is repeated. This iterative process continues until the original high-level instruction succeeds. This enables the robot to recover from errors and continue task execution, as shown in the second sequence of Fig. 2.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

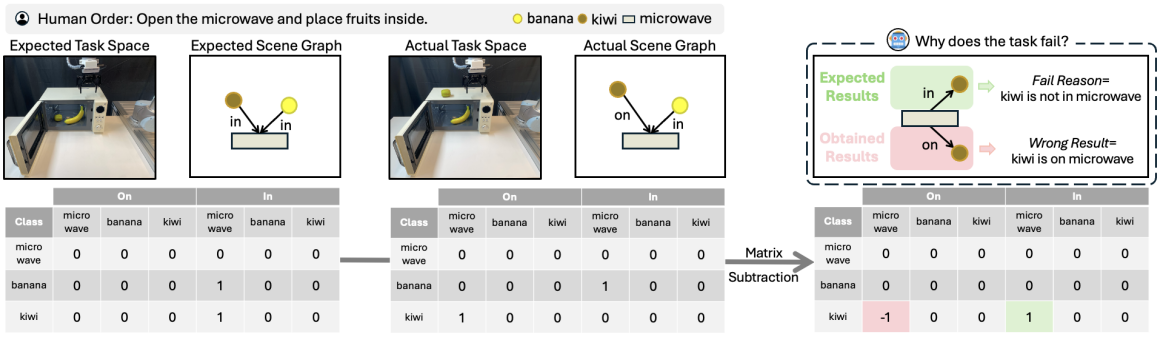

Fig. 3. Failure detection via matrix subtraction: ground truth scene graph from LLMs and scene graph from scene construction method are used to detect failure via matrix subtraction. An element value of -1 denotes a **wrong result**, meaning the predicted relation exists but is not expected. Conversely, an element value of $+1$ signifies a **fail reason**, indicating that an expected relation is not observed in the current scene.

D. 2D Scene Graph from Multi-Views

To perceive the task space using images captured from multiple viewpoints, we utilize an RGB-D camera mounted on the robot's end-effector. The robot follows a predefined trajectory, capturing a sequence of multi-view images. These images are processed by a pretrained 2D Scene Graph Generation (2DSGG) model [18], [19], which infers pairwise object relationships r as scene graph triplets $\langle s, r, o \rangle$. Original 2DSGG models, trained on datasets as Visual Genome [22], focus on human-centric and view-dependent relations and are limited in capturing consistent inter-object relations across viewpoints. To address this, we restrict the relation types $\mathcal{R}$ to four task-relevant categories defined in Sec. III-A, and train our 2DSGG model on a custom dataset, as detailed in the supplementary video.

Each image frame constructs the 2D scene graph $\mathcal{G}$ by identifying nodes $\mathcal{V}$, labeled with the object class cls, using YOLO as a 2D bounding box detector, and edges $\mathcal{E}$ using the 2DSGG model. For object tracking, we use DeepOCSort [23], which assigns each detected object a unique track ID i. For each object i, we record its observation interval $[t^i_{\text{start}}, t^i_{\text{end}}]$, during which depth maps are aligned with RGB images, and the robot's end-effector pose is recorded. Using this information, the object's 3D position is estimated per frame and averaged over the interval to obtain a final position $p_i = (p_i^x, p_i^y, p_i^z)$ in the robot's base frame. Finally, we make a feature for each object: $f_i = [i, cls, t_i^{\text{start}}, t_i^{\text{end}}, p_i] \in \mathbb{R}^8$, capturing its identity, class, temporal visibility, and position. This forms the nodes $\mathcal{V}$ of $\mathcal{G}$, providing a spatially grounded and temporally coherent multi-view representation of the task space.

For every m sub-tasks, the scene graph $\mathcal{G}_m$ is computed. To track changes, the scene graph $\mathcal{G}_m$ is aligned with $\mathcal{G}_{m-1}$ from the previous sub-task through object ID re-association. An object in the scene graph is matched based on an identical class label and minimal spatial distance. This allows the explicit and robust determination of how object relations have changed from the sub-task.

E. 3D Scene Recognition by E-RGCN

In practice, DeepOCSort often fails to track objects consistently, even with hyperparameter tuning. A common failure is to assign multiple IDs to the same object as the camera viewpoint changes over time. If all 2D scene graphs are integrated without resolving tracking failures, duplicate nodes will appear. This leads to incorrect inter-object relationship modeling, such as between nodes 1 and 7 in the redundant scene graph, as shown in Fig. 4.

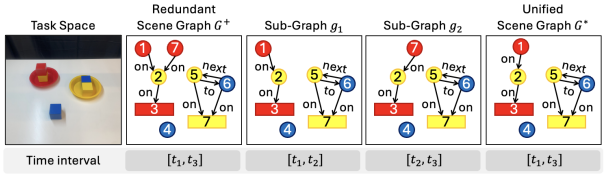

Fig. 4. Unified scene graph generation. The first image shows the real-world task space. The second image shows the redundant scene graph G^+, with unique tracking IDs on every detected object, including results from the tracker's failure. The third and fourth images show the temporally segmented sub-graphs g_1 and g_2, which capture local 2D scene graphs within different time intervals. The fifth image presents the final unified scene graph G^*, obtained by merging the nodes using the proposed E-RGCN.

To address this issue, we propose the Enhanced Relational Graph Convolutional Network (E-RGCN), an extension of the RGCN [24] adapted to our task-specific domain for link prediction. Link prediction is a binary classification task that classifies whether the link connecting two nodes is positive or negative. We define positive samples as connected nodes that refer to the same physical object with different IDs due to the tracker's error, and negative samples as distinct objects with different IDs. E-RGCN takes the redundant scene graph G^+ as input and predicts links between nodes to construct a unified scene graph G^*. Positively linked nodes are merged, reducing redundancy and enhancing semantic consistency, as illustrated in Fig. 4.

1) Encoder: The encoder builds node representations using a Graph Neural Network (GNN), which operates in a layer-wise manner. Each layer aggregates information from a node's neighbors and updates the node's feature representation accordingly. By stacking multiple layers, the encoder gradually refines node representations to capture both local structures and relation-specific patterns in the graph.

Vanilla RGCN. The standard RGCN encoder updates each node's representation by aggregating information from its neighbors based on relation types, as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_r^i}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right), \qquad (2)$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

where $h_i^{(l)}$ represents the hidden state of i-th node feature f_i in the l-th layer of the neural network. In our experiments, we use a 2-layer RGCN for training. Here, $\sigma(\cdot)$ denotes an element-wise activation function such as the Rectified Linear Unit(ReLU), $\mathcal{N}_r^i$ denotes a set of neighbor indices of the node feature f_i under relation $r \in \mathcal{R}$, and $c_{i,r}$ is a normalization constant that can either be pre-defined (e.g., $c_{i,r} = |\mathcal{N}_r^i|$) or learned. For relation-specific weight matrix $W_r^{(l)}$, we adapt the basis decomposition technique from the RGCN [24].

Sub-Graph Construction. To better capture the temporal dynamics in the scene, we divide the redundant scene graph G^+ into K time-specific sub-graphs g_k as shown in Fig. 4, where each sub-graph corresponds to a discrete temporal segment defined as follows:

$$\text{sorted}\left(\bigcup_i \{t^i_{\text{start}}, t^i_{\text{end}}\}\right) = (t_1, t_2, \ldots, t_{K+1}) \tag{3}$$

Each sub-graph g_k spans a time interval $[t_k, t_{k+1}]$, which is partitioned based on the points in time, where a new object ID i appears, or an existing one disappears. Therefore, an object f_i is included in g_k if it satisfies $t^i_{\text{start}} \leq t_k$ and $t^i_{\text{end}} \geq t_{k+1}$. This condition ensures that only objects persistently visible throughout the sub-interval are included in the corresponding sub-graph, enabling consistent relational modeling.

E-RGCN. Unlike the standard RGCN, which processes a static graph structure, our E-RGCN captures temporal variations by operating on sequential sub-graphs and aggregating their representations in a temporally-aware manner. The E-RGCN encoder begins by applying a multi-layer perceptron (MLP) to transform the input node features into a richer latent representation. This is followed by relational graph convolutions applied independently to each sub-graph g_k, allowing the model to capture context-aware relational structures within specific temporal segments. The node features within each sub-graph are updated as follows:

$$h_{i_{g_k}}^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}_r^{i_{g_k}}} \frac{1}{c_{i_{g_k},r}} W_r^{(l)} h_{j_{g_k}}^{(l)} + W_0^{(l)} h_{i_{g_k}}^{(l)}\right), \tag{4}$$

After encoding of all sub-graphs, a weighted aggregation is applied, where the weights are determined by the temporal duration of each sub-graph. Specifically, the temporal weighting coefficient α^k for the k-th sub-graph is computed as:

$$\alpha^k = \frac{e^{t_{k+1}-t_k}}{\sum_{p=1}^{K} e^{t_{p+1}-t_p}}, \tag{5}$$

These coefficients are used to aggregate the node representations across sub-graphs:

$$h_i^{(l+1)} = \sum_{k=1}^{K} \alpha^k \cdot h_{i_{g_k}}^{(l+1)}. \tag{6}$$

2) Decoder: To perform link prediction, we adapt the DistMult scoring function [25], a widely used method in knowledge graph embedding. Given a subject node s and an object node o, with corresponding embeddings e_s and e_o, the DistMult score is computed as:

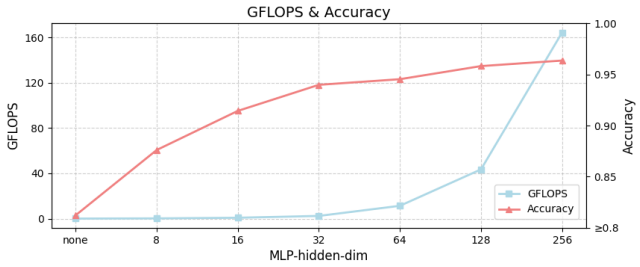

Fig. 5. GFLOPS and Accuracy (y-axis) with respect to the hidden state dimension of MLP (x-axis) in E-RGCN model.

$$\mathcal{D}(s, o) = e_s^\top M e_o, \tag{7}$$

Here, $M \in \mathbb{R}^{d\times d}$ is a learnable, non-diagonal matrix that models pairwise interactions, extending standard DistMult. To enhance training stability and generalization, we apply data augmentation by swapping subject s and object o in training data. This is valid since the link prediction task is symmetric and unaffected by the order of nodes.

F. Loss Function

In the task space, object pairs that share the same class label cls but are assigned different tracking IDs i are used to construct training samples for the link prediction task.

1) Binary Cross Entropy Loss: To train the E-RGCN, we use binary cross-entropy (BCE) loss, as the link prediction task is framed as a binary classification problem that predicts whether given pair nodes correspond to the same object:

$$\mathcal{L}_{BCE} = -\frac{1}{|\mathcal{P}| + |\mathcal{T}|} \sum_{(s,o,y)\in\mathcal{P}\cup\mathcal{T}} y \log \sigma\big(\mathcal{D}(s,o)\big) + (1-y)\log\big(1-\sigma\big(\mathcal{D}(s,o)\big)\big), \tag{8}$$

where $\mathcal{P}$ is the set of positive triplets and $\mathcal{T}$ is the set of negative triplets. y is the ground truth label: $y = 1$ for positive pairs (same object with different IDs), $y = 0$ for negative pairs (different objects with different IDs).

2) L1 Loss: To enforce symmetry in the learned similarity scores for positive pairs, we introduce an additional L1 loss term. This regularization encourages the model to produce consistent outputs when the subject and object nodes are swapped, as shown below:

$$\mathcal{L}_1 = \frac{1}{\mu} \sum_{(s,o,y=1)\in\mathcal{T}} |f(s,o) - f(o,s)|, \quad \text{where } s \neq o \tag{9}$$

3) Final Loss: The overall training loss is formulated as a weighted combination of the BCE loss and the L1 loss:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{BCE} + (1-\lambda)\cdot \mathcal{L}_1, \tag{10}$$

where $\lambda \in [0, 1]$ is a weighting coefficient that controls the trade-off between binary cross entropy loss and L1 loss.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE I
SUCCESS RATE (%) OF PLANNING AND REPLANNING ACROSS DIFFERENT ENVIRONMENT CONFIGURATIONS. SINGLE-INSTANCE REFERS TO SETTINGS WHERE ONLY ONE OBJECT EXISTS PER CLASS, WHILE MULTI-INSTANCE INCLUDES MULTIPLE OBJECTS PER CLASS.

Phase	Category	Task 1	Task 2	Task 3	Task 4	Task 5
Planning	Single-Instance	90.00	90.00	90.00	95.00	90.00
	Multi-Instance	80.00	90.00	-	-	-
	Total	85.00	90.00	90.00	95.00	90.00
Replanning	Single-Instance	95.00	95.00	80.00	95.00	70.00
	Multi-Instance	90.00	85.00	-	-	-
	Total	92.25	90.00	80.00	95.00	70.00

IV. EXPERIMENT

A. Finetuning Processes

Our framework integrates four distinct models for robot task planning and execution. The core planning model, LLaMA 3.2-8B, was fine-tuned using the LoRA method on one million task planning and replanning scenarios to generate plans from task instructions. YOLOv8 manages perception, trained on 10,000 images covering 16 object classes for our tabletop tasks. The 2DSGG model provides spatial and relational understanding, trained on YOLOv8 outputs and image data augmented with Visual Genome format annotations, encompassing 40,000 objects and 70,000 relationships across four types. Finally, the E-RGCN model performs unified scene graph construction, trained on 195 task scenarios derived from the 2DSGG model output, specifically utilizing 173 positive and 251 negative samples.

B. Real-World Robot Tasks

Our experiments are based on five real-world tabletop robotic manipulation tasks, designed to evaluate the robot's ability to handle objects under various conditions, as illustrated in Fig. 6. We evaluate performance across two phases, planning and replanning. Planning starts from a random configuration in the task space, while replanning starts from a randomly selected failure scenario, such as an incomplete step or a misplaced object. The success rates for both phases are presented in Table I, where success is strictly defined as the complete and correct execution of the task, excluding cases of partial completion. The Multi-Instance scenarios, which include multiple objects per class, are specifically implemented in Task 1 and Task 2 to introduce more complex relational reasoning challenges for E-RGCN, particularly in link prediction for both positive and negative samples. Each task involves 20 trials, with varying task space configurations used across trials. For Task 1 and Task 2, 10 trials are conducted in the Single-Instance setting and 10 trials in the Multi-Instance setting.

During the planning phase, our robot demonstrates consistently high performance, with success rates exceeding 85.00% across all tasks. In the replanning phase, a slight decrease in performance is observed, particularly in Task 5, where the success rate drops to 70.00%. This decline is attributed to the lower accuracy of the 2DSGG model, which affects the failure detection and reasoning processes. Despite this, the overall replanning performance remains competitive, highlighting the system's ability to adapt and recover during execution. Notably, the system performs well in both Single-Instance and Multiple-Instance object settings, demonstrating the strong generalization ability of our framework. However, in Task 1 and Task 2, a noticeable drop in performance is observed in the Multiple-Instance setting, likely due to the failure of the E-RGCN module in constructing an unified scene graph, as well as increased collisions during execution due to a cluttered environment. Nevertheless, considering the complexity of the setting, the performance remains a strong indication of the method's robustness.

TABLE II
ABLATION STUDY ON E-RGCN. SG DENOTES SCENE GRAPH, L_1 DENOTES L1 LOSS, AND DA DENOTES DATA AUGMENTATION.

models	Components				Results (per Seed)			
	MLP	SG	L_1	DA	Seed 1	Seed 2	Seed 3	Seed 4
RGCN					0.8437	0.8123	0.7861	0.7916
RGCN	✓				0.8981	0.8919	0.9012	0.8966
RGCN	✓	✓			0.9496	0.9508	0.9629	0.9305
RGCN	✓		✓		0.9270	0.9435	0.9696	0.9345
RGCN	✓			✓	0.9357	0.9289	0.9595	0.9285
RGCN	✓	✓	✓		0.9548	0.9526	0.9764	0.9384
RGCN	✓	✓		✓	0.9635	0.9544	0.9747	0.9464
RGCN	✓		✓	✓	0.9218	0.9453	0.9663	0.9365
E-RGCN	✓	✓	✓	✓	**0.9670**	**0.9581**	**0.9797**	**0.9523**

C. Ablation Studies of E-RGCN

In Fig. 5, we analyze the relationship between the performance of E-RGCN and its hidden layer dimension. In its x-axis, "none" refers to directly decoding the raw node vectors without using MLP, while the other values from 8 to 256 denote hidden dimension. The result shows that the link prediction accuracy saturates as the number of hidden dimension increases. However, the GFLOPS referring to the calculation cost increases rapidly. Among the tested configurations, the 128-dimension MLP-based E-RGCN exhibited reasonable accuracy compared to 256-dimension, lower than 0.54%. Based on this observation, we employ the 128-dimensional setting to ensure consistency for all experiments.

Table II shows the result of the ablation study, which compares the performance of E-RGCN with or without various components, such as sub-graph (SG), L_1 loss (L_1), and data augmentation (DA). These effects are evaluated by accuracy, which is obtained after combining the components with RGCN in various ways. To conduct this study, we employ 10-fold cross-validation to obtain more reliable performance estimates. This table shows the results over four distinct test data, obtained from different random seeds when splitting folds. The results highlight that the proposed E-RGCN is more effective than the baseline and all partial configurations.

D. Results on Failure Detection and Reasoning

We compare ours with prior **Methods** that use different forms of task space representation, including Palm-E [15], SuccessVQA [20], Reflect [10], Manipulate Anything (MA) [5], and AHA [16]. They use either a VLM or an LLM for failure detection and reasoning. Reflect and MA employ the state-of-the-art ChatGPT-4o model [26], whereas the others rely on proprietary VLMs that are not publicly available. To ensure a consistent evaluation setting, we apply ChatGPT-4o

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

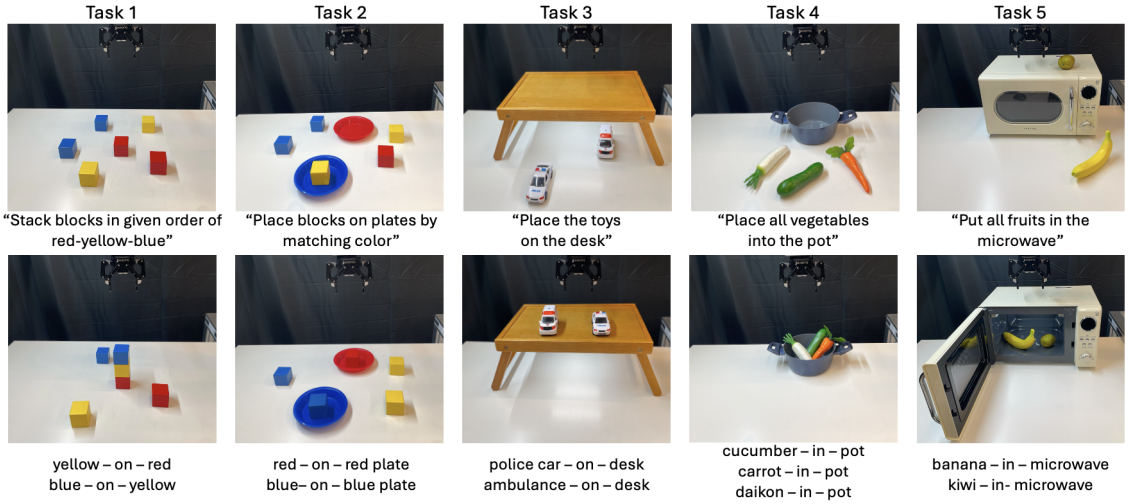

Fig. 6. Real-World Robot Tasks. (Top) Initial states of five tasks along with their corresponding instructions. (Bottom) Resulting states of the five tasks, showing the expected object relations.

TABLE III

COMPARISON OF FAILURE DETECTION (D) AND REASONING (R) ACCURACY ACROSS TASK VALIDATION SCENARIOS (%). THE *Harsh* SETTING REFERS TO ENVIRONMENTS IN WHICH OBJECTS ARE OCCLUDED OR PARTIALLY HIDDEN WITHIN THE VIEW FRAME. CAPABILITY TYPES INCLUDE SINGLE VIEW (SV), MULTI VIEW (MV), SEQUENCE VIEW (SEQ), AND SCENE GRAPH (SG).

Methods	**Capability Info**				**Setting**	**Sub-Task Validation**										**Final Task Validation**									
						Task 1		**Task 2**		**Task 3**		**Task 4**		**Task 5**		**Task 1**		**Task 2**		**Task 3**		**Task 4**		**Task 5**	
	SV	MV	Seq	SG	Harsh	D	R	D	R	D	R	D	R	D	R	D	R	D	R	D	R	D	R	D	R
Palm-E [15]	✓					85	85	**100**	85	**100**	**100**	**100**	**100**	**100**	**100**	**100**	90	**100**	85	**100**	**100**	**100**	**100**	**100**	90
Palm-E [15]	✓				✓	-	-	-	-	**100**	**100**	**100**	**100**	**100**	**100**	-	-	-	-	**100**	**100**	**100**	95	**100**	50
SuccessVQA [20]	✓		✓			45	40	**100**	55	0	0	**100**	**100**	50	50	70	45	**100**	30	95	95	**100**	**100**	**100**	75
SuccessVQA [20]	✓		✓		✓	-	-	-	-	0	0	55	45	50	50	-	-	-	-	**100**	**100**	75	40	55	0
Reflect [10]	✓			✓		**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**95**	**100**	**100**	75	75	**100**	**100**	**100**	**100**
Reflect [10]	✓			✓	✓	-	-	-	-	**100**	**100**	**100**	**100**	**100**	**100**	-	-	-	-	55	55	55	40	50	40
MA [5]		✓	✓			40	35	80	80	30	30	35	35	90	60	75	70	85	50	70	70	65	55	**100**	45
AHA [16]		✓	✓			85	55	**100**	70	85	85	**100**	**100**	**100**	90	95	55	85	60	**100**	75	95	95	**100**	70
Ours†	✓			✓		**100**	85	95	95	95	95	**100**	**100**	**100**	65	**100**	80	95	95	95	95	**100**	**100**	**100**	60
Ours†	✓			✓	✓	-	-	-	-	90	0	95	0	**100**	0	-	-	-	-	0	0	40	0	0	0
Ours		✓		✓		**100**	95	**100**	**100**	**100**	**100**	**100**	**100**	**100**	75	**100**	**95**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	70

to all methods and adapt both the input format and the visual information to match each method's original design.

The compared methods differ in the type of **capability information** they utilize, including *Single-View (SV), Multi-View (MV), Sequential Frames (Seq), and Scene Graphs (SG)*. Our framework combines *MV* and *SG* to support both robust perception and structured reasoning. Among these, our method and Reflect use of scene graphs to represent spatial relationships for failure detection and reasoning. However, Reflect employs a broader and fundamentally different set of relation types, and its scene graph is constructed using heuristic extraction from depth images, which is not compatible with our experimental setup. Therefore, we use ChatGPT-4o for Reflect with human-annotated ground-truth relational information based on the provided viewpoints to ensure fair and consistent comparison. To ablate the effectiveness of our proposed framework, we define a variant, denoted as Ours†, which replaces the *MV* capability information with *SV*, while retaining all other components of our framework.

For Palm-E, SuccessVQA, Reflect, and Ours†, which operate under *SV* constraints, we use the *Harsh* **Setting**, where key objects are occluded from a fixed, single-camera view. This assesses each model's ability to reason with incomplete visual information, and was applied specifically to Tasks 3-5. In contrast, Tasks 1 and 2 were conducted under standard conditions with fully visible scenes, as small object sizes and scene compositions made the *Harsh* setting infeasible. For other methods that rely on multi-view inputs, the *Harsh* condition does not apply, as multiple viewpoints inherently reduce the likelihood of object occlusion.

For validation, we apply dual-level validation, **Sub-Task Validation** and **Final Task Validation** for all **Methods**. Although Reflect and our method perform this dual-level validation in their original papers, other methods typically assess **Sub-Task Validation** alone. Dual-level validation is more robust and comprehensive in evaluating a model's reasoning ability and practical effectiveness. Therefore, our experimental setup applies this approach to all methods to ensure a fair and thorough comparison. We evaluate each task based on its internal failure *detection (D)* and *reasoning (R)* capabilities. Failure *D* is evaluated based on whether the model can correctly identify the occurrence of a failure in binary terms (i.e., yes or no). In contrast, failure *R* is assessed by the model's ability to explain the cause of the failure using spatial relationships between objects. For consistency, we restrict the types of reasoning relations to those defined in Sec. III-A.

For evaluation, we manually defined 20 different scenarios that all contain failure, for each of the five tasks as shown in Fig. 6. These conditions were assessed with dual-level validation, for a total of 200 trials per **Methods**. Therefore, the

success rate of failure detection and reasoning is calculated as the number of scenarios where failure was correctly detected divided by the total number (20) of scenarios with failure. As shown in Table III, our method achieves strong failure-detection performance across all five tasks. However, the accuracy of fail reasoning is comparatively lower in certain cases, primarily due to limitations of the 2DSGG model. This issue is particularly evident in Task 1 and Task 5 during sub-task validation, as well as in Task 5 during final-task validation. In these cases, the Reflect method achieves higher reasoning accuracy. It is important to note that Reflect leverages human-annotated explanations of the task space, which provides a significant advantage during dual-level validation. Moreover, under the *Harsh* **Setting** in final-task validation, Reflect performance drops considerably, as it struggles to infer occluded objects from a single viewpoint. In the ablation study, our framework achieves significantly better performance on dual-validation than Ours†, thereby decisively confirming the effectiveness of the *MV*.

V. Conclusion

In this work, we propose a robust task planning framework that addresses key limitations of existing robotic task planners based on large language models (LLMs) or vision-language models (VLMs). By leveraging unified scene graphs to model object-level relationships and incorporating multi-view inputs, our method improves failure detection and reasoning in complex environments. Our strategy to compare predicted and observed scene structures further aids in identifying failure causes. Experiments on five benchmark tasks demonstrate the effectiveness of our method in both planning and replanning. Comparative results show superior performance over existing LLM- and VLM-based approaches, underscoring the value of combining structured spatial reasoning with large models. Despite these promising results, certain limitations remain. Specifically, the overall accuracy of failure detection is hindered by the limited performance of the current 2D scene graph generation (2DSGG) model, which struggles to capture precise and consistent spatial representations from 2D images. Furthermore, the method's applicability is currently constrained by the limited object types and relationships used in our experiments. Future work will therefore focus on significantly expanding the scope to include a more diverse range of object relationships and material properties. This expansion is essential for verifying the robustness and superior performance of our approach across more diverse and complex real-world scenarios.

References

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

[2] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *Conf. on Robot Learning*. PMLR, 2023, pp. 1769–1782.

[3] A. Goyal, A. Mousavian, C. Paxton, Y.-W. Chao, B. Okorn, J. Deng, and D. Fox, "Ifor: Iterative flow minimization for robotic object rearrangement," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 787–14 797.

[4] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conf. on robot learning*. PMLR, 2022, pp. 894–906.

[5] J. Duan, W. Yuan, W. Pumacay, Y. R. Wang, K. Ehsani, D. Fox, and R. Krishna, "Manipulate-anything: Automating real-world robots using vision-language models," in *Conf. on Robot Learning*, 2024.

[6] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," in *Conf. on Robot Learning*. PMLR, 2023, pp. 540–562.

[7] W. Zhang, M. Wang, G. Liu, X. Huixin, Y. Jiang, Y. Shen, G. Hou, Z. Zheng, H. Zhang, X. Li *et al.*, "Embodied-reasoner: Synergizing visual search, reasoning, and action for embodied interactive tasks," *arXiv preprint arXiv:2503.21696*, 2025.

[8] A. Mei, G.-N. Zhu, H. Zhang, and Z. Gan, "Replanvlm: Replanning robotic tasks with visual language models," *IEEE Robotics and Automation Letters*, 2024.

[9] M. Skreta, Z. Zhou, J. L. Yuan, K. Darvish, A. Aspuru-Guzik, and A. Garg, "Replan: Robotic replanning with perception and language models," *arXiv preprint arXiv:2401.04157*, 2024.

[10] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," *Conf. on Robot Learning*, 2023.

[11] C. Cornelio and M. Diab, "Recover: A neuro-symbolic framework for failure detection and recovery," in *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems*. IEEE, 2024, pp. 12 435–12 442.

[12] D. Honerkamp, M. Büchner, F. Despinoy, T. Welschehold, and A. Valada, "Language-grounded dynamic scene graphs for interactive object search with mobile manipulation," *IEEE Robotics and Automation Letters*, 2024.

[13] H. Jiang, B. Huang, R. Wu, Z. Li, S. Garg, H. Nayyeri, S. Wang, and Y. Li, "RoboEXP: Action-conditioned scene graph via interactive exploration for robotic manipulation," in *First Vision and Language for Autonomous Driving and Robotics Workshop*, 2024.

[14] Z. Ni, X. Deng, C. Tai, X. Zhu, Q. Xie, W. Huang, X. Wu, and L. Zeng, "Grid: Scene-graph-based instruction-driven robotic task planning," in *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems*. IEEE, 2024, pp. 13 765–13 772.

[15] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," in *Inter. Conf. on Machine Learning*. PMLR, 2023, pp. 8469–8488.

[16] J. Duan, W. Pumacay, N. Kumar, Y. R. Wang, S. Tian, W. Yuan, R. Krishna, D. Fox, A. Mandlekar, and Y. Guo, "Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation," *arXiv preprint arXiv:2410.00371*, 2024.

[17] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. D. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable task planning," *CoRR*, 2023.

[18] K. Tang, Y. Niu, J. Huang, J. Shi, and H. Zhang, "Unbiased scene graph generation from biased training," in *Proc. of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2020, pp. 3716–3725.

[19] M. Neau, P. E. Santos, K. Sammut, A.-G. Bosser, and C. Buche, "Real-time scene graph generation," *arXiv preprint arXiv:2405.16116*, 2024.

[20] Y. Du, K. Konyushkova, M. Denil, A. Raju, J. Landon, F. Hill, N. de Freitas, and S. Cabi, "Vision-language models as success detectors," in *Conf. on Lifelong Learning Agents*. PMLR, 2023, pp. 120–136.

[21] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: open and efficient foundation language models. arxiv," *arXiv preprint arXiv:2302.13971*, 2023.

[22] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma *et al.*, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *Inter. jour. of computer vision*, vol. 123, pp. 32–73, 2017.

[23] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric sort: Rethinking sort for robust multi-object tracking," in *Proc. of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2023, pp. 9686–9696.

[24] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The semantic web: 15th inter. conf., ESWC*. Springer, 2018, pp. 593–607.

[25] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[26] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.