GNN-Transformer Task Planning Enhanced with Semantic-Driven Data Augmentation

Soojin Jeong*, Seongwan Byeon*, Sangwoo Kim, HyeokJun Kwon, Yoonseon Oh

Department of Electronic Engineering, Hanyang University, Seoul, Korea
{jenicloun, bite3123, kimz1121, june2450, yoh21}@hanyang.ac.kr

Abstract

Natural language is the most intuitive means for humans to interact with robots, making task planning based on natural language commands a longstanding area of research. Large language models (LLMs) have significantly improved task planning by enhancing understanding of language and common sense. However, current methods still face several challenges: they lack a deep understanding of physical environments, their performance relies heavily on prompt examples, LLMs are oversized and not customized for specific tasks, and the planning costs remain high. To overcome these issues, we introduce the GNN-Transformer Task Planner (GTTP), designed to predict task-level actions by leveraging the semantic environment and incorporating historical state data. The GTTP architecture is scalable through the use of GNN layers, while transformer layers facilitate understanding task progression. In addition, our model uses a text encoder to embed environments, allowing it to be trained on simulated datasets and applied directly in real-world scenarios. We also propose an automated data generation method that includes semantic augmentation, planning verification, and instruction generation via LLM. This method enables the collection of 14k instruction-annotated tasks in the VirtualHome environment with minimal human effort. The model has been validated across diverse scenes containing up to 715 objects, achieving significantly higher success rates compared to baseline models. It has also been successfully deployed on a physical mobile manipulator, demonstrating its practical applicability and effectiveness.

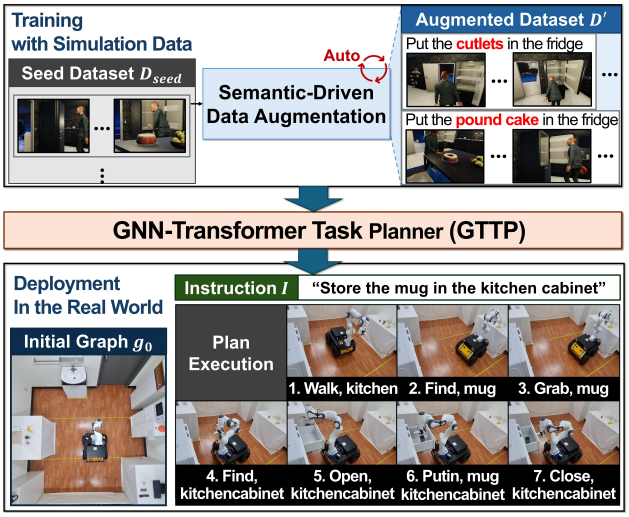

Figure 1: The whole process of our frameworks. The top of the figure shows the automatic data augmentation procedure in a simulation environment from a small seed dataset based on the semantic meaning of the task. We trained our model GTTP by these augmented datasets. The bottom of the figure is the inference procedure, where the goal instruction of the task and the initial scene graph serve as input. The model, trained only on simulation data, can be directly deployed in the real world.

Introduction

Natural language serves as a powerful instruction tool between humans and robots. Research on understanding natural language instructions for task planning has been continuously explored (Wächter et al. 2018; Nyga et al. 2018). Recently, with the advancement of large language models (LLMs), active research has been conducted leveraging the common sense understanding and natural language processing capabilities of LLMs for task planning (Ichter et al. 2022; Zeng et al. 2023; Huang et al. 2023). In (Huang et al. 2022; Singh et al. 2023), LLMs act as high-level planners via prompt engineering, incorporating admissible actions, the robot's state, and the environment into the prompts. However, this approach is constrained by the requirement for precisely engineered prompts to achieve desired outcomes and heavily relies on examples in the prompt. Furthermore, these planners struggle with ensuring feasibility and optimizing plans. To overcome these limitations, the Monte Carlo tree search method is employed, while leveraging LLMs to enhance search efficiency through common sense knowledge (Zhao, Lee, and Hsu 2023). The approach (Ichter et al. 2022) ensures feasibility and common sense understanding by combining value functions of primitive skills with LLMs.

On the other hand, some researchers have developed specialized models for task planning (Zhu et al. 2021; Ni et al. 2024; Gupta et al. 2024; Sun et al. 2022). The model in (Zhu et al. 2021) utilizes images as inputs, yet the model trained

*These authors contributed equally.

on simulators presents challenges due to the sim-to-real gap. Moreover, they require extensive training datasets, demanding considerable human effort. In response, a text-based planning model has been introduced to improve training efficiency with state graph representation. (Ni et al. 2024) present a method where the robot iteratively plans subgoals by leveraging text-based scene graphs of state with natural language instruction. (Gupta et al. 2024) have demonstrated the hybrid approach to neural-network-based learning and classical planning with PDDL.

Despite these advancements, current models still perform sub-optimally, partly due to the lack of comprehensive open datasets for task planning. Most datasets rely on costly and sometimes unreliable human annotation, making it challenging to scale (Traum et al. 2018; Chen et al. 2020). To overcome these limitations, recent studies have working on generating data automatically (Ahn et al. 2024; Wang et al. 2024; Zhang et al. 2024). (Ahn et al. 2024) propose large-scaling automatic data collection in real-world settings. (Zhang et al. 2024) automate task instruction annotation using LLM. However, research in this area is still in its early stages.

We introduce the **G**raph Neural Network (GNN)-**T**ransformer based **T**ask **P**lanner (**GTTP**), a learning model for high-level task planning with language instructions, as shown in Figure 1. Our model inputs a scene graph that incorporates semantic information about objects, encoded via a text embedding module. This information is processed by a GNN to accommodate varying object counts, and combined with a transformer model to consider historical states. The planner selects the most confident and feasible high-level plan for execution. Advantages of GTTP include *superior performance* compared to other models, *scalability* up to 715 objects, and excellent *sim-to-real transferability*.

Training the model requires a considerable amount of data, which is currently insufficient. To overcome this, we propose an LLM-based semantic data augmentation method that can automatically generate a substantial dataset of 14,666 data points from a seed data of only 45. This method not only automatically augments task plans with semantic object information and annotates them with language instructions but also evaluates the common sense validity of the data to ensure quality. This approach significantly reduces human effort, enhances training robustness, and improves the efficiency of developing large datasets.

Our main contributions are summarized as follows:

- GNN-Transformer based Task Planner: We introduce a cutting-edge planner that excels in generating high-level plans from language instructions. This planner demonstrates superior performance and excellent scalability.
- Semantic-Driven Data Augmentation: Our novel augmentation method significantly streamlines data preparation and ensures dataset quality.
- Real-World Validation: We validate our method using a mobile manipulator in a real-world setting, demonstrating its sim-to-real-transferability.

Problem Formulation

This paper proposes a method for high-level robotic task planning based on natural language commands, with a particular focus on household tasks in complex environments. For instance, when an embodied agent receives a goal instruction *"Put the chicken in the microwave,"* it must walk to the kitchen, find the chicken, grab it, open the microwave, and place the chicken inside. Our approach is designed to generate high-level and interpretable subgoals.

Formally, a goal instruction is denoted by I. The state of both the environment and the robot are represented by the directed scene graph $g_t = (\mathcal{V}_t, \mathcal{E}_t)$ at time t, where $\mathcal{V}_t$ and $\mathcal{E}_t$ correspond to node and edge features, respectively. Each node $v_t^i \in \mathcal{V}_t$ contains following five types of knowledge: (1) ***unique identifier (id)***, which assigns a distinct value to each object to distinguish it from others within the graph, (2) ***class name***, denoting the specific name of the object, such as `apple`, (3) ***category***, which refers to the broader classification groups such as `Appliances`, (4) ***properties***, which describe the attributes of an object, and (5) ***states***, indicating the current condition of the object. The state of an object can change over time. A directed edge $e_t^{ij} \in \mathcal{E}_t$ indicates a semantic ***relation*** between two nodes v_t^i and v_t^j. Further details of semantic information can be found in Appendix A.

The subgoal at time t is expressed as $\pi_t = (a_t, o_t)$, where $a_t \in \mathcal{A}$ and $o_t \in \mathcal{O}$ represent a robot action skill and its target object, respectively. Here, a set of action skill is defined as $\mathcal{A} = \{$`Walk`, `Find`, `Open`, `Close`, `Grab`, `PutIn`, `PutBack`, `SwitchOn`, `SwitchOff`, `Finish`$\}$. The object set $\mathcal{O}$ is the set of all objects in the given environment, which is assumed to be fully observable. Our objective is to design a task planning network architecture to predict an appropriate subgoal based on given instruction I and the historical sequence of states $g_{0:t} = \{g_0, \ldots, g_{t-1}, g_t\}$. The output of the planner at time t is the subgoal π_{t+1}.

GNN-Transformer Task Planning

To address the instruction-driven high-level task planning problem, we introduce a GNN-Transformer Task Planner where the input consists of a goal instruction I and state graph history $g_{0:t}$. The overview of our method is shown in Figure 2. The goal instruction I and the scene graph g_t are encoded by *instruction encoder* ϕ_I and *scene graph encoder* ϕ_g. Subsequently, the encoded goal instruction and scene graph are fed into the *subgoal planning network* $\mathcal{F}_\pi$, along with the history of previous state graphs, to estimate the probability distribution of potential subgoals based on the context of the ongoing task sequence. Consequently, the *subgoal selector* chooses the feasible subgoal π_{t+1} with the highest probability.

Encoding Scene Graphs and Instructions

Each node v_t^i and edge e_t^{ij} in the scene graph g_t contains semantic information about the environment in JSON format, as shown in Figure 2. The *scene graph encoder* ϕ_g leverages text embedding models to transform v_t^i and e_t^{ij} into 384-dimensional vectors, termed node feature $\hat{v}_t^i$ and edge

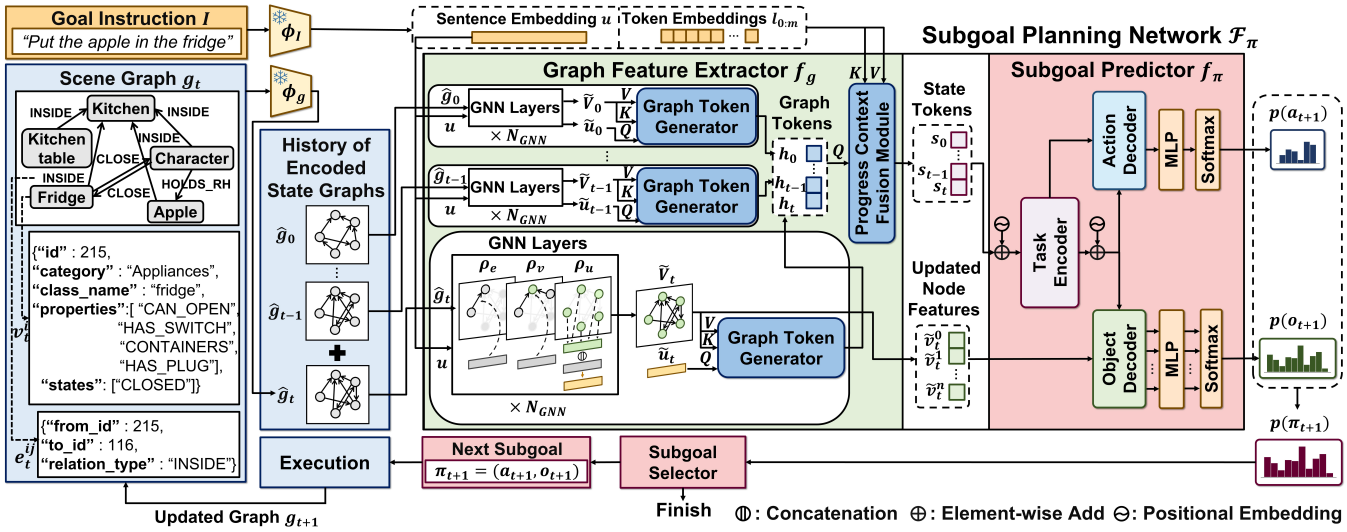

Figure 2: Overview of the planning process for GTTP. The procedure is divided into three primary stages: (1) The scene graph encoder ϕ_g and the instruction encoder ϕ_I extract features from the environment and the natural language instructions, respectively; (2) *Subgoal planning network* predicts the probability of action $p(a_{t+1})$ and object $p(o_{t+1})$ for next time step $t+1$, based on the encoded state graph history $\hat{g}_{0:t}$ and encoded instruction $\hat{I}$; (3) *Subgoal selector* determines the feasible subgoal π_{t+1} with the highest probability. The planning process iterates until it achieves action *finish*.

feature $\hat{e}_t^{ij}$, respectively. The embedded scene graph $\hat{g}_t$ preserves the original graph structure, maintaining the semantic relations between objects.

The instruction sentence I is input into the instruction encoder ϕ_I, which produces an instruction feature $\hat{I}$ consisting of a sentence embedding vector u and the set of token embeddings $l_{0:m}$, where m is the number of tokens. In this paper, Sentence-BERT (Reimers and Gurevych 2019) is adopted for both the state encoder ϕ_g and the goal encoder ϕ_I to ensure they share the same latent space.

Subgoal Planning Network Architecture

In this section, we propose the *subgoal planning network*, which processes the history of encoded state graphs $\hat{g}_{0:t}$ and encoded instruction feature $\hat{I}$ to plan a subgoal π_{t+1}. The network consists of a *graph feature extractor* f_g and a *subgoal predictor* f_π, as shown in Figure 2.

Graph Feature Extractor The graph feature extractor interprets both the encoded scene graph and the encoded instruction using *GNN Layers*, *graph token generator*, and *progress context fusion module*. To extract task-relevant features from given scene graphs $\hat{g}_{0:t}$ and goal instruction u, we used a stack of GNN layers. Following (Battaglia et al. 2018), we designed three update functions: ρ_e, ρ_v and ρ_u. The edge update function ρ_e can be formulated as: $\tilde{e}_t^{ij} = \rho_e(\hat{v}_t^i, \hat{v}_t^j, \hat{e}_t^{ij}, u) = g(\hat{v}_t^i \| \hat{v}_t^j \| \hat{e}_t^{ij} \| u)$, where $g(\cdot)$ is Multi-Layer Perceptron (MLP) function and $\|$ is a vector concatenation. Each updated edge $\tilde{e}_t^{ij}$ aggregates information from connected nodes and the provided instruction features u. Similarly, the node feature $\tilde{v}_t^i$ is updated by the function ρ_v: $\tilde{v}_t^i = \rho_v(\hat{v}_t^i, \mathcal{N}_t^i, u) = g\left(\hat{v}_t^i \| \frac{1}{|\mathcal{N}_t^i|} \sum_{k \in \mathcal{N}_t^i} g(\hat{v}_t^k \| \tilde{e}_t^{ki}) \| u\right)$, where $\mathcal{N}_t^i$ is a set of neighbor nodes of v_t^i. The updated node $\tilde{v}_t^i$ contains surrounding environment information. At last, u is inserted as a global feature of the graph and is updated by simply averaging all updated nodes: $\tilde{u}_t = \rho_u(\tilde{\mathcal{V}}_t, u) = g(u \| \frac{1}{|\tilde{\mathcal{V}}_t|} \sum_{i \in \tilde{\mathcal{V}}_t} \tilde{v}_t^i)$, where $\tilde{\mathcal{V}}_t$ is a set of updated node embeddings. In this paper, the number of GNN layers N_{GNN} is set as 3 and the dimensions of the updated node, edge, and instruction feature are reduced to 256, 128, 64, respectively.

To emphasize the features of goal-relevant objects, we added a cross-attention layer called *graph token generator* for each updated graph $\tilde{g}_t$. The module takes the updated sentence embedding vector $\tilde{u}_t$ as query Q and the updated node embeddings $\tilde{\mathcal{V}}_t$ as keys K and values V. It outputs a graph token h_t, which is repeated for all time steps, resulting in $h_{0:t} = [h_0, ..., h_t]$. Furthermore, the agent has to know which goal condition should be resolved for each state. For example, in the command "Go to the living room and turn the light off ", "the living room" should be the focus in the earlier state, while "the light" should be emphasized in the later state within the state history. For this reason, we added *progress context fusion module*, which is also cross-attention between graph tokens $h_{0:t}$ as queries and instruction token embeddings $l_{0:m}$ as keys and values. The resultant state tokens $s_{0:t}$ emphasize important instruction tokens for each state. Therefore, our planner takes the progress context to determine the proper subgoal.

Subgoal Predictor The *subgoal predictor* interprets historical state information to generate probability distributions for the next action and the next target object, denoted by $p(a_{t+1})$ and $p(o_{t+1})$, respectively. The overall structure is a modified version of vanilla Transformer architecture (Vaswani et al. 2017). Initially, the *Task Encoder* consumes

a sequence of state tokens along with positional embeddings to produce task tokens that encode historical information of ongoing tasks. These task tokens, after being element-wise added with positional embeddings, are fed into both the *action decoder* and *object decoder* to deliver sequential history features. The *action decoder* takes state tokens and outputs the action probabilities for each input state through the MLP and softmax layer. In inference, the planner only uses the probability $p(a_{t+1})$ from the current state token s_t. The *object decoder* takes the updated node embeddings $\tilde{\mathcal{V}}_t$ to find important target objects among them.

Subgoal Selector

The *subgoal selector* determines the next subgoal based on the following equation: $\pi_{t+1} = \arg\max_{\pi \in \Pi_f} p(\pi_{t+1})$, where Π_f is a set of successfully executable subgoals at current state. In simulations, the feasibility of each subgoal is verified using the VirtualHome simulator (Puig et al. 2018). For experimental implementations, we have developed a feasibility checker that accesses the feasibility according to predefined rules. Once π_{t+1} is determined, the state is updated to g_{t+1}. Consequently, the process is iteratively repeated in a single scenario until the predicted next action a_{t+1} reaches *finish* or cannot find a feasible action.

Node Dropout and Loss Function

During training, we introduce the node dropout technique for generalization across scenes with varying numbers of objects. Starting with the initial scene g_0 and the ground truth task plan, we categorize the objects in g_0, $\mathcal{O}$, into essential objects related to the plan, $\mathcal{O}_e$, and a set of background objects, $\mathcal{O}_b$. We randomly drop out 30%, 60%, and 90% of O_b, appending node-dropped graph sequences to the training set. It improves the robustness of GTTP against changes in background objects. For training, we employ *Cross-entropy* as the loss function in our model as follows: $\mathcal{L}_{total} = \alpha \mathcal{L}_{action} + \beta \mathcal{L}_{object}$, where α and β are coefficients for loss functions. $\mathcal{L}_{action}$ and $\mathcal{L}_{object}$ denote the loss functions associated with actions and objects, respectively.

Semantic-Driven Data Augmentation

We propose an automated data augmentation pipeline that considers the semantic information of objects to augment a seed dataset, which is a small set of hand-crafted task data. Additionally, we propose an LLM-based automatic plan verification and instruction generation method, which not only generates language instructions appropriate for the augmented task plans but also evaluates the reasonableness of these plans using common sense reasoning. The seed dataset is denoted by $D_{seed} = \{(\Pi_g^i, g_0^i) | i = 1, 2, \ldots, N\}$, where Π_g^i is the sequence of subgoals, g_0^i is the initial scene graph of ith data point. We collected seven seed tasks from (Singh et al. 2023) and six from our own task dataset. We got additional seed tasks by combining or dividing them. Consequently, we constructed the seed dataset with $N = 45$. Details are provided in Appendix A. Figure 3 demonstrates the process of automated semantic augmentation where the input task is *"Put the apple in the fridge,"* and the newly generated task is *"Store the pound cake back in the fridge."*

Semantic Augmentation

To scale up the seed dataset, we first choose a datapoint $\Pi_g = \{\pi_1, ..., \pi_T\}$, where T denotes the length of the plan and $\pi_t = (a_t, o_t)$. Let a set of all objects within Π_g be $O_{seed} = \{o^t | \pi_t = (a_t, o_t),\ \pi_t \in \Pi_g\} := \{o_1^s, \ldots, o_n^s\}$, where n is the number of unique objects in Π_g. Any object $o^s \in O_{seed}$ can be replaced with other objects, by considering semantic information such as location, category, and property. Replacement is allowed only if the substitute objects are located in the same room and share identical categories and properties with o^s. We denote the set of such potential replacement objects as $O_{aug}(o^s)$. This approach maintains contextual relevance and consistency with the original datapoint characteristics in augmented data. We then sample a set of objects $O' = \{o_1', \ldots, o_n'\}$, where each o_i' is sampled from $O_{aug}(o_i^s)$. The total number of distinct sampling combinations is given by $\prod_{i=1}^{n} |O_{aug}(o_i^s)|$. After sampling O', each o_i^s in Π_g is replaced with with o_i' to generate a new plan Π'. More formally, the revised plan is $\Pi' = \{\pi_1', \cdots, \pi_T'\}$, where $\pi_t' = (a_t, o_k')$ if $o_t = o_k^s$. The sampling process is iteratively repeated for N_{sample} times. In this paper, we set $N_{sample} = \min\left(\prod_{i=1}^{n} |O_{aug}(o_i^s)|, 20\right)$ in fifty different types of environments.

LLM Plan Verifier

In this section, we introduce the *LLM plan verifier*, a module that generates natural language instructions to explain augmented task plans and checks if the plans make sense for everyday household robotic tasks. For instance, as depicted in Figure 3, the given task plan is annotated in two formats: a goal instruction, *"Store the pound cake back in the fridge,"* and a step-by-step instruction, *"Go to the kitchen, open the fridge, grab the pound cake, put it back in the fridge, and close the fridge,"* respectively. However, not all augmented plans are practical or typical. For example, our *plan verifier* can exclude the task annotated as *"microwave candybar"* due to the fact that *"Microwaving a candy bar is not a typical household task, and could be potentially dangerous if it involves wrappers or ingredients that are unsuitable for microwaving."* This method enables the automatic exclusion of such unreasonable data points.

The *LLM plan verifier* employs the LLM foundation model in a few-shot learning method. The prompts designed with the Chain-of-Thought (CoT) method (Wei et al. 2022) generate both verification decisions and the reasoning behind them. Each prompt begins with a guidance sentence for summarizing and evaluating robot task plans. This is followed by few-shot examples that include a task plan, its natural language instruction I, and common sense verification results, labeled as *True* or *False* with reasoning, detailed in Appendix B. The dataset only includes output from the *LLM plan verifier* when the plan are verified as *True*. We employed the GPT-4o model, configured with three few-shot examples. We have generated a total number of 14k dataset including various categories such as object relocation, appliance operation, cooking foods, and object retrieval.

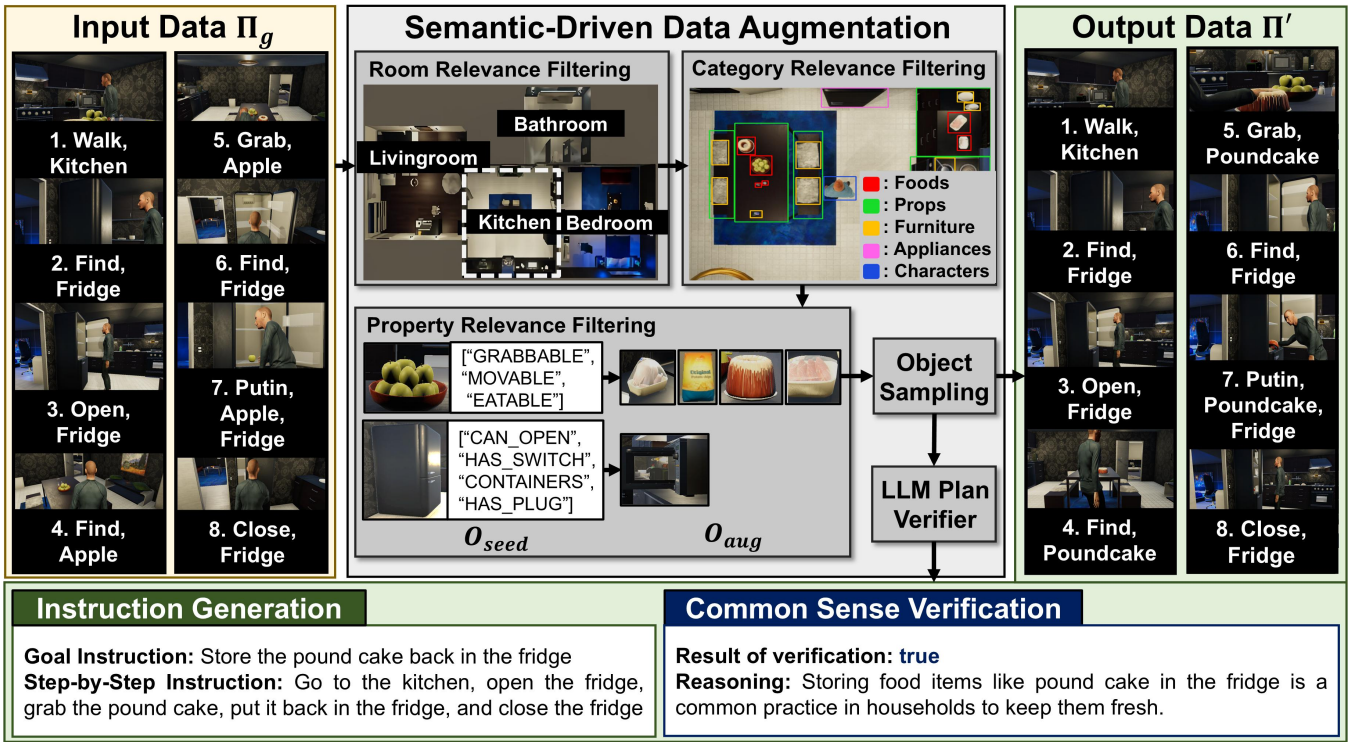

Figure 3: Overview of semantic-driven data augmentation. The input is a subgoal sequence for the task "put the apple in the fridge." Objects within these subgoals are substituted with others sharing the same room, category, and properties. The *LLM plan verifier* annotates and verifies the generated subgoals, ensuring the task is consistent with common sense.

Experimental Setup

Simulation Environment

We evaluate our baselines using the VirtualHome (VH) (Puig et al. 2018), a simulator have a human-like agents equipped with two hands, capable of executing daily household tasks. VH presents 50 different environments, each containing diverse combinations of interactable objects, up to 715 objects in each environment. We split the proposed dataset into four subsets: a training set, a validation set, a seen environment test set, and unseen environment test set. The unseen environment test set comprises data points from the five environments with the most distinct scene graph structures, based on the proposed *env-similarity score* detailed in Appendix C. we distributed the remaining data into training, validation, and seen environment test sets in a 0.7:0.15:0.15 ratio, ensuring no overlap of environment and instruction pairs between training and test sets. The dataset includes 10,054 for training, 1,918 for validation, 2,044 for seen tests, and 650 for unseen tests.

Baseline Models

We evaluate our model against a comprehensive set of baselines. Each leverages distinct methodologies within the realm of LLM and planning. ZERO-SHOT PLANNER (Huang et al. 2022) utilizes a pre-trained LLM in a zero-shot manner, which does not require fine-tuning on specific task data. PROGPROMPT (Singh et al. 2023) employs a pre-trained LLM with prompts formulated in a pythonic programming language. GOALNET (Gupta et al. 2024) is a neuro-symbolic model that merges neural network predictions with a classical symbolic planner. GRID (Ni et al. 2024) designed a GNN-based planner that divides graph observation into robot and scene graphs and plans with Graph Attention Networks. Our experiments utilize GPT-4O, GPT-4-TURBO (GPT-4-T) and GPT-3.5-TURBO-INSTRUCT (GPT-3.5-TI) as the backbone LLMs. For fair comparison, we provide our training data as a prompt for LLM-based planners for ZERO-SHOT PLANNER and PROGPROMPT, and we train GOALNET and GRID using our dataset.

Real World Environment

To evaluate our algorithm in a real-world setting, we used a mobile manipulator combining a Husky mobile base robot with a Franka Emika Panda arm. As shown in Figure 1, the environment consists of three rooms: a living room, a kitchen, and a bathroom. We manually constructed an initial scene graph with 26 objects, and interactable objects are tagged with an ArUco marker to detect their position. A hook inspired by Stretch3 design (Kemp et al. 2022) was attached for one-handed actions like opening cabinet doors. The robot control system is implemented on ROS2 humble environment with open-source packages. Moveit2 and NAV2 with Slam Toolbox are utilized for manipulation and navigation, respectively. Motion primitives are predefined and executed based on the position of target objects. We

Methods	LLM Backbone	Seen Environment				Unseen environment			
		%Exec (↑)	**%GCR** (↑)	**%SR** (↑)	**$Cost** (↓)	**%Exec** (↑)	**%GCR** (↑)	**%SR** (↑)	**$Cost** (↓)
ZERO-SHOT	GPT-4O	89.1±0.2	12.4±0.5	6.3±0.4	8.7±0.1	87.4±0.1	18.6±0.8	8.2±0.5	3.0±0.0
	GPT-4-T	91.0±0.3	49.4±0.4	36.3±0.6	74.8±0.6	94.2±0.4	55.4±0.2	40.1±0.5	22.7±0.2
	GPT-3.5-TI	83.7±0.2	60.2±0.17	44.8± 0.2	13.0±0.1	85.3±0.7	60.9± 0.6	46.0±1.0	3.5±0.0
PROGPROMPT	GPT-4O	65.4±1.1	54.3±1.6	30.2±1.8	24.6±0.1	62.6±0.0	48.3±2.3	20.2±2.1	7.6±0.0
	GPT-4-T	68.5±1.2	42.1±0.4	19.2±0.1	48.4±0.2	72.2±1.6	38.4±1.5	12.0±1.1	14.6±0.0
	GPT-3.5-TI	67.8±0.4	44.4±0.8	22.0±0.6	6.0±0.0	64.3±2.9	45.3±1.0	18.8±0.9	1.8±0.0
GOALNET	-	72.7	58.5	32.4	-	75.2	56.5	31.3	-
GRID	-	83.7	22.5	9.7	-	83.6	18.7	9.7	-
GTTP (Ours)	-	**99.9**	**76.1**	**62.9**	-	**100.0**	**75.8**	**57.7**	-

Table 1: Evaluation against baseline algorithms in the VH environment. The evaluation is conducted in both seen and unseen environments, where **seen** refers to environments included in the training set, and **unseen** refers to those not present in the training set. The best-performing model is highlighted in bold.

Instruction	Format	*Plan SR*(↑)	*GCR*(↑)	*SR*(↑)
(Task 1) *Turn off light switch in the bathroom*	*G*	0.4	1.0	1.0
Find the light switch in the bathroom and turn it off	*S*	-	0.5	0.0
(Task 2) *Store the bar soap in the bathroom cabinet*	*G*	0.8	1.0	1.0
Grab the bar soap, open the bathroom cabinet, place the soap inside, and close the cabinet	*S*	0.8	1.0	1.0
(Task 3) *Store the mug in the kitchen cabinet*	*G*	0.6	1.0	1.0
Grab the mug, open the kitchen cabinet, put the mug inside, and close the cabinet	*S*	0.6	1.0	1.0
(Task 4) *Move the cellphone to the kitchen table*	*G*	1.0	1.0	1.0
Grab the cellphone from the living room, walk to the kitchen, and place it on the kitchen table	*S*	1.0	1.0	1.0
(Task 5) *Turn off the table lamp in the kitchen*	*G*	-	0.0	0.0
Find the table lamp in the kitchen and turn it off	*S*	-	0.0	0.0
(Task 6) *Place the apple on the plate*	*G*	1.0	1.0	1.0
Grab the apple, find a plate, and place the apple on the plate	*S*	1.0	1.0	1.0
(Task 7) *Put the bananas on the desk in the living room*	*G*	1.0	1.0	1.0
Grab the bananas from the kitchen, walk to the living room, and put the bananas on the desk	*S*	1.0	1.0	1.0

Table 2: Real World Experimental results: The experiments include seven types of tasks, each presented in two formats: goal instruction (*G*) and step-by-step instruction (*S*). GCR and SR metrics indicate how accurately the GTTP predicts subgoals, while Plan SR reflects overall performance, including motion planning.

tested seven different tasks with both a goal instruction and a step-by-step instruction, as shown in Table 2. Each task was executed five times only if the planner accurately predicted the high-level subgoal sequence.

Evaluation Metrics

We evaluate our planner using four metrics: Executability (*Exec*), Goal Conditions Recall (*GCR*), Success Rate (*SR*), and LLM API Cost (*Cost*) (Singh et al. 2023). The *Exec* is the rate of an executable subgoals to predicted subgoals. The *GCR* is computed by first determining the symmetric difference between ground truth goal conditions and the predicted goal conditions, and then dividing this result by the total number of goal conditions. The *SR* indicates the proportion of scenarios that successfully achieved the goal. The *Cost* indicates the token efficiency, calculated with different prices per token for each LLM backbone. In real-world experiments, we employ an additional metric *Plan SR*, which evaluates the success rate of the entire subgoal execution, including the motion and path planning of the physical robot.

Results

The results of compared baselines in the VH environment are shown in Table 1. Our model achieved an 18.1% higher success rate than the best-performing baseline in seen environments. LLM-based planners, including ZERO-SHOT PLANNER and PROGPROMPT, underperform despite using our training data as prompts. This is because LLMs are not specifically designed for plan generation, and model enhancement is limited to prompt-based fine-tuning. Their performance heavily relies on the LLM Backbone and incorporating more examples to boost performance increases costs. Our model does not require additional costs, once trained. GOALNET and GRID, although trained with our training set, significantly underperform. It shows that the superiority of our architecture, which effectively handles complex environments and varied instructions.

Furthermore, our model shows robust performance in unseen environments. While LLM-based planners maintain consistent performance in both seen and unseen environments due to their high capacity for understanding new environments, they fail to achieve satisfactory performance overall. Although our GTTP do not directly utilize the LLM backbone for planning, our model effectively compre-

Methods	Short		Medium		Long	
	%GCR	%SR	%GCR	%SR	%GCR	%SR
ZERO-SHOT	60.3	53.3	61.0	43.0	59.6	41.6
PROGPROMPT	57.6	48.6	48.7	21.2	58.2	28.6
GOALNET	76.2	56.7	57.1	49.4	32.2	11.4
GRID	27.9	20.9	17.0	7.8	25.1	5.1
GTTP	73.7	72.6	87.2	75.0	66.1	44.7

Table 3: Comparative results relative to task plan length (l). Short, medium, and long task plans correspond to lengths l in the ranges of $1 \leq l \leq 6$, $7 \leq l \leq 12$, and $13 \leq l \leq 16$. We omit the variance due to space limitations.

Loss	Seen Env.		Unseen Env.		Noise	Seen Env.		Unseen Env.	
	GCR	SR	GCR	SR		GCR	SR	GCR	SR
30%	78.1	64.8	75.8	57.6	10%	78.2	59.7	74.9	54.6
60%	78.9	66.0	74.6	56.8	20%	75.7	58.6	74.7	54.4
90%	79.0	66.0	72.1	53.9	30%	74.2	58.3	74.5	54.0

Table 4: Robustness against information loss in scene graphs and noise in object states.

hends various descriptions, as our dataset includes substantial natural language descriptions. Interestingly, our method achieves over 99% in Exec, attributable to our subgoal selector's ability to choose only feasible subgoals.

Table 3 presents the performance relative to the length of ground truth task plans. GTTP consistently outperforms others, especially for medium-length task plans, where it shows even higher performance. Although GTTP's performance decreases significantly for long task plans, it still remains substantially higher compared to baseline models. Additionally, we evaluated robustness by randomly removing objects from full scene graphs in simulation (*node dropout*) or inducing errors in object state predictions. Our model maintained consistent performance under partial observability or noise of objects, suggesting resilience to real-world uncertainties shown in Table 4.

Ablation Studies

In order to analyze the effectiveness of the proposed architecture, we conducted ablation studies for three key components; *progress context fusion module*, *global update function ρ_u in GNN layers*, and *node dropout*. In Table 5, the *progress context fusion module* carries considerable results with about 50% in *GCR* and 45% in SR of enhancements in the seen environment. It shows that the planner can understand the relevance between instruction tokens and state tokens, which helps it grasp the historical context. Furthermore, the planning capability decreases about 11% for *SR*, when global update function ρ_u is omitted in *GNN layers*. This result indicates that injecting sentence embedding u into the *GNN layer* as a global feature helps extracting semantic embeddings for each node. Node dropout is essential for generalization. While it may not appear significant in simulations, node dropout is crucial in real-world experiments. Without it, performance declines markedly, about 64%, because the size of real-world scene graphs is much smaller than those in the dataset.

Model Ablations	Seen Env.		Unseen Env.		Real-world	
	GCR	SR	GCR	SR	GCR	SR
w/o Progress Context	25.7	17.6	37.5	22.9	17.9	7.1
w/o ρ_u	69.6	51.5	75.1	53.1	67.9	57.1
w/o Node Dropout	72.2	60.3	**80.9**	**60.6**	17.9	14.3
GTTP (Ours)	**76.1**	**62.9**	75.8	57.7	**82.1**	**78.6**

Table 5: Ablation Study of GTTP. It examines the contribution of each component to the overall performance of GTTP. Key components, including the progress context fusion module (Progress Context), the global update function ρ_u, and node dropout, are analyzed.

Real World Experiment

Figure 1 shows the execution of the plan for the instruction *"store the mug in the kitchen cabinet."* Despite the experimental environment differing significantly from those in simulations, it demonstrates successful sim-to-real transferability. Table 5 presents high GCR and SR in real-world settings, while Table 2 evaluates whether GTTP accurately predicts subgoals for each instruction. Failures are primarily associated with tasks like turning off the light, as the training dataset did not include a table lamp in the kitchen. Execution failures are raised in opening or closing the cabinets and turning off the light switch because of the locating error of the target object or even the robot itself in the environment. Despite these challenges, the results still indicate robust overall real-world performance. Qualitative results can be viewed in the supplementary video. To demonstrate applicability using the perception model, we set up smaller environments for tasks 4, 6, and 7 in Table 2 and employ GPT-4o to construct the scene graph. Although we do not have the ground-truth scene graph, we successfully execute 18 trials across three tasks (three trials per task), achieving an 83.3% success rate. Details are provided in Appendix F.

Conclusion

We propose a GNN-Transformer task planner designed for robot task planning with natural language instructions. Our architecture combines GNN layers to interpret semantic information from scenes and manage varying numbers of objects, with Transformer layers to account for historical processes. Additionally, we present a semantic-driven augmentation method that enables the construction of a large and high-quality training dataset with minimal human intervention. Experiments demonstrate that the proposed method significantly outperforms baseline algorithms across tasks of varying lengths and maintains robustness. Furthermore, our planner, trained solely on simulation data, proves effective in real-world settings. It exhibits excellent performance, scalability in complex environments, and effective sim-to-real transferability. In the future, we aim to refine our augmentation method to broaden the diversity of task plans and improve the verification process. Additionally, we plan to enhance the GTTP to account for robot affordances and support replanning in case of failures.

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2022-II220907, Development of AI Bots Collaboration Platform and Self-organizing AI), Alchemist Project Program (RS-2024-00422269, Interactive humanoid platform with transcendental sensory augmentation) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.RS-2020-II201373, Artificial Intelligence Graduate School Program (Hanyang University)), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2024-00409492, Beyond-G Global Innovation Center).

References

Ahn, M.; Dwibedi, D.; Finn, C.; Arenas, M. G.; Gopalakrishnan, K.; Hausman, K.; Ichter, B.; Irpan, A.; Joshi, N. J.; Julian, R.; Kirmani, S.; Leal, I.; Lee, T.-W. E.; Levine, S.; Lu, Y.; sharath maddineni; Rao, K.; Sadigh, D.; Sanketi, P. R.; Sermanet, P.; Vuong, Q.; Welker, S.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; and Xu, Z. 2024. AutoRT: Embodied Foundation Models for Large Scale Orchestration of Robotic Agents. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

Battaglia, P. W.; Hamrick, J. B.; Bapst, V.; Sanchez-Gonzalez, A.; Zambaldi, V. F.; Malinowski, M.; Tacchetti, A.; Raposo, D.; Santoro, A.; Faulkner, R.; Gülçehre, Ç.; Song, H. F.; Ballard, A. J.; Gilmer, J.; Dahl, G. E.; Vaswani, A.; Allen, K. R.; Nash, C.; Langston, V.; Dyer, C.; Heess, N.; Wierstra, D.; Kohli, P.; Botvinick, M. M.; Vinyals, O.; Li, Y.; and Pascanu, R. 2018. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261.

Chen, H.; Tan, H.; Kuntz, A.; Bansal, M.; and Alterovitz, R. 2020. Enabling robots to understand incomplete natural language instructions using commonsense reasoning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 1963–1969. IEEE.

Gupta, J.; Sharma, S.; Tuli, S.; Paul, R.; and Mausam. 2024. GOALNET: Interleaving Neural Goal Predicate Inference with Classical Planning for Generalization in Robot Instruction Following. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18): 20113–20122.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. In *International Conference on Machine Learning, ICML*, volume 162, 9118–9147.

Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J. J. R.; Mordatch, I.; Chebotar, Y.; Sermanet, P.; Brown, N.; Jackson, T.; Luu, L.; Levine, S.; Hausman, K.; and Ichter, B. A. 2023. InnerMonologue: Embodied Reasoning through Planning with Language Models. In *Conference on Robot Learning*.

Ichter, B.; Brohan, A.; Chebotar, Y.; Finn, C.; Hausman, K.; Herzog, A.; Ho, D.; Ibarz, J.; Irpan, A.; Jang, E.; Julian, R.; Kalashnikov, D.; Levine, S.; Lu, Y.; Parada, C.; Rao, K.; Sermanet, P.; Toshev, A. T.; Vanhoucke, V.; Xia, F.; Xiao, T.; Xu, P.; Yan, M.; Brown, N.; Ahn, M.; Cortes, O.; Sievers, N.; Tan, C.; Xu, S.; Reyes, D.; Rettinghouse, J.; Quiambao, J.; Pastor, P.; Luu, L.; Lee, K.-H.; Kuang, Y.; Jesmonth, S.; Jeffrey, K.; Ruano, R. J.; Hsu, J.; Gopalakrishnan, K.; David, B.; Zeng, A.; and Fu, C. K. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. In *Annual Conference on Robot Learning*.

Kemp, C. C.; Edsinger, A.; Clever, H. M.; and Matulevich, B. 2022. The Design of Stretch: A Compact, Lightweight Mobile Manipulator for Indoor Human Environments. In *2022 International Conference on Robotics and Automation (ICRA)*, 3150–3157.

Ni, Z.; Deng, X.; Tai, C.; Zhu, X.; Xie, Q.; Huang, W.; Wu, X.; and Zeng, L. 2024. GRID: Scene-Graph-based Instruction-driven Robotic Task Planning. arXiv:2309.07726.

Nyga, D.; Roy, S.; Paul, R.; Park, D.; Pomarlan, M.; Beetz, M.; and Roy, N. 2018. Grounding Robot Plans from Natural Language Instructions with Incomplete World Knowledge. In *Proceedings of Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, 714–723.

Puig, X.; Ra, K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8494–8502.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 11523–11530.

Sun, J.; Huang, D.-A.; Lu, B.; Liu, Y.-H.; Zhou, B.; and Garg, A. 2022. PlaTe: Visually-Grounded Planning With Transformers in Procedural Tasks. *IEEE Robotics and Automation Letters*, 7(2): 4924–4930.

Traum, D. R.; Henry, C.; Lukin, S. M.; Artstein, R.; Gervits, F.; Pollard, K. A.; Bonial, C.; Lei, S.; Voss, C. R.; Marge, M.; Hayes, C. J.; and Hill, S. G. 2018. Dialogue Structure Annotation for Multi-Floor Interaction. In *International Conference on Language Resources and Evaluation*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. arXiv:1706.03762.

Wächter, M.; Ovchinnikova, E.; Wittenbeck, V.; Kaiser, P.; Szedmak, S.; Mustafa, W.; Kraft, D.; Krüger, N.; Piater, J.;

and Asfour, T. 2018. Integrating multi-purpose natural language understanding, robot's memory, and symbolic planning for task execution in humanoid robots. *Robotics and Autonomous Systems*, 99: 148–165.

Wang, L.; Ling, Y.; Yuan, Z.; Shridhar, M.; Bao, C.; Qin, Y.; Wang, B.; Xu, H.; and Wang, X. 2024. GenSim: Generating Robotic Simulation Tasks via Large Language Models. In *The International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; ichter, b.; Xia, F.; Chi, E.; Le, Q. V.; and Zhou, D. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35, 24824–24837. Curran Associates, Inc.

Zeng, A.; Attarian, M.; brian ichter; Choromanski, K. M.; Wong, A.; Welker, S.; Tombari, F.; Purohit, A.; Ryoo, M. S.; Sindhwani, V.; Lee, J.; Vanhoucke, V.; and Florence, P. 2023. Socratic Models: Composing Zero-Shot Multimodal Reasoning with Language. In *The International Conference on Learning Representations*.

Zhang, J.; Pertsch, K.; Zhang, J.; and Lim, J. J. 2024. SPRINT: Scalable Policy Pre-Training via Language Instruction Relabeling.

Zhao, Z.; Lee, W. S.; and Hsu, D. 2023. Large Language Models as Commonsense Knowledge for Large-Scale Task Planning. In *Advances in Neural Information Processing Systems*, volume 36, 31967–31987. Curran Associates, Inc.

Zhu, Y.; Tremblay, J.; Birchfield, S.; and Zhu, Y. 2021. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.