2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

GRID: Scene-Graph-based Instruction-driven Robotic Task Planning

Zhe Ni†, Xiaoxin Deng†, Cong Tai†, Xinyue Zhu, Qinghongbing Xie, Weihang Huang, Xiang Wu, Long Zeng*
https://github.com/jackyzengl/GRID

***Abstract*— Recent works have shown that Large Language Models (LLMs) can facilitate the grounding of instructions for robotic task planning. Despite this progress, most existing works have primarily focused on utilizing raw images to aid LLMs in understanding environmental information. However, this approach not only limits the scope of observation but also typically necessitates extensive multimodal data collection and large-scale models. In this paper, we propose a novel approach called Graph-based Robotic Instruction Decomposer (GRID), which leverages scene graphs instead of images to perceive global scene information and iteratively plan subtasks for a given instruction. Our method encodes object attributes and relationships in graphs through an LLM and Graph Attention Networks, integrating instruction features to predict subtasks consisting of pre-defined robot actions and target objects in the scene graph. This strategy enables robots to acquire semantic knowledge widely observed in the environment from the scene graph. To train and evaluate GRID, we establish a dataset construction pipeline to generate synthetic datasets for graph-based robotic task planning. Experiments have shown that our method outperforms GPT-4 by over 25.4% in subtask accuracy and 43.6% in task accuracy. Moreover, our method achieves a real-time speed of 0.11s per inference. Experiments conducted on datasets of unseen scenes and scenes with varying numbers of objects demonstrate that the task accuracy of GRID declined by at most 3.8%, showcasing its robust cross-scene generalization ability. We validate our method in both physical simulation and the real world. More details can be found on the project page https://jackyzengl.github.io/GRID.github.io/.**

I. Introduction

Recent advancements in large language models (LLMs) have shown promising outcomes in robotic task planning with instructions [1]–[3]. LLM-based robotic task planning necessitates a cohesive comprehension of instructions and semantic knowledge of the environment. The primary challenge lies in adequately representing environmental information for LLM-based approaches to leverage, given that LLMs are designed for processing textual content.

In recent years, several studies have concentrated on integrating the visual modality to convey environmental knowledge. Some efforts combined the outcomes of vision models and LLMs in parallel [4]–[6], while others reasonably aligned the two modalities using vision-language models (VLMs) [2], [3], [7]–[9]. However, employing images as input limits the comprehension of global scenario information, and the feature alignment between pixels and instructions is challenging.

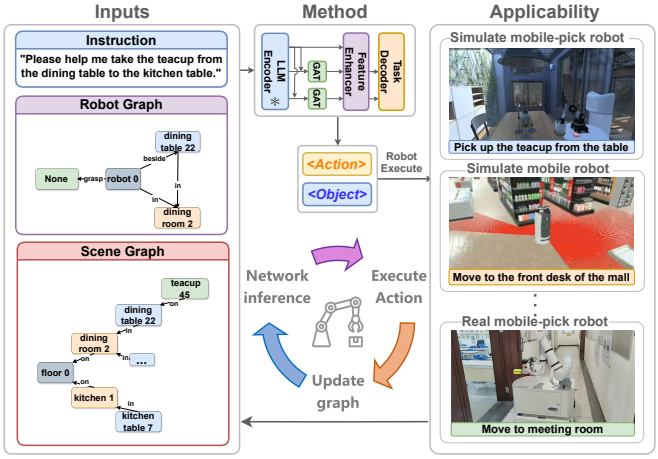


Fig. 1: Our network, GRID, leverages instructions, scene graphs, and robot graphs as inputs for robotic task planning. Both environmental knowledge and the robot's state are densely represented through graphs. The robot iteratively updates the graphs and executes the subtasks planned by GRID until completing the entire task. GRID can be deployed to robots in different forms, operating effectively in various environments.

To overcome the limitations of images, a widely observed and rich-semantic representation of a scene, called the scene graph, has been introduced to robotics research [10], [11]. Recently, faster and more precise methods for scene graph generation [12]–[15] facilitated the widespread utilization of graphs for long-term task planning [16]–[18]. The applications of scene graphs [19], [20] have demonstrated that the introduction of graphs effectively addresses issues such as catastrophic forgetting, hallucination predictions, language ambiguity, and the lack of real-world environmental experience in LLM models. However, graph-based task planning with natural language instructions has not garnered sufficient attention.

Unlike previous works, we utilize scene graphs to represent the environment for robotic task planning. In scene graphs, objects and their relationships within a scene are structured into graph nodes and edges. To provide a precise portrayal of the robot's status, we separate the robot from the scene graph to create a robot graph, which contains the robot's location, nearby objects, and grasped objects. We introduce a lightweight transformer-based model **GRID**, designed for deployment on offline embodied agents. As illus-

†Equal contribution.

*Corresponding author. (e-mail: zenglong@sz.tsinghua.edu.cn)
Zhe Ni, XiaoXin Deng, Cong Tai, Xinyue Zhu, Qinghongbing Xie, Weihang Huang, and Long Zeng are with Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, China.
Xiang Wu is with Shenzhen Pudu Technology Inc., Shenzhen, China.
This research was funded by Shenzhen Major Science and Technology Project (KJZD20230923115503007, KJZD20230923114900002)

trated in Fig. 1, the model takes instruction, scene graph, and robot graph as inputs, subsequently determining the subtask for the robot to execute. In GRID, the instruction and both graphs are mapped into a unified latent space by a shared-weight LLM encoder named $INSTRUCTOR$ [21]. Then the encoded graph nodes and their relationships are refined by graph attention network (GAT) modules. Integrating the outputs from GAT and the encoded instruction with a cross-attention-based feature enhancer, the resultant enhanced features are fed into a transformer-based task decoder to yield the robot subtask. Compared with the holistic sentence form, our subtask expressed as `<action>-<object>` pair form requires less enumeration to cover diverse object categories. GRID iteratively plans the subtask, enabling it to respond to real-time scene changes and human interference, thereby correcting unfinished tasks.

Since existing scene graph datasets are not designed for robotic task planning [22], we develop a synthetic dataset construction pipeline to generate scene graph datasets for instruction-driven robotic task planning. We use subtask accuracy and task accuracy as evaluation metrics. Subtask accuracy is the proportion of the subtasks that simultaneously predict the correct action and object. Task accuracy refers to the proportion of instruction tasks for which all associated subtasks are predicted correctly and sequentially. Evaluations on our datasets have shown that our method outperforms GPT-4 by over 25.4% in subtask accuracy and 43.6% in task accuracy. The small parameter size of GRID enables real-time offline inference with a time consumption of 0.11s per inference. To test the model's generalization ability, we constructed five datasets with scenes of different sizes and an unseen scenes dataset (i.e. all the scenes were unseen during the training phase). The maximum decrease in task accuracy on these datasets was only 3.8% without additional training. Tests in simulated environments and the real world demonstrate that our approach enables cross-room task planning, which is challenging for visual-based approaches.

Summary of Contributions:

- We first introduce scene graphs to facilitate instruction-driven robotic task planning by leveraging the graphs' ability to comprehend wide-perspective and rich semantic knowledge of the environment.
- We propose a novel GAT-based network called GRID, which takes instructions, robot graphs, and scene graphs as inputs. GRID outperforms GPT-4 by over 43.6% in task accuracy while maintaining a low inference time cost of only 0.11s.
- We construct a synthetic dataset generation pipeline to produce datasets for scene-graph-based instruction-driven robotic task planning.

II. Related Work

A. Scene Graph in Robotic Research

Scene graphs are rich in semantic and dense representations of scenes [23]. Previous works have explored robotic applications of scene graphs in navigation and task planning. Sepulveda *et al.* [24] and Ravichandran *et al.* [25] utilized scene graphs to enable platform-agnostic mobile robot navigation. Jiao *et al.* [26] demonstrated that scene graphs can assist robots in understanding the semantics of a scene to perform task planning. Amiri *et al.* [17] and Han *et al.* [18] showed that robots can extract more contextual information from scene graphs than from a single image, benefiting from the wide observable domains of scene graphs, which leads to improved long-term task planning. SayPlan [27] is closely related to our work. They utilized an online LLM to process the collapsed scene graph and incorporated simulated feedback. However, the text-based output lacks object IDs, making it unable to handle multiple objects of the same type, and the LLM's hallucination may impact executability. In the scene graph research field, instruction-driven robotic task planning has not received sufficient attention. To our knowledge, we are the first to use graph networks to process scene graphs and integrate LLMs to address instruction-driven robotic task planning.

B. Understand Scene Information With LLMs

Recent advancements in LLMs have promoted robotic systems' comprehension of instructions, where perceiving scene information is essential for solving a wide range of grounded real-world robotics problems [28]. Previous studies have explored various approaches to incorporate the visual modality, aiming to obtain richer scene information, with the primary challenge lying in fusing vision and text modalities. Ahn *et al.* [4] and Huang *et al.* [6] multiplied the outputs of visual models and LLMs to integrate the two. Dorbala *et al.* [29] converted results from distinct visual modules (e.g., object detection, image captioning) into text modality prompts. Inspired by CLIP [30], several studies have aligned both modalities into the same space [1], [5], [8], [31], [32]. Recent research indicates that incorporating features from one modality into another within the model yields better performance. Brohan *et al.* [33] inserted instruction embeddings into a visual model, whereas Jin *et al.* [7] and Driess *et al.* [2] projected visual features onto a large-scale LLM to leverage its inference capabilities. However, these models extract sparse and low-level semantic knowledge from raw images, rely heavily on large-scale data for training, and face challenges in analyzing information beyond what is depicted in the images.

In contrast, we utilize scene graphs instead of images in instruction-driven robotic task planning. Our approach enables the direct utilization of LLM to process rich semantic knowledge within scene graphs, eliminating the need for additional vision-text alignment or massive multi-modal data. Additionally, scene graphs offer a broader observable domain to embodied agents compared to images, enabling robots to navigate to various locations and perform tasks beyond the immediate field of view.

III. Problem Statement

We frame the graph-based instruction-driven task planning problem as follows: given an instruction, a robot graph, and

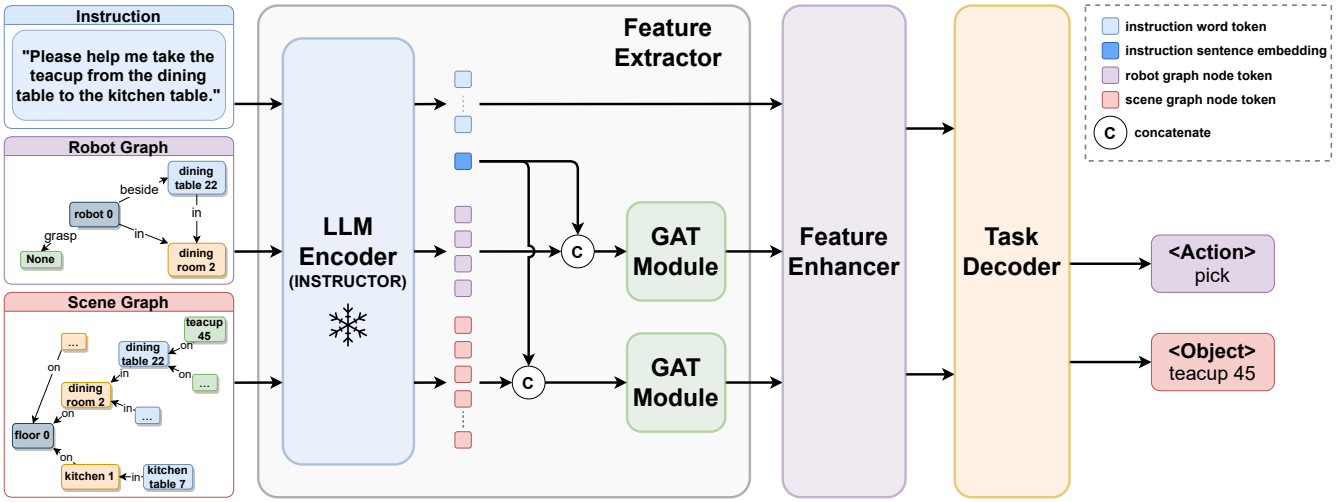


Fig. 2: The architecture diagram of GRID. The instruction, robot graph, and scene graph are all transformed into tokens through $INSTRUCTOR$ [21]. Subsequently, GAT modules extract structural information from the graphs. The resulting tokens undergo reinforcement by a feature enhancer and are then fed into a task decoder, ultimately generating outputs for action and object in ID form.

a scene graph, our objective is to predict the subtask required for the robot to execute in the current stage. Typically, the instruction is a human-language task that can be decomposed into a series of subtasks.

The scene graph $\mathbf{s}$ and robot graph $\mathbf{r}$ comprise multiple nodes $\mathbf{n}$ and edges $\mathbf{e}$. Nodes denote the objects within the environment characterized by attributes such as colors, positions, and other properties. Edges depict the relationships between objects. They are formulated as:

$$\begin{aligned} \mathbf{s} &= \langle \mathbf{n}^S, \mathbf{e}^S \rangle, \\ \mathbf{r} &= \langle \mathbf{n}^R, \mathbf{e}^R \rangle, \end{aligned} \tag{1}$$

where $\mathbf{n}^S, \mathbf{e}^S$ represent the nodes and edges of the scene graph respectively, while $\mathbf{n}^R, \mathbf{e}^R$ denote the nodes and edges of the robot graph.

Let $\mathbb{I}$ denote the space of instructions, $\mathbb{A}$ be the action space of robot skills, and $\mathbb{O} = [0, M)$ represent the set of IDs of all nodes in the scene graph, where M is the number of nodes in the scene graph. The output subtask space $\mathbb{P} = \mathbb{A} \times \mathbb{O}$, where $\times$ denotes the Cartesian product.

At stage t, given $\mathbf{r}^t, \mathbf{s}^t$ and a human-language instruction $I \in \mathbb{I}$, let $\hat{\pi}^t \in \mathbb{P}$ represent the predicted subtask, $\hat{a}^t \in \mathbb{A}$ denote the predicted action, and $\hat{o}^t \in \mathbb{O}$ be the ID of the predicted node in $\mathbf{s}^t$. The objective function is formulated as follows:

$$I, \mathbf{r}^t, \mathbf{s}^t \xrightarrow{infer\ subtask} \hat{\pi}^t = \langle \hat{a}^t, \hat{o}^t \rangle \tag{2}$$

After the robot interacts with the environment by executing the subtask predicted at stage t, the graphs are updated to the next stage $\mathbf{r}^{t+1}, \mathbf{s}^{t+1}$. The robot graph can be updated by the robot sensor algorithm, while the scene graph can be constructed and updated using scene graph generation methods [10]–[15].

$$\begin{aligned} \mathbf{r}^t &\xrightarrow{update} \mathbf{r}^{t+1} \\ \mathbf{s}^t &\xrightarrow{update} \mathbf{s}^{t+1} \end{aligned} \tag{3}$$

The prediction iterates until all subtasks associated with the given instruction I have been executed.

IV. Method

An overview of our method is depicted in Fig. 2. Our model **GRID** predicts a pair of action and object ID $\langle \hat{a}^t, \hat{o}^t \rangle$ for a given instruction, robot graph, and scene graph triad $\langle I, \mathbf{r}^t, \mathbf{s}^t \rangle$ at stage t:

$$\langle \hat{a}^t, \hat{o}^t \rangle = \mathbf{GRID}(\langle I, \mathbf{r}^t, \mathbf{s}^t \rangle) \tag{4}$$

For clarity, the stage label t will be omitted in the formulas in the subsequent discussions.

GRID employs an encoder-decoder architecture. The feature extractor comprises a shared-weight LLM and two GAT modules. The instruction, robot graph, and scene graph all undergo processing by the LLM, serving as a text backbone for extracting raw semantic features. Two GAT modules are utilized to extract structural features from the respective graphs (Sec. IV-A). The feature enhancer collectively reinforces the overlapping information of instructions and graphs (Sec. IV-B). The task decoder comprehensively analyzes the features and generates the predicted results (Sec. IV-C).

A. Feature Extractor

We employ a frozen pre-trained LLM named $INSTRUCTOR$ as a shared-weight text backbone. Given a sentence of length m, it generates task-aware embeddings [21]. These embeddings consist of a sequence of word tokens $\mathbf{w} = (w_1, ..., w_m)$ and a sentence embedding y, which represents the features of the entire sentence. Employing the same text backbone enables the features of instructions and graphs to be mapped into the same latent space without requiring additional alignment. For instructions, we compute both word tokens $\mathbf{w}^I$ and sentence embedding y^I:

$$\langle \mathbf{w}^I, y^I \rangle = INSTRUCTOR(I) \tag{5}$$

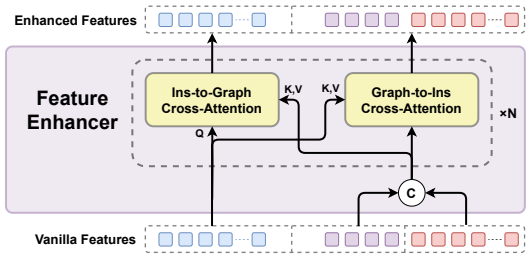


Fig. 3: Vanilla instruction and graph features are fed into N layers of parallel cross-attention to enhance crossover information between instructions and graphs.

For both robot graphs and scene graphs, each node's attributes are treated as distinct sentences. Only the sentence embedding for each node is considered, which is interpreted as a node token:

$$\begin{aligned} \mathbf{nt}^R &= INSTRUCTOR(\mathbf{n}^R), \\ \mathbf{nt}^S &= INSTRUCTOR(\mathbf{n}^S), \end{aligned} \tag{6}$$

where $\mathbf{nt}^R$ denotes the sequence of node tokens from nodes in the robot graph, and $\mathbf{nt}^S$ denotes the sequence of node tokens from nodes in the scene graph.

To incorporate instruction information when extracting graph features, we concatenate the sentence embedding of the instruction with every node token. Then, the node token, along with the edge information in each graph, is inputted to their respective graph attention network (GAT) modules [34]:

$$\begin{aligned} \mathbf{y}^R &= GAT_{robot}(concat(\mathbf{nt}^R, y^I), \mathbf{e}^R), \\ \mathbf{y}^S &= GAT_{scene}(concat(\mathbf{nt}^S, y^I), \mathbf{e}^S), \end{aligned} \tag{7}$$

where GAT_{robot}, GAT_{scene} denote GAT modules for the robot and scene graph respectively.

B. Feature Enhancer

Inspired by Grounding DINO [35], we have designed a feature enhancer, as illustrated in Fig. 3. The feature enhancer utilizes a stack of parallel multi-head cross-attention layers $CA(Q;K;V)$.

CA_I, CA_g denote instruction-to-graph and graph-to-instruction multi-head cross-attention layers with residual connections respectively. At layer l, CA_I takes the word tokens from instructions $\mathbf{w}^I_{l-1}$ as queries, and the node tokens from graphs $\langle \mathbf{y}^R_{l-1}, \mathbf{y}^S_{l-1} \rangle$ as keys and values. Conversely, in CA_g, $\mathbf{w}^I_{l-1}$ serve as keys and values, while $\langle \mathbf{y}^R_{l-1}, \mathbf{y}^S_{l-1} \rangle$ functions as queries:

$$\begin{aligned} \mathbf{w}^I_l &= CA_I(\mathbf{w}^I_{l-1}; \langle \mathbf{y}^R_{l-1}, \mathbf{y}^S_{l-1} \rangle; \langle \mathbf{y}^R_{l-1}, \mathbf{y}^S_{l-1} \rangle) \\ \langle \mathbf{y}^R_l, \mathbf{y}^S_l \rangle &= CA_g(\langle \mathbf{y}^R_{l-1}, \mathbf{y}^S_{l-1} \rangle; \mathbf{w}^I_{l-1}; \mathbf{w}^I_{l-1}) \end{aligned} \tag{8}$$

These structures emphasize the crossover information between instructions and graphs, particularly when the same object appears in both. After traversing the final layer of the feature enhancer, the tokens from the instruction act as fusion features $\mathbf{f}^I = \mathbf{w}^I_{last}$, and the tokens from graphs serve as graph queries $\langle \mathbf{q}^R, \mathbf{q}^S \rangle = \langle \mathbf{y}^R_{last}, \mathbf{y}^S_{last} \rangle$. Subsequently, they are inputted into the task decoder.

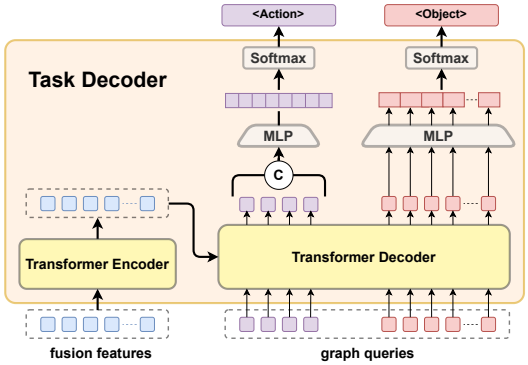


Fig. 4: The enhanced tokens are segregated into fusion features and graph queries, which are input into the transformer encoder and decoder, respectively. The tokens from the robot graph are mapped to scores for each action, and each token from the scene graph is converted to the score for the corresponding node.

C. Task Decoder

We develop a task decoder to integrate instruction and graph features for predicting the action and object of the subtask, as depicted in Fig. 4.

Different from treating the subtask as an integral sentence in previous works, we divide it into `<action>` and `<object>` as two parts. This division enables diverse combinations of robot "atomic" skills with objects in the scene, without being limited by the types of objects like SayCan [4] (which covers 551 subtasks spanning only 17 objects) or requiring enumeration of a large number of subtasks like in LAVA [32] (87K subtasks). We use IDs from the scene graph to uniquely identify objects instead of relying on textual descriptions, thus avoiding ambiguities for downstream modules [2], [29], [36].

Fusion features are fed into a transformer encoder $Enc(src)$. Graph queries are fed into a transformer decoder $Dec(tgt, src)$. The output embeddings enter the action branch and object branch respectively:

$$\langle \mathbf{f}^R, \mathbf{f}^S \rangle = Dec(\langle \mathbf{q}^R, \mathbf{q}^S \rangle, Enc(\mathbf{f}^I)), \tag{9}$$

where $\mathbf{f}^R$ denotes the output embeddings of the transformer decoder from queries $\mathbf{q}^R$, and $\mathbf{f}^S$ from queries $\mathbf{q}^S$.

Given the close relationship between the current state of the robot and the subsequent action, the output embeddings from the robot graph $\mathbf{f}^S$ are fed into an MLP (Multi-Layer Perceptron) after concatenation to yield N_{act} scores. The action associated with the highest score will be selected as the action to be executed:

$$\hat{a} = softmax(MLP_{act}(concat(\mathbf{f}^R))) \tag{10}$$

The output embeddings $\mathbf{f}^S$ from each node in the scene graph undergo individual processing through an MLP to derive node-specific scores. These scores indicate the probabilities of each object becoming the operated object:

$$\hat{o} = softmax(MLP_{obj}(\mathbf{f}^S)) \tag{11}$$

Subtask Type	Count
<move> - <object>	18347
<pick> - <object>	11811
<place to> - <object>	17924
<revolute open> - <object>	3456
<revolute close> - <object>	3456
<longitudinal open> - <object>	3503
<longitudinal close> - <object>	3503
<finish> - <object>	19981
Total	**81981**

TABLE I: The list of subtask types in the 70-object dataset, along with their respective counts. Subtasks are classified into different types based on their actions, as each is associated with a unique action.

D. Loss Function

We utilize the standard categorical cross-entropy loss for both the action branch $\mathcal{L}_{act}$ and the object branch $\mathcal{L}_{obj}$. The total loss function employed to train the complete model is defined as a weighted combination of these two losses with L_1 and L_2 regularization terms:

$$\mathcal{L} = \alpha\mathcal{L}_{act} + \beta\mathcal{L}_{obj} + \gamma\sum_i |w_i| + \frac{\delta}{2}\sum_i w_i^2, \qquad (12)$$

where $\mathcal{L}_{act} = \mathbf{CE}(\hat{a}^t, a^t_{gt}), \mathcal{L}_{obj} = \mathbf{CE}(\hat{o}^t, o^t_{gt})$, with the ground-truth action id a^t_{gt} and object id o^t_{gt}. α, β denote loss weights, γ, δ denote L_1 and L_2 regularization strengths respectively, and w_i denotes trainable weights in the network.

V. Dataset

Graph-based robotic task planning takes an instruction, robot graph, and scene graph as inputs and plans the subtask in the current state as output. Similar datasets (ALFRED [37], CALVIN [22], ARNOLD [38], THUD [39] etc.) provide instructions and corresponding subtasks. However, these datasets lack scene graphs and robot graphs corresponding to the subtasks or metrics indicating the completion of the entire task. Thus, we developed a synthetic dataset construction pipeline to generate datasets consisting of four components: instructions, subtasks, robot graphs, and scene graphs. The subtasks in the dataset serve as the ground truth and are used for network training and evaluation. We pioneered the construction of datasets that establish the interconnections between instructions, robot graphs, scene graphs, and subtasks providing fundamental benchmarks for task planning in this domain.

To generate one set of data, we use a randomly generated domestic environment to initialize a scene graph, $\mathbf{s}_0$, as a blueprint. Meanwhile, the robot graph, $\mathbf{r}_0$, is initialized with a predefined robot state. We then generate a list of feasible subtasks in this scene, e.g., pick-teacup. For each set of subtasks, corresponding human-language instructions are synthesized with the assistance of ChatGPT-4 in diverse expressions. These subtasks are used to iteratively update the blueprint $\mathbf{r}_0, \mathbf{s}_0$, obtaining the robot graphs and scene graphs for different stages.

We generated five datasets containing different object counts in each scene ranging from 30 to 70. The dataset with 70 objects in each scene (70-object dataset) includes about 20K instruction tasks. These tasks can be decomposed into 82K subtasks, as shown in Tab. I. Additionally, we constructed an unseen scene dataset in which all scenes are not seen during the training phase.

VI. Experiments

In this section, we first describe the training details. Then we compare our model with GPT-4 and validate the effectiveness of two key modules through ablation studies. We also explore the generalization ability of the model with different datasets. Finally, we demonstrate our approach both in the simulation environment and the real world.

A. Training Details

GRID is implemented on the PyTorch toolbox. It is optimized using AdamW, with a peak learning rate of 1e-4 that decays according to a one-cycle learning rate schedule with a diversity factor of 10 and a final diversity factor of 1e-4. We utilize 80% of the data from the 70-object dataset, which is the largest dataset in terms of scale, for training, while the remaining 20%, along with all other datasets, is used for evaluation. Our models are trained for 500 iterations with a total batch size of 240 on two NVIDIA RTX4090 GPUs. The loss weights are set to $\alpha = 5$ and $\beta = 25$, while the L_1 and L_2 regularization strengths are $\gamma = 0.2$ and $\delta = 0.8$, respectively.

B. Evaluation

Metrics. To evaluate the performance of our method, we proposed four main accuracy metrics: action accuracy (Act. Acc.), object accuracy (Obj. Acc.), subtask accuracy (Sub. Acc.), and task accuracy (Task Acc.). Considering each subtask sample as an independent prediction, action accuracy, and object accuracy measure the model's precision in predicting actions and objects, respectively. Subtask accuracy records the probability of correctly predicting both action and object, while task accuracy represents the proportion of all subtasks being predicted sequentially and correctly.

Comparison with baseline. Following Chalvatzaki *et al.* [36] and referring to SayPlan [27], we evaluate the task planning performance of LLaMA2 and GPT-4 on our datasets. The baseline methods settings are similar to LLM-As-Planner in SayPlan [27], where we utilize LLMs to generate the sequence of plans. Given that GPT-4 is one of the best-performing LLMs, its performance will surpass that of smaller-weight models like LLaMA2, Falcon, Vicuna, etc., making it a representative baseline. The prompt settings of LLM-As-Planner are shown in Tab. II. We use the chain of thought (CoT) and examples to enhance performance because detailed prompting can elicit better results in LLMs. To focus on the task planning ability without being influenced by the accuracy of scene graph generation, the input graphs are set to be complete and accurate. The results, as shown in Tab. III, demonstrate that in our 70-object dataset, our approach achieved 83.0% subtask accuracy and 64.1% task accuracy significantly outperforming GPT-4's 47.5% and 8.0% respectively. This may be because our model leverages

#Role: You are an instruction planning model for a mobile grasping robot, you aim to break down high-level instruction into subtasks for the robot to execute.
#Output Restriction: Your output is in a form of <action><operating object name><operating object id>, where <action>is one of the actions in ['move', 'pick', ...], The meaning of the actions: move: controls the robot's movement. pick: ... The <operating object name><operating object id>are the object and ids given in the prompt #Scene ...
#Scene: The current scene has objects with ids: <scene graph nodes description>. The relationships between objects in the scenes are: <scene graph edges description>
#Robot: ... #Instruction: <instruction>
#Task: Please tell me what the current subtask the robot should perform according to the scene and robot status. Write down your thought process.
#Example1:
input:
#Scene: The current scene has objects with IDs: floor 0, dining room 1, black dining table 2, purple display shelves 3, ... The relationships between objects in the scenes are: dining room 1 is on floor 0, ...
#Robot: The objects and object IDs around the robot: robot 0, brown pen 2. The relationships between objects around the robot are: pen 2 is grasped by robot 0.
#Instruction: Your goal is to get the object to the purple display shelves.
#Task: Please tell me ...
output:
#Think: Because I do not know where I am, so I should first move to purple display shelves... So output: move purple display shelves 3.
#Example2: ...

TABLE II: A summarized version of the input prompt settings example of LLM-As-Planner. For the complete and detailed prompt settings example, please refer to the project page https://jackyzengl.github.io/GRID.github.io/.

Model	#Params	30-object	40-object	50-object	60-object	70-object	Avg.
		S. A./T. A.	S. A./T. A.	S. A./T. A.	S. A./T. A.	S. A./T. A.	Infer Time
LLM-As-Planner(LLaMA2)	7B	21.0%/1.5%	24.8%/0.0%	20.5%/0.0%	22.5%/0.0%	22.0%/0.0%	3.10s(Offline)
LLM-As-Planner(LLaMA2)	70B	31.0%/3.0%	31.0%/4.3%	27.5%/4.3%	28.5%/2.2%	27.5%/0.0%	4.87s (Online)
LLM-As-Planner(GPT-4)	>1T	55.3%/16.7%	57.7%/14.3%	54.5%/11.1%	49.6%/10.0%	47.5%/8.0%	2.52s (Online)
GRID	**2.3M+1.5B**	**81.6%/60.3%**	**83.1%/63.2%**	**82.8%/63.0%**	**82.2%/62.1%**	**83.0%/64.1%**	**0.11s (Offline)**

TABLE III: Comparison between GRID and LLM-As-Planner on datasets with different numbers of objects in each scene. The offline inferences are conducted on one NVIDIA RTX4090 GPU. **S. A.** denotes subtask accuracy, **T. A.** denotes task accuracy. **Bold** values indicate the optimal data.

Model	#Params	Act. Acc.	Obj. Acc.	Sub. Acc.	Task Acc.
GRID	**2.3M+1.5B**	**84.8**	**94.1**	**83.0**	**64.1**
GRID w/o Feature Enhancer	2.2M+1.5B	83.2	93.1	81.0	56.2
GRID w/o GAT modules	314K+1.5B	83.1	93.3	80.2	55.5
GRID w/o GAT modules & Feature Enhancer	245K+1.5B	76.2	92.0	73.0	28.4

TABLE IV: Ablation studies of key modules. The number of parameters is recorded as the sum of trainable parameters and the parameters in the frozen LLM ($INSTRUCTOR$ [21]).

structural information in the graphs using GAT. Our model fuses and enhances the features of instructions and graphs repeatedly, allowing the model to focus on the objects mentioned in the instructions. Meanwhile, predicting actions and objects separately can effectively reduce the difficulty of predicting the entire subtask. Although GRID outperforms GPT-4 due to the benefits from trainable weights, it has significantly fewer parameters (2.3M+1.5B) and a shorter average inference time (0.11s) compared to GPT-4 (>1T, 2.52s) and LLaMA2 (7B, 3.10s; 70B4.87s). This enables its offline deployment and real-time inference on edge devices such as robots.

Ablation studies. We perform ablation studies on the 70-object dataset to analyze the importance of the GAT modules (together with concatenation) and the feature enhancer. The results shown in Tab. IV indicate that both the GAT modules and feature enhancer improve accuracy. The accuracy of the complete model is significantly improved compared to the simplified model. This may be because the full model both considers the relationships between objects in the graphs and sufficiently fuses the information from the instruction and graphs.

Generalization to scenes of different sizes. The results of evaluation on five datasets with different numbers of objects in each scene in Tab. III show that subtask and task accuracy of GRID are only slightly affected by the number of objects in each scene. The subtask and task accuracy of GPT-4 decrease by 8.7% as the number of objects in the scene increases, possibly because it is difficult for dialogue models to process long texts in larger scenes. In contrast, the task accuracy of GRID fluctuates within $62.2\% \pm 2\%$ indicating that our network can generalize well to scenes of different sizes (with different numbers of objects).

Generalization to unseen scenes. To evaluate the generalization ability of our model on unseen scenes, we separately constructed a dataset where all scenes had not appeared during model training. The results in Tab. V show that, compared to the 70-object dataset, the subtask accuracy in the unseen scenes dataset only decreases by about 2%, and the task accuracy decreases by about 3%. This demonstrates that our model has relatively good generalization performance for different scenes without any additional training.

Dataset	Act. Acc.	Obj. Acc.	Sub. Acc.	Task Acc.
70-object	84.8	94.1	83.0	64.1
Unseen Scenes	84.1	93.7	81.9	61.3

TABLE V: Generalization to unseen scenes.

C. Simulation Experiment

System Setup. To validate the functionality of the GRID network, we developed a robot simulation platform using Unity to conduct simulation experiments. The simulation system design is illustrated in Fig. 5. GRID and the robot control algorithm are deployed in ROS2 and interact with the physics simulation environment in Unity through the ROS-TCP Connector communication framework.

Simulation Demonstration. We deployed GRID on different types of robots (turtlebot, mobile robot, mobile-pick robot, etc.) and tested it in various scenarios (office, shop, home, etc.), as shown in Fig. 6. Fig. 7 illustrates an exemplary scenario where a mobile-pick robot successfully executes a complete task within the simulation environment. The provided language instruction for this example is as follows: "Please help me take the teacup from the dining table to the kitchen table." Subfigures (a) to (f) demonstrate the step-by-step process of the robot's movement, starting from the living room, then proceeding to the dining room to retrieve the teacup, and finally advancing to the kitchen for its placement. This demonstration indicates that our approach can be deployed on a mobile-pick robot and perform robotic task planning based on a given human-language instruction.

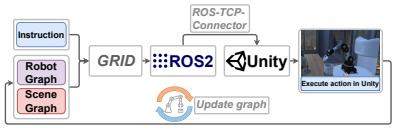
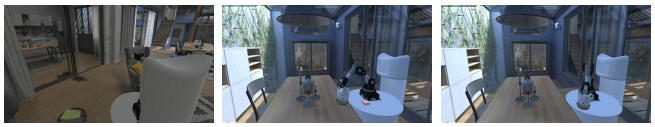


Fig. 5: Experimental system design in simulation.

Fig. 6: Simulation demonstration.



(a) Go to dining room (b) Picking up teacup (c) Picked up teacup

(d) Go to kitchen (e) Placing teacup (f) Done all subtask

Fig. 7: Mobile-pick simulation process.

D. Real Deployment Experiment

System Setup. To validate the efficacy of our work in real robot motion planning and explore its potential for handling a wider range of tasks in the future, we devised a process for deploying the GRID network onto physical robots for experimental purposes. The scene knowledge is constructed into a scene graph using scene graph generation methods [10], wherein the pose of objects can be obtained by a 6DoF pose estimation model [16]. The system structure is similar to the simulation experiment, as depicted in Fig. 8.

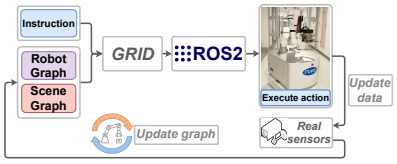


Fig. 8: Experimental system design in real world.

Fig. 9: Real world demonstration.

(a) Picking up box (b) Moving (c) Placing box

Fig. 10: The delivery task process.

Real Robot Demonstration. In real-world scenarios, GRID can be deployed to robots in various forms such as turtlebots, mobile robots, and mobile-pick robots, among others, as illustrated in Fig. 9. Fig. 10 demonstrates a mobile-pick robot performing a delivery task in a campus scene. The specific language instruction: "Please bring the yellow box on the table in the laboratory to the table in the meeting room."

The above demonstration suggests that our network, GRID, empowers robots of diverse configurations to plan instruction-driven tasks and execute subtasks in real-world scenarios. We hope that our approach will inspire further exploration into utilizing scene graphs for instruction-driven robotic task planning.

VII. Conclusions

In this paper, we present a novel approach utilizing scene graphs, rather than images, as inputs to enhance the comprehension of the environment in robotic task planning. At each stage, our network, GRID, receives instructions, robot graphs, and scene graphs as inputs, producing subtasks in `<action>-<object>` pair format. GRID incorporates a frozen LLM for semantic encoding and employs GAT modules to consider relationships between objects. The results indicate superior performance of our method over GPT-4 in scene-graph-based instruction-driven robotic task planning, showcasing adaptability to diverse scenes.

REFERENCES

[1] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 894–906.

[2] D. Driess, F. Xia, M. S. Sajjadi *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[3] B. Zitkovich, T. Yu, S. Xu *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[4] A. Brohan, Y. Chebotar, C. Finn *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on Robot Learning*. PMLR, 2023, pp. 287–318.

[5] S. Nair, A. Rajeswaran, V. Kumar *et al.*, "R3m: A universal visual representation for robot manipulation," *arXiv preprint arXiv:2203.12601*, 2022.

[6] W. Huang, C. Wang, R. Zhang *et al.*, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[7] C. Jin, W. Tan, J. Yang *et al.*, "Alphablock: Embodied finetuning for vision-language reasoning in robot manipulation," *arXiv preprint arXiv:2305.18898*, 2023.

[8] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding language with visual affordances over unstructured data," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 576–11 582.

[9] Y. Mu, Q. Zhang, M. Hu *et al.*, "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[10] I. Armeni, Z.-Y. He, J. Gwak *et al.*, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5664–5673.

[11] A. Rosinol, A. Gupta, M. Abate *et al.*, "3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans," *arXiv preprint arXiv:2002.06289*, 2020.

[12] J. Wald, H. Dhamo, N. Navab *et al.*, "Learning 3d semantic scene graphs from 3d indoor reconstructions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3961–3970.

[13] S.-C. Wu, J. Wald, K. Tateno *et al.*, "Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7515–7525.

[14] S.-C. Wu, K. Tateno, N. Navab *et al.*, "Incremental 3d semantic scene graph prediction from rgb sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5064–5074.

[15] S. Koch, N. Vaskevicius, M. Colosi *et al.*, "Open3dsg: Open-vocabulary 3d scene graphs from point clouds with queryable objects and open-set relationships," *arXiv preprint arXiv:2402.12259*, 2024.

[16] Y. Zhu, J. Tremblay, S. Birchfield *et al.*, "Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6541–6548.

[17] S. Amiri, K. Chandan, and S. Zhang, "Reasoning with scene graphs for robot planning under partial observability," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5560–5567, 2022.

[18] M. Han, Z. Zhang, Z. Jiao *et al.*, "Scene reconstruction with functional objects for robot autonomy," *International Journal of Computer Vision*, vol. 130, no. 12, pp. 2940–2961, 2022.

[19] F. Kenghagho Kenfack, F. Ahmed Siddiky, F. Balint-Benczedi *et al.*, "Robotvqa — a scene-graph- and deep-learning-based visual question answering system for robot manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 9667–9674.

[20] R. Liu, X. Wang, W. Wang *et al.*, "Bird's-eye-view scene graph for vision-language navigation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 10 968–10 980.

[21] H. Su, W. Shi, J. Kasai *et al.*, "One embedder, any task: Instruction-finetuned text embeddings," *arXiv preprint arXiv:2212.09741*, 2022.

[22] O. Mees, L. Hermann, E. Rosete-Beas *et al.*, "Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022.

[23] J. Bae, D. Shin, K. Ko *et al.*, "A Survey on 3D Scene Graphs: Definition, Generation and Application," in *Robot Intelligence Technology and Applications 7*, ser. Lecture Notes in Networks and Systems, J. Jo, H.-L. Choi, M. Helbig, H. Oh, J. Hwangbo, C.-H. Lee, and B. Stantic, Eds. Cham: Springer International Publishing, 2023, pp. 136–147.

[24] G. Sepulveda, J. C. Niebles, and A. Soto, "A deep learning based behavioral approach to indoor autonomous navigation," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4646–4653.

[25] Z. Ravichandran, L. Peng, N. Hughes *et al.*, "Hierarchical Representations and Explicit Memory: Learning Effective Navigation Policies on 3D Scene Graphs using Graph Neural Networks," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 9272–9279.

[26] Z. Jiao, Y. Niu, Z. Zhang *et al.*, "Sequential Manipulation Planning on Scene Graph," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2022, pp. 8203–8210, iSSN: 2153-0866.

[27] K. Rana, J. Haviland, S. Garg *et al.*, "Sayplan: Grounding large language models using 3d scene graphs for scalable task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[28] S. Tellex, N. Gopalan, H. Kress-Gazit *et al.*, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 25–55, 2020.

[29] V. S. Dorbala, J. F. Mullen Jr, and D. Manocha, "Can an embodied agent find your "cat-shaped mug"? llm-based zero-shot object navigation," *IEEE Robotics and Automation Letters*, 2023.

[30] A. Radford, J. W. Kim, C. Hallacy *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[31] E. Jang, A. Irpan, M. Khansari *et al.*, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[32] C. Lynch, A. Wahid, J. Tompson *et al.*, "Interactive language: Talking to robots in real time," *IEEE Robotics and Automation Letters*, 2023.

[33] A. Brohan, N. Brown, J. Carbajal *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[34] P. Velickovic, G. Cucurull, A. Casanova *et al.*, "Graph attention networks," *stat*, vol. 1050, no. 20, pp. 10–48 550, 2017.

[35] S. Liu, Z. Zeng, T. Ren *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[36] G. Chalvatzaki, A. Younes, D. Nandha *et al.*, "Learning to reason over scene graphs: a case study of finetuning gpt-2 into a robot language model for grounded task planning," *Frontiers in Robotics and AI*, vol. 10, 2023.

[37] M. Shridhar, J. Thomason, D. Gordon *et al.*, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749.

[38] R. Gong, J. Huang, Y. Zhao *et al.*, "Arnold: A benchmark for language-grounded task learning with continuous states in realistic 3d scenes," *arXiv preprint arXiv:2304.04321*, 2023.

[39] Y.-F. Tang, C. Tai, F.-X. Chen, W.-T. Zhang, T. Zhang, X.-P. Liu, Y.-J. Liu, and L. Zeng, "Mobile robot oriented large-scale indoor dataset for dynamic scene understanding," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 613–620.